Delay-time modelling and critical-path verification for CMOS digital designs

K H Kim and S B Park

A new switch-level critical-path verifier for CMOS digital circuits is presented. The signal flow through MOS transistors is determined by combining the designer's tag with a set of direction-derivation rules, and the delays are evaluated on a stage-by-stage basis. The paper uses a semianalytic CMOS delay-time model that takes into account the configuration ratio, the input waveform slope and the load condition. This model is derived from the optimally weighted switching peak current, which is, in turn, derived from the Shichman–Hodges' DC equations. The delay equations are computationally effective, and the error is found to be typically within 10% of the SPICE results. A novel methodology for reporting multiple paths, called a modified depth-first search with predictor, is investigated. In comparison with the conventional method without predictor, it requires much less CPU time.

computer-aided design, CMOS delay time, semianalytic model, critical path, design verification

As the scale of integration increases rapidly with the progress of semiconductor technology, computer-aided design has become indispensable for design verification. Timing verification, the so-called critical-path verification, is perhaps one of the most important aspects of verification, because the maximum operation speed in a synchronous digital system is limited by the slowest path of all of its possible signal propagation paths, which is called the critical path. Even if the physical design is perfect, the electrical performance of the design may be insufficient, or even incorrect. Simulation on all levels[1–4] is still used to a large extent for this purpose; however, the usual approach is to use static timing-verification tools[5–7], which involve identifying critical paths and estimating propagation delays along these paths.

The results of conventional simulators are only as good as the test vectors used to obtain the results. If some portions of the circuit have not been probed by the test vectors, then the bugs in the unprobed portion of the circuit may remain undetected until the circuit is actually fabricated. Unfortunately, the number of possible input patterns can grow exponentially with the complexity of the circuit. Even if the input vector causing the critical path is found, it is difficult to locate the corresponding path, because there may be a number of signal propagation paths between the input and the output. The critical-path verifier, by probing the circuit for all possible combinations of inputs, ensures that every part of the circuit is exercised and tested for possible faults. The model for critical-path delay evaluation must be simple enough to be computationally efficient compared with SPICE[1], and yet simultaneously maintain a reasonable accuracy. Owing to its test completeness and cost effectiveness, critical-path verification is preferred for the timing designs of CMOS VLSI systems.

An improved modelling of the delay times is the key problem in performing fast and relatively accurate timing analysis of MOS logic circuits. Some more precise delay models for CMOS logic have been proposed in the past that are based on either the step response[8,9] or the ramp response[10,11].

A precise timing delay is mainly determined by such factors as the process parameters, circuit structure, transistor size, loading capacitance, and input waveform shapes. These factors constitute a set of nonlinear timing-delay equations that, in general, requires the use of iterative methods for the solution. The rigorous approach to delay-time evaluation is to use SPICE, which is the most accurate, but is computationally intensive. It models the circuit by a set of differential equations, which are then solved numerically. Unfortunately, the circuit simulators are impractical for today's VLSI circuits. The second method for the computation of delay time uses the switch-level RC model[12,13], which treats each transistor as a perfect switch in series with a resistor. It uses tables to compute the resistance; therefore, small tables produce inaccurate results if there is much variation in the circuit parameters. On the other hand, the empirical delay models[14] are reliable in accuracy and extremely efficient in speed. They determine the delay's dependence on the circuit parameters by numerically simulating many circuits and performing curve fitting on the results. This

Department of Electrical Engineering, Korea Advanced Institute of Science & Technology, PO Box 150, Chongyangni, Seoul 130–650, South Korea

Paper received: 11 June 1990. Revised: 10 September 1990

 0010–4485/91/090604–11

technique is dis-advantageous, owing to the lack of error control and difficulty in relating the delay back to the circuit elements.

This paper proposes a switch-level CMOS delay-time model that can be explicitly represented in terms of the process parameters, device sizes, input waveform slope and load capacitance. This semianalytic delay model takes advantage of the simplicity of a table RC model and the accuracy of an empirical model. Being the most frequently called section of the source code in the verifier, the delay-calculation routine is a crucial factor in the speed of the proposed verifier (called the KAIST Critical Path Verifier, or KCPV).

Most of the existing MOS digital-circuit timing tools construct a graph to model the design, and seek to locate the critical path. The problem of examining paths to locate the critical path can be grouped into two main categories[15]: the path-enumeration technique, and the critical-path analysis technique. The path-enumeration method, which finds all the paths in the circuits, tends to have a long computation time, as the number of paths to be examined grows exponentially with the number of the circuit branches. Therefore, this technique is inadequate for large combinational logic circuits. However, the availability of an entire path is often advantageous in the analysis stage. It is also easier to tell such a path-oriented method to ignore false paths that will never be activated in the real circuit, as the path may be masked before analysis, or it can be neglected after analysis. The critical-path method, sometimes called the block-oriented method, finds only the worst time in which the signal could propagate through the block. This second approach tends to require much less computation time than the enumerative path-analysis approach, and hence it is more suitable for large circuits.

However, reporting only one critical path to the designer often fails to give user-informative paths, for the following two reasons[6]. First, even if the most-critical path of a circuit is known, the optimal way to improve the performance may not be clear, because the designer will not know if the design can be speeded up without a requirement for major changes. Second, if the reported path is a false one, the critical-path verification provides invalid information. As a single pointer to the worst predecessor is not intrinsic to critical-path analysis, a new algorithm has been developed that can report the multiple critical paths (the most critical path and the successively less critical path) to the designer.

A new scheme for a switch-level CMOS delay-time evaluation is given in the second section of this paper. First, the delay estimation based on the RC model is discussed. Then, the semianalytic delay model is derived on the basis of the DC characteristic equations, and is corrected by the weighting function. A comparison between the propagation delays of SPICE and the new model is made to verify the reliability of the latter. Several aspects of the critical-path verification techniques are described in the third section of the paper. A method for finding the direction of the signal flow through MOS transistors, and a modified depth-first search algorithm for reporting the multiple critical paths, are given. Also, the stage that is a path leading from a signal source through transistor channels and other nodes to a target is explained, and the issues associated with path delay computation are described. The new KCPV CAD tool has been tested on a number of example circuits, and the experimental results are given in the fourth section of the paper, followed by the conclusions in the fifth section.

SWITCH-LEVEL CMOS DELAY-TIME MODEL

Delay estimation by RC model

Penfield and Rubinstein developed a method to bound the delays through an arbitrary resistance–capacitance tree[12]. The exact calculation of the signal delays of such networks is difficult, but calculation of the upper and lower bounds for the delay is computationally simple. The application of these bounds to MOS designs requires the modelling of all of the MOS transistors as linear resistors.

The determination of the delay through the RC tree gives the delay of the particular stage that is to be used in a critical-path verification programme. This provided the first formal model for delay estimation in MOS integrated circuits (ICs). This technique has the following two advantages over the empirical models. First, it can be applied to any type of transistor group, and so a separate model for each type of cluster is no longer needed. In addition, it can relate the delay back to the circuit, giving information as to which portions need improvement. However, the linear RC tree model has several limitations. As transistors are not linear devices, their effective resistance values must be determined empirically. Although the timing model produces the bounds, these waveforms only bound the output of the ideal linear model, and not the output of the nonlinear circuit. Finally, as the error in the estimated delay can be large, the resulting accuracy is dubious.

A very rough estimate of the signal propagation delay in the RC tree is given by the product $R_{tot}C_{tot}$, where R_{tot} and C_{tot} are the sums of all the resistance and capacitance values, respectively. This is necessarily crude, because, among other problems, it is independent of the choice of output nodes; in practice, it usually overestimates the delay. The Elmore time constant, defined as

$$T_{Di} = \sum_i R_i C_i \tag{1}$$

where R_i is the resistance from the source to load i, and C_i is the capacitance at point i, is generally better[13], and a more accurate estimate can be fashioned from T_{Di}.

The rationale for using T_{Di} as a delay estimate is that T_{Di} is the first moment, or mean, of the impulse response, provided that the capacitors are initially discharged[16]. If $V_i(t)$ is the zero-state step response, then $\dot{V}_i$ is the impulse response, and

$$T_{Di} = \int_0^\infty t\dot{V}_i(t)\,dt \tag{2}$$

Thus, the approximation of the delay by T_{Di} can be viewed as a method of data reduction that is equivalent to replacing a complete probability-density function by its mean value.

Delay-time formulation

The approach that varies most from tool to tool is the definition and implementation of the delay of a circuit block. This is directly related to the accuracy achievable in a tool. In general, the more complex the delay calculation is, the greater accuracy is achievable. This paper is concerned with a calculation model for the circuit delay that is simple and is as accurate as those models used in more complex methods. To guarantee a positive delay, it is assumed that the low and high voltage thresholds V_{TL} and V_{TH} are equal to V_{TN} and $V_{DD} - |V_{TP}|$, respectively, where $V_{TN(TP)}$ is the zero-bias threshold voltage of NMOS (PMOS) transistors in the CMOS circuit. Various delays of a CMOS inverter are defined below (see Figure 1).

- The fall (rise) offset time $t_{HLO(LHO)}$ is the time interval from the instant when the rising (falling) input reaches $V_{TL(TH)}$ to the instant when the falling (rising) output reaches $V_{TH(TL)}$.
- The fall (rise) time $t_{HL(LH)}$ is the time interval during which the falling (rising) output decreases (increases from $V_{TH(TL)}$ to $V_{TL(TH)}$.
- The fall (rise) propagation time $t_{PHL(PLH)}$ is defined between the V_{INV} crosspoints of the input and output waveforms, where V_{INV} is the logic inversion voltage.

In the dynamic behaviour of a CMOS inverter, the current waveforms are very much like triangular[10,17] waveforms, and current imbalance is practically negligible before either of the N or P switches is cut off by the input drive[9–11,17]. Therefore, if it is assumed that the load charge $C_l V_{DD}$ is discharged (charged) by an average current $WI_{pn(pp)}$, then, from the definition of the fall (rise) time,

$$t_{HL(LH)} = \frac{C_l(V_{DD} - V_{TN} - |V_{TP}|)}{WI_{pn(pp)}} \tag{3}$$

where C_l is the lumped output capacitance, V_{DD} is the supply voltage, $I_{pn(pp)}$ is the NMOS (PMOS) transistor peak current, and W is a weighting factor ($W = 0.5$ if the current waveform is exactly triangular). From Equation 3, it can be seen that $t_{HL(LH)}$ can be calculated once $I_{pn(pp)}$ and W have been found. In the following, only the case of the rising input $V_{in} = V_{DD}t/\tau_{LH}$ $(0 < t < \tau_{LH})$ is considered, where τ_{LH} is the input-rising duration. (The case of falling input can be analysed in a similar way.)

If the model does not use tables or empirical equations, crucial consideration should be given to the fact that delay values need to be analysed in the time interval during which the MOS transistor is switching. Let V_s and V_f be the output voltages when the transistor leaves saturation and when the input voltage reaches its final value, respectively. Then, the following two cases arise (see also Figure 2).

Case 1: fast input ($V_s \leqslant V_f$)

The transistor is still saturated when the input ramp reaches its final value. In this case, the differential equation for the output voltage $V_o(> V_f)$ is

$$-C_l\frac{dV_o}{dt} = \frac{k_N}{2}\left(\frac{V_{DD}t}{\tau_{LH}} - V_{TN}\right)^2 \tag{4}$$

where k_N is the N-channel transistor constant. With the initial condition $V_o = V_{DD}$ at $V_{in} = V_{TN}$, integration yields

$$V_o = V_{DD} - \frac{k_N\tau_{LH}}{6C_lV_{DD}}\left(\frac{V_{DD}t}{\tau_{LH}} - V_{TN}\right)^3 \tag{5}$$

When the input reaches its final value, the condition $V_s \leqslant V_f$ yields

$$\tau_{LH} \leqslant \frac{6C_lV_{DD}V_{TN}}{k_N(V_{DD} - V_{TN})^3} \tag{6}$$

While V_{in} is kept at V_{DD}, and the N-channel transistor is still saturated, the output voltage is

$$V_o = V_f - \frac{k_N(V_{DD} - V_{TN})^2(t - \tau_{LH})}{2C_l} \tag{7}$$

Letting $V_o = V_s = V_{DD} - V_{TN}$,

$$t_{HLO} = \tau_{LH} + \frac{2V_{TN}C_l}{k_N(V_{DD} - V_{TN})^2} - \frac{\tau_{LH}(V_{DD} - V_{TN})}{3V_{DD}} \tag{8}$$

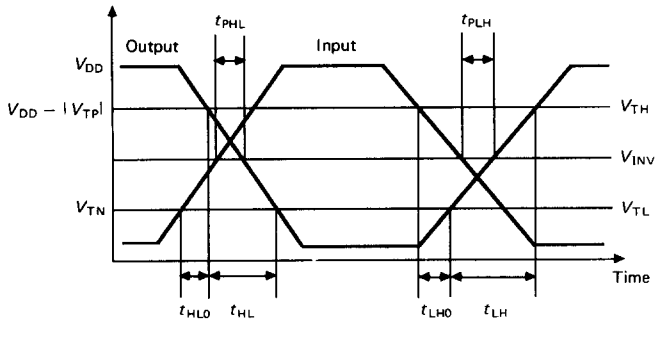

Figure 1. Definitions of delay times

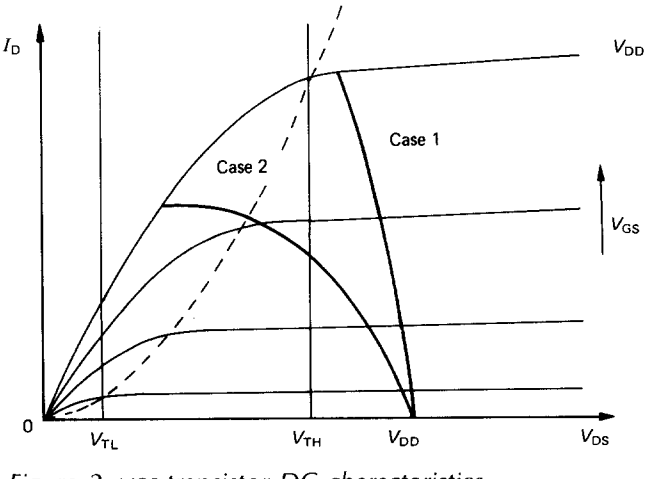

Figure 2. MOS-transistor DC characteristics

is obtained. In this case, I_{pn} equals the N-channel saturation current with the gate-source voltage V_{GS} equal to V_{DD}, i.e.

$$I_{pn} = \frac{k_N}{2}(V_{DD} - V_{TN})^2 \tag{9}$$

Case 2: slow input ($V_s > V_t$)

The N-channel transistor leaves saturation while the input voltage is still increasing linearly. Equation 5 is valid in the region of $V_o > V_s$, and the substitution of $V_o = V_{DD} - V_{TN}$ and $t = t_{HLO}$ yields

$$t_{HLO} = \frac{\tau_{LH}}{V_{DD}}\left(V_{TN} + \left(\frac{6C_l V_{DD} V_{TN}}{k_N \tau_{LH}}\right)^{1/3}\right) \tag{10}$$

The value of V_s is determined by substituting $V_o = V_s = V_{DD}t/\tau_{LH} - V_{TN}$ into Equation 5, i.e.

$$V_s = V_{DD} - \frac{k_N \tau_{LH}}{6C_l V_{DD}} V_s^3 \tag{11}$$

The substitution of the input ramp into the linear-region equation results in a rather complicated nonlinear differential equation. Thus, simplification is achieved by the setting of V_{GS} to a constant value V_{avg}, where V_{avg} is the average of the gate voltage at the saturation point and V_{DD}, i.e. $V_{avg} = ((V_s + V_{TN}) + V_{DD})/2$. As the transistor is in the linear region, and the maximum current occurs at $V_o = V_s$,

$$I_{pn} = k_N\left((V_{avg} - V_{TN})V_s - \frac{V_s^2}{2}\right) \tag{12}$$

is obtained. It can easily be proved that, in the limit for which $V_s = V_{DD} - V_{TN}$, Equation 12 is reduced to Equation 9, the peak current of Case 1. On the other hand, t_{PHL} is derived from the definition and Equations 3, 8 and 9, or Equations 3, 10 and 12, i.e.

$$t_{PHL} = \frac{\tau_{LH} V_{TN}}{V_{DD}} + t_{HLO} + t_{HL}\frac{V_{DD} - |V_{TP}| - V_{INV}}{V_{DD} - V_{TN} - |V_{TP}|} - \tau_{LH}\frac{V_{INV}}{V_{DD}} \tag{13}$$

Therefore, from the relationship $t_{PHL} = R_{eff(N)}C_l$

$$R_{eff(N)} = \frac{t_{PHL}}{C_l} \tag{14}$$

where $R_{eff(N)}$ is an effective linear resistance of NMOS transistors. Similar expressions may be derived for t_{LHO}, I_{pp}, t_{PLH} and $R_{eff(P)}$ by solving the corresponding differential equations for the P-channel transistor with negative input ramps.

Compensation by optimized weighting function

To compensate for the expected error due to the assumptions made, a weighting factor for the delay for a given process technology is introduced that will be determined through extensive computer simulation. For several delays, the appropriate weighting factors are given in Table 1. For $t_{PHL(PLH)}$, the derived Equation 13 is multiplied by the nonlinear weighting factors. All W_is, $i = 1, 2, 3, 4, 5, 6, 7$, are positive real constants. For negative input ramps, when the output is becoming high, the same forms of weighting factors should be used. The Fletcher–Powell technique[18] is used in determining W_is that minimize the sum of the normalized squared errors

$$\sum_{n=1}^{M} \varepsilon_n^2 = \sum_{n=1}^{M} (t_n - t_n^*)^2/t_n^2$$

where t_n and t_n^* are the delay times calculated by SPICE and by the proposed model, respectively, for a large number M of combinations of C_l, $\beta_{N(P)}$ and $\tau_{LH(HL)}$, where $\beta_{N(P)}$ is the NMOS (PMOS) configuration ratio. The delay values calculated from the derived equations are compared with those obtained from SPICE for an inverter chain for a wide range of $\beta_N(4-8)$, $\beta_P(10-14)$, $\tau_{LH(HL)}(0.1-5.0\,ns)$, and $C_l(0.1-1.0\,pF)$, for which the optimized W_is are shown in Table 2. The results are shown in Figure 3, which shows the SPICE delay (on the horizontal axis) and the proposed delay (on the vertical axis) with the same scale. The fact that all the data points are clustered close to the diagonal line indicates that the proposed model is very accurate.

Table 1. Form of weighting factors

Delay times		Weighting functions
$t_{HL(LH)}$	Fast case	W_1
	Slow case	$W_2\left(\frac{C}{\beta\tau}\right)^{W_3}$
$t_{PHL(PLH)}$	Fast case	$W_4(\beta\tau C)^{-W_5}$
	Slow case	$W_6(\beta\tau C)^{-W_7}$

Table 2. Optimized W_is in 2 μm CMOS process

Coefficients	Falling delay	Rising delay
W_1	0.44211	0.44055
W_2	0.68568	0.67566
W_3	0.11696	0.09528
W_4	1.13890	1.04390
W_5	0.06156	0.05331
W_6	1.00260	0.93219
W_7	0.04781	0.04289

[$V_{TN} = 0.75$ V, $V_{TP} = -0.83$ V, $T_{OX} = 380$ Å, $K_{PN} = 56\,\mu AV^{-2}$, $K_{PP} = 22\,\mu AV^{-2}$.]

CRITICAL-PATH VERIFICATION TECHNIQUES

Signal-flow determination

Paths in a circuit are identified by following the signal-flow directions assigned to each transistor. A signal is a logical quantity, and not current or electrons. Signals can be generated by power-supply rails and the inputs to the circuit. The logical signals at these sources are propagated into the circuit, i.e. its output and the gates of transistors. The logic '0' flows in the opposite direction to the current flow, while the logic '1' signal

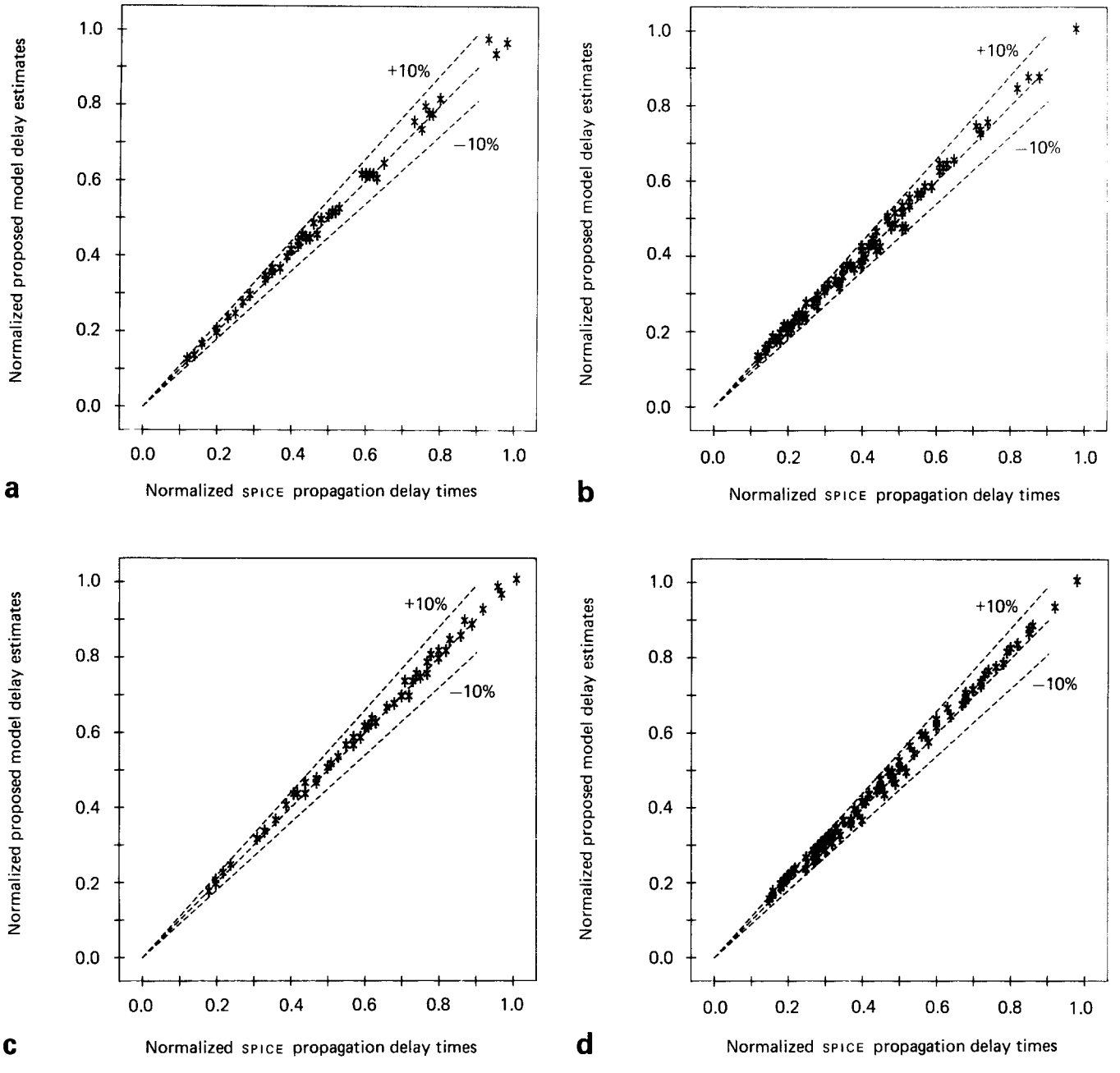

Figure 3. Normalized propagation-delay comparisons between simulation results of SPICE and proposed method; (a) t_{PHL} (fast case), (b) t_{PHL} (slow case), (c) t_{PLH} (fast case), (d) t_{PLH} (slow case)

flows in the same direction. In the case of the inverter's discharge, the current flows from the output node to ground through a pull-down transistor, but the logic '0' signal flows from ground to the output node. Transistors conduct signals to their outputs depending on the signal controlling their gate. In some circuits, transistors change their input and output terminals, and inputs and outputs may play interchangeable roles. These transistors are called bidirectional. An example of a bidirectional transistor can be found in a RISC barrel shifter[19]. In a typical design, however, most transistors are unidirectional such that signals flow only in one direction during operation, and only a small number of MOS transistors are bidirectional. Without correct signal-flow determination, the false paths that cannot be activated under real operating conditions are produced from incorrectly assumed bidirectionality. Therefore, these paths tend to be slower than real critical paths. This prevents the critical-path verifier from reporting the valid paths. Thus, the proper signal-flow determination is crucial in switch-level CMOS critical-path verification.

Two possible approaches exist for finding the proper signal flow. The one solution requires the designer to tag all transistors with direction flags[5]. This complicates the task of the designer, but it is possible to assign the signal flow correctly. Although the amount of work required by the designer is very large, the actual number of tags can be much lower, as cells can be replicated or created by design synthesis tools in many cases. Another approach relies on a set of direction-derivation rules. This eases the designer's task, but has proved somewhat restrictive. Recently, some switch-level timing verifiers, such as TV[6] and Pearl[7], have pursued

the second method. For example, TV applies two sets of rules for deriving the direction of the signal flow in MOS circuits, i.e. the safe and unsafe rules. However, its rules for NMOS circuits depend on the transistor ratios to find pass transistors, rendering them virtually unusable for CMOS designs. Therefore, Pearl attempts to prove that flow is impossible from either the source to the drain or *vice versa* by searching for transistor gate inputs, feedback paths, and power sources up to the maximum search depth. Unless the search depth is limited to a reasonable level, the program cannot avoid getting stuck in circuits that require inordinate amounts of computation.

KCPV combines the user-input tagging with the following three kinds of rules, where the use of the latter rules reduces the extra burden of tagging for most transistors, and the former can set ambiguous transistors. Therefore, the signal flow can be derived efficiently and correctly. It should be noted that the user's tagging has precedence over the three signal-flow derivation rules.

The fundamental rules used for KCPV are as follows.

- *Source rule:* Nodes with a voltage source or ground are strong source nodes. This fact is used to determine the signal flow from these source nodes to the crosspoint of the NMOS pull-down part and the PMOS pull-up part on the basis of the characteristic of the CMOS logic structure. Because many transistors are connected to the supply rails, this rule is sufficient to assign a signal-flow direction to the majority of the transistors in a typical circuit. (Rule 1.)
- *KCL rule:* If all but one of the MOS transistors connected to the given node *N* are set, and the set transistors are all sinks (sources) for node *N*, the logic signal must enter (leave) the unset transistor, because the node is not an information sink (source). (Rule 2.)
- *From-gate rule:* This rule tries to search for the source in the reverse direction, starting from the gate node. During the trace process, if there is any transistor whose flow has been determined, the signal-flow direction of the MOSFETS on the search path is decided according to the set transistor's signal direction. (Rule 3.)

KCPV sets the signal flow through unset transistors by applying Rules 1, 2 and 3, sequentially. An example of signal-flow direction assignment is shown in Figure 4. In particular, a CMOS transmission gate that consists of an N-channel transistor and a P-channel transistor with mutually exclusively controlled gate connections and common source and drain connections is identified in the input file-reading phase, and is viewed as one pseudotransistor, when Rules 2 and 3 are applied.

Multiple-path selection method

The critical-path verifier traces possible paths to each node in a circuit, and the delay times along the paths are calculated and stored. After the path search has been completed, the accumulated times along the paths are provided to the designer in descending or ascending order. Most of the earlier work is based on

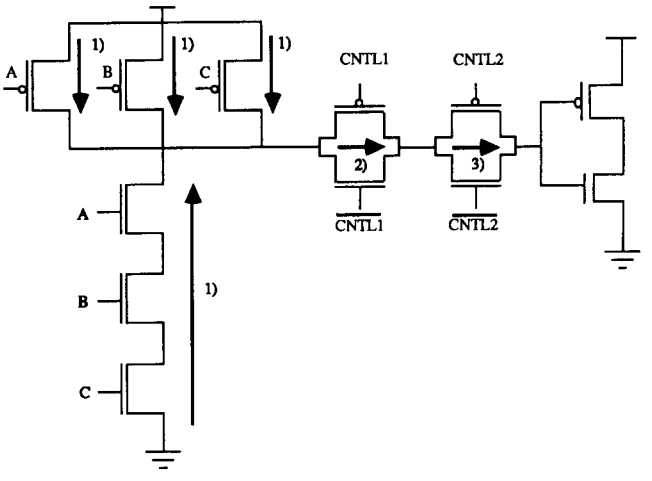

Figure 4. Example of signal-flow direction determination

one of the following two approaches for this purpose: the modified depth-first search[5], and the modified breadth-first search[6].

As its name implies, the modified depth-first search is a variation of the depth-first search (DFS). The plain DFS[20] traces all possible paths through the graph, and then computes the total delay along each path. This technique suffers from a severe problem of path explosion. As an example, a DFS visits nodes in the solid-line graph shown in Figure 5 as follows:

A—C—E—G—H—I—J—(I)—(H)—(G)—(E)—(C)—F—(C)—(A)—B—D—(B)—(A) (15)

where the nodes in parentheses represent a backtracking. In a modified DFS, whenever a node is revisited during the course of a search, the new arrival time is propagated to the child nodes only if it is worse than the previous worst arrival time at that node. When an algorithm based on a DFS with pruning examines path A—C—E—G—H—I, the other part of nodes I to F has not yet been searched. Thus, the path verifier allocates the path A—C—E—G—H—I as the critical path leading to node I. Similarly, when node J is visited next, path A—C—E—G—H—I—J is labelled as its slowest path. After tracing back to C from J, this method visits F, and then I again through a new path A—C—F—I. At this moment, it compares a new path delay with an old path delay of node I. If the new path delay is smaller than the old one, the verifier traces back and visits nodes B–D; otherwise, the critical path to I is updated by the new path A—C—F—I. In the latter case, node J, which is a child of I, is visited again

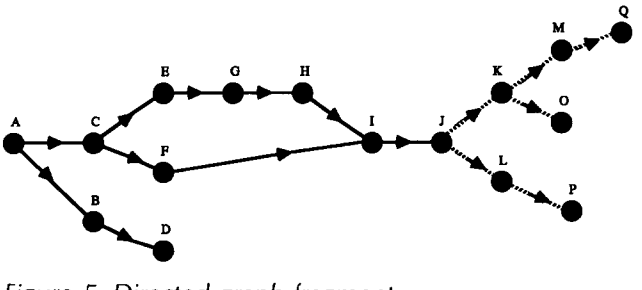

Figure 5. Directed-graph fragment

to update the new critical path. This can result in the same node being processed several times, with its arrival time being updated. The computational complexity for the best and worst cases in the modified DFS algorithm is linear and exponential with the number of edges, respectively.

An alternative to the DFS is the breadth-first search (BFS), which does not process a node until all the nodes that can drive it have been completely visited. For the directed solid-line graph shown in Figure 5, the BFS visits nodes as follows:

$$A—(B, C)—(D, E, F)—(G, I)–(H, J) \quad (16)$$

where the nodes in parentheses can be visited in any order. From the above example, the BFS visits I before H, but no information is available on the path coming from node H until node I is visited. Thus, the modified BFS, which visits the child node only if all of its parent nodes have been visited, is used in TV for finding the critical paths. This approach, the so-called level-based analysis, visits nodes from the lowest to the highest level, where the level of a node is defined such that the level of a start point is 0, and that of the other nodes is equal to the maximum of the levels of its parents plus 1. The search sequence of the solid-line graph shown in Figure 5 proceeds as follows:

$$A—(B, C)—(D, E, F)—G—H—I—J \quad (17)$$

As each node is processed only once, the execution time is of linear complexity. The disadvantage of the BFS approach is that it can be used only in circuits without cycles. Note that the DFS handles the loop easily. Also, a modified DFS selects an edge to cut dynamically, while the modified BFS selects it statically. In this paper, to reduce the execution time and to report the multiple paths, a new algorithm, called the modified DFS with predictor, is proposed. For example, the DFS used in Figure 5 has the following three paths through node J:

$$\begin{aligned} &A—C—E—G—H—I—J—K—M—Q \\ &A—C—E—G—H—I—J—K—O \\ &A—C—E—G—H—I—J—L—P \end{aligned} \quad (18)$$

Suppose that it is required to inform the three paths, the path buffer storing the path information is already full, and the smallest delay time is set to the threshold value. Then, by backtracking and forward stepping, path A—C—F—I—J is visited one node at a time. At this time, the algorithm predicts the maximum arrival time and compares the threshold time to the computed value. If the new path delay is smaller than any of the stored multiple-path delays in the past, the path search starting from node J is no longer performed. The maximum arrival time is calculated via the sum of the arrival time of the new path and the maximum elapsed time from the evaluated node.

Stages and delay-time computation

To compute the path delay in the switch-level approach, the critical-path verifier extracts linear chains of transistors leading to the node; this is called a stage[5,7]. A stage is defined as a path leading from the signal source through transistor channels and other nodes to a target. When a stage is triggered, the output of the stage is charged (discharged) to the power (ground) rail at the source of the stage. One advantage of using stages as a basis for critical-path verification is that they can uniformly handle both normal gates and pass transistors. KCPV finds stages by searching from the source (drain) node of the transistor until power or ground is encountered, and from the drain (source) node until a node connected to the transistor gates or chip outputs is encountered. Then, if the target is a gate, KCPV repeats the whole analysis recursively by finding other stages that the gate change might activate. Each of the stages is passed to a delay-evaluation routine that computes the time that it takes for the output of the stage to change values. The distributed RC delay model[13] is used to calculate delays through the transistor chains in which nonlinear MOS transistors are replaced by linear resistors using the method proposed in the second section of this paper. For example, if a logic '1' signal is applied to node A as shown in Figure 6a, the delay value T at the output node through the equivalent RC chain in Figure 6b is given as follows:

$$\begin{aligned} T = &(R_0 + R_1 + R_2)C_2 + (R_0 + R_1 + R_2 + R_3)C_3 \\ &+ (R_0 + R_1 + R_2 + R_3 + R_4)C_4 \end{aligned} \quad (19)$$

Because this approach is very efficient in delay computation, and relative, not absolute, accuracy is sufficient in many cases[6], KCPV uses it as a delay-evaluation method.

The transmission gate (TG) has been the subject of great attention as an effective solution to the density-time challenge, and a cell with a critical delay to be evaluated. The NMOS pass transistor transfers logic '0' well, but degrades logic '1', and the PMOS pass transistor transfers logic '1' well, but logic '0' poorly. Therefore, if both the pass transistors are not treated as one TG, the results turn out to be very pessimistic. Thus, KCPV treats NMOS and PMOS pass transistors that are in parallel and controlled by complementary signals as one CMOS TG. Note that the resistance of the TG in Figure 6, i.e. R_3 and R_4, is given by the equivalent parallel resistance of both the NMOS and PMOS transistors.

IMPLEMENTATION AND EXPERIMENTAL RESULTS

The algorithms described in the previous sections of this paper have been implemented in a program called KCPV. This has about 12 000 lines of source code written in C, and it runs on a Sun-4/370 workstation under Sun operating system 4.0.3. The input format of KCPV is very similar to that of SPICE. KCPV consists of the following two parts: the KAIST MOdel GEnerator (KMOGE) and the critical-path verifier (CPV). KMOGE helps to automate the preparation of the KCPV model file. It builds SPICE input decks for characterization, and analyses the simulation results. KMOGE is designed so that a complete KCPV model file can be built in a reasonable time. The CPV program can find critical paths and estimate delays at the transistor level. The

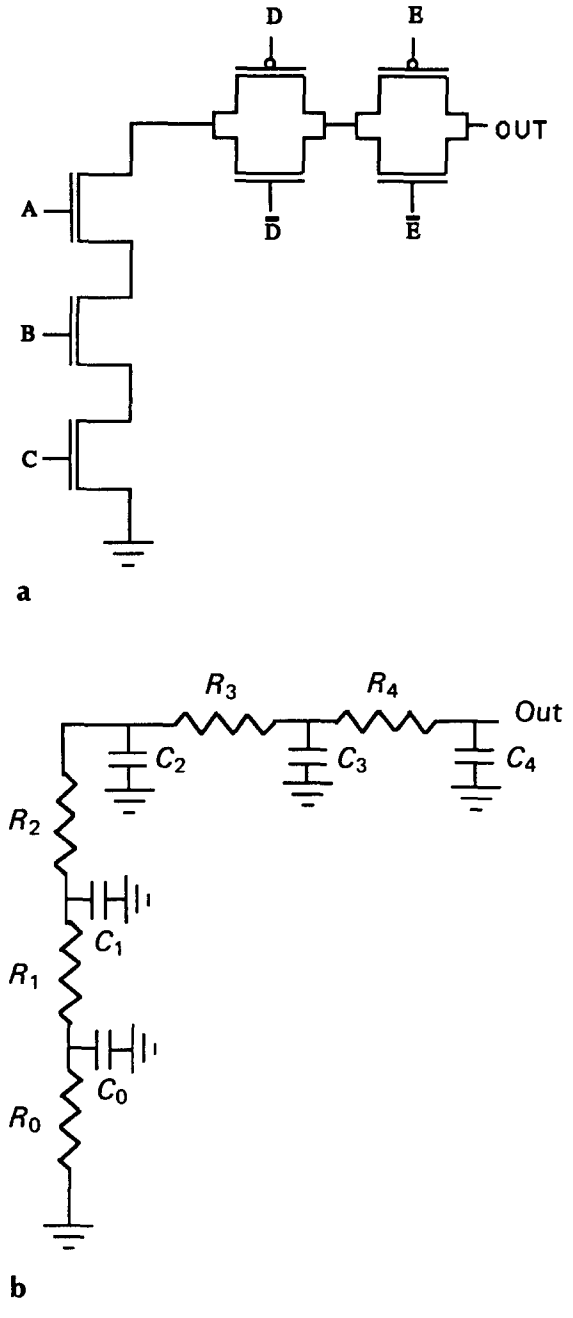

Figure 6. Example of RC delay-time computation; (a) typical network of CMOS stage, (b) linear RC equivalent circuit model for network of (a)

performance of KCPV with several CMOS digital circuits being used is now evaluated.

Table 3 shows a comparison of the CPU times for a modified DFS without a predictor and a modified DFS with a predictor in multiple-path selection for several test cases. Data are obtained by measuring the elapsed CPU time from the time of the triggering of all the inputs to the finishing of the analysis. These examples demonstrate that the modified DFS with a predictor executes from a minimum of two to a maximum of 27 times faster than the conventional search algorithm (modified DFS without a predictor), and that the speedup factor increases with the size of the circuit.

Finally, the proposed KCPV and SPICE are compared for the accuracy of the accumulated delay time by testing them on several test circuits. The analysis results are summarized in Table 4. As an example, Figure 7 shows the critical path of circuit 4, with each block being treated as having the detailed configuration shown in Figure 7b. (Incidentally, Figure 7 is only the combinational part of the original circuit which contains a 1 bit register for each input data bit; this sequential part has to be identified in the program, but omitted in the figure, as it cannot constitute a part of the critical path). The CPU time of KCPV shown in Table 4 is composed of two parts: the unparenthesized number shows the time required to search 20 multiple critical paths and calculate their delay times, and the parenthesized number shows the time required for other jobs, such as input file reading, data-structure setup and connectivity checking. In the cases of circuits 5 and 6, SPICE data are not available, owing to there being no convergence. It can be seen from the table that the accumulated delay error between KCPV and SPICE is less than 11%.

From the experimental results, it has been shown that KCPV provides accurate delay information while requiring short CPU times; this is adequate for practical use.

CONCLUSIONS

Critical-path verification is necessary for proper delay performance to be assured in the actual implementation of the system.

Table 3. CPU-time comparisons of path-search methods for several test circuits (multiple-path count = 20)

Test case	Number of MOSS	Number of nodes	Modified DFS without predictor		Modified DFS with predictor		Speedup factor
			Stage count	CPU time s	Stage count	CPU time s	
Case 1	200	111	2475	1.7	1184	1.1	1.5
Case 2	472	252	8651	8.3	3176	3.0	2.7
Case 3	800	435	35235	25.9	5656	5.7	4.5
Case 4	808	420	309992	311.7	32484	34.5	9.0
Case 5	3712	1987	867075	642.6	34998	39.8	16.1
Case 6	6912	3715	2763139	2539.1	78581	92.7	27.4

Table 4. Comparison of accuracy in accumulated delay-time calculation between KCPV and SPICE

Test circuit	Number of MOSS	CPU time*	Critical-path delay		Deviation
		s	KCPV ns	SPICE ns	%
Circuit 1	154	0.1 (0.5)	7.33	7.72	5.1
Circuit 2	200	0.2 (0.6)	31.17	34.19	8.8
Circuit 3	642	0.4 (2.3)	11.26	11.14	1.1
Circuit 4	808	9.2 (5.0)	33.13	37.14	10.8
Circuit 5	3712	5.7 (20.3)	191.28	NA	NA
Circuit 6	6912	13.6 (52.2)	380.33	NA	NA

[*For 20 multiple critical paths. NA: not available.]

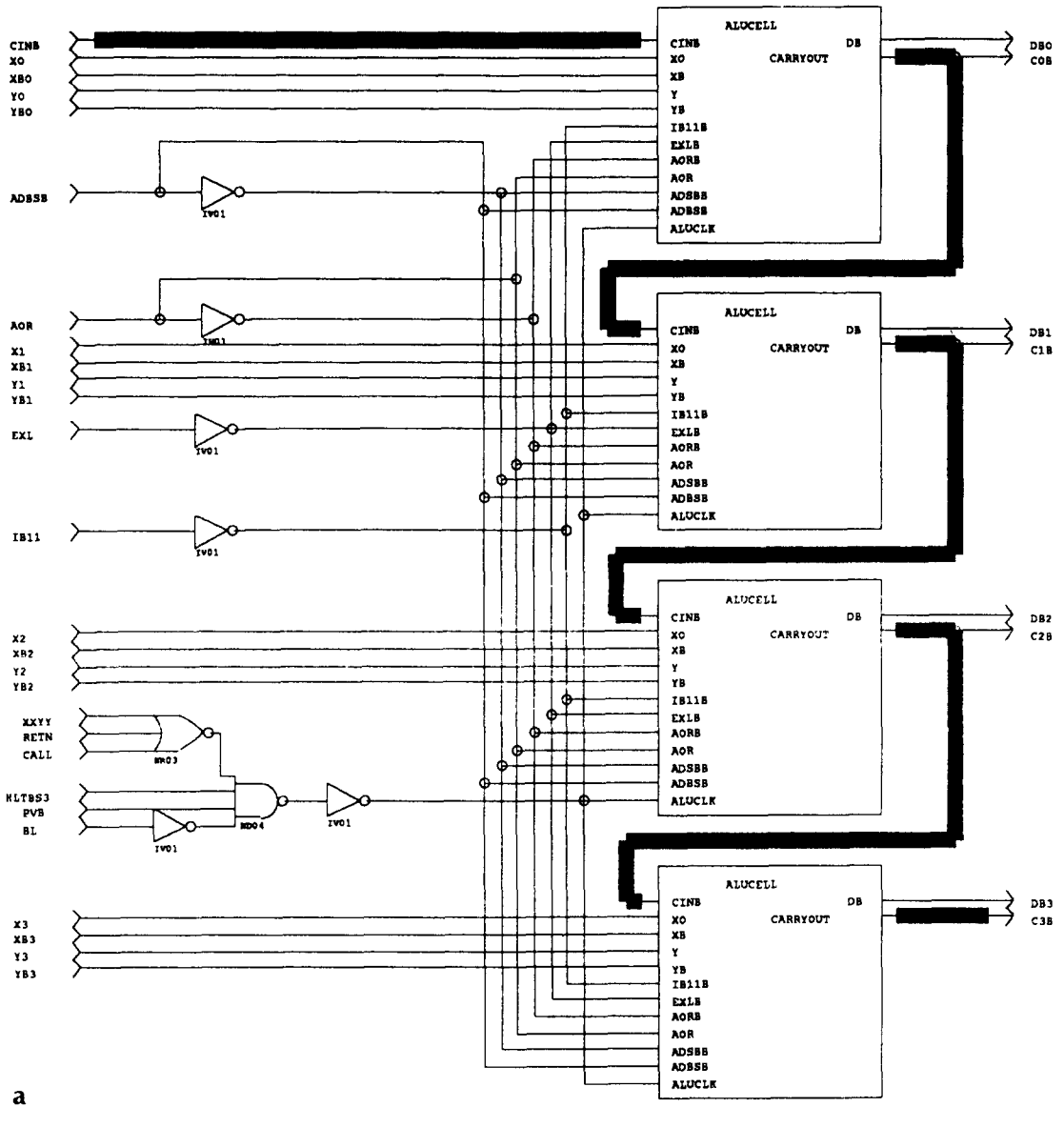

Figure 7. Example of critical path that software finds (shown by heavy line); (a) critical path on block diagram of 4 bit ALU

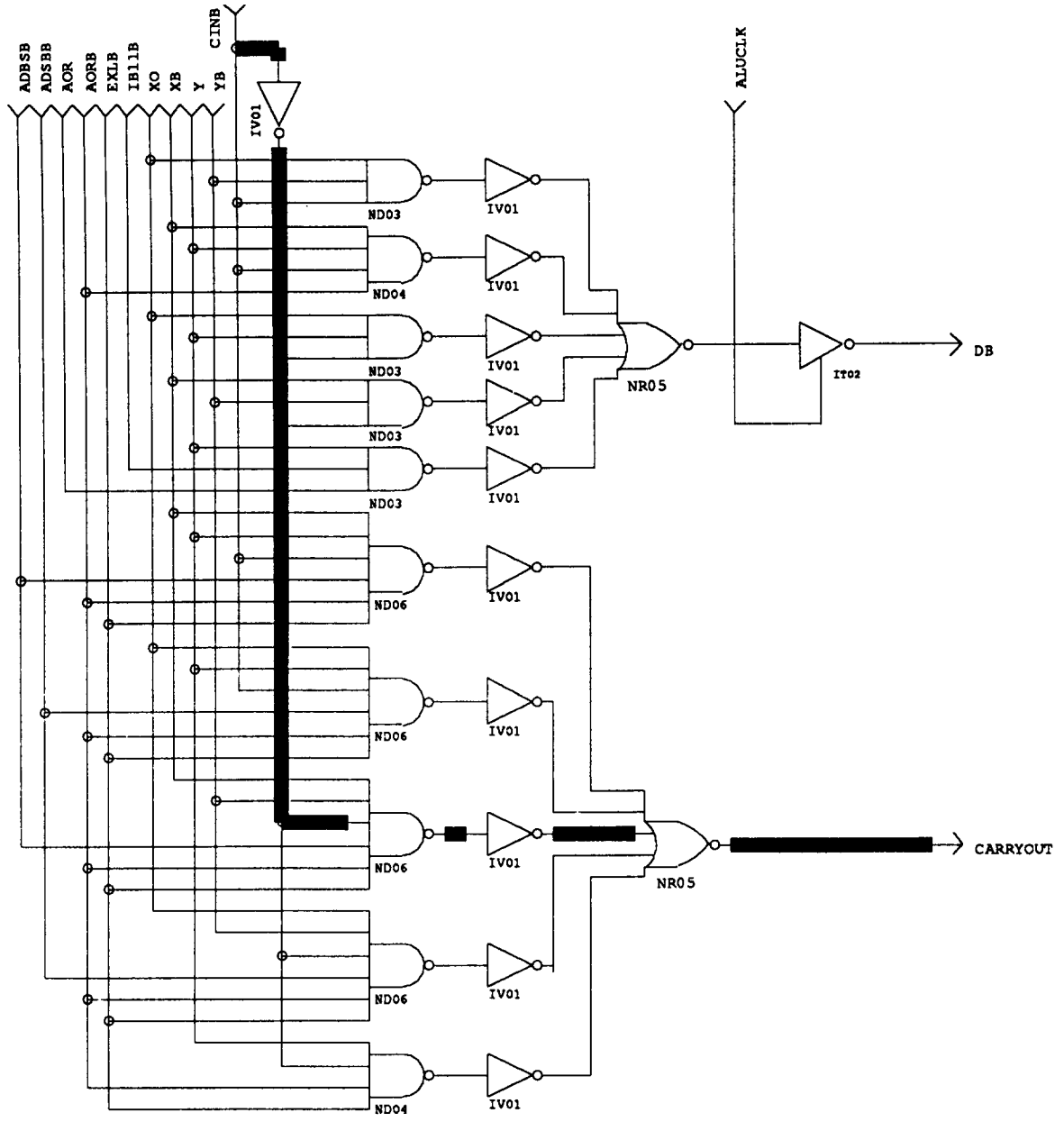

Figure 7. Example of critical path that software finds (shown by heavy line); (b) critical path on circuit of one ALU cell

Owing to the growing size of VLSI circuits, the main concern in critical-path verification has been to develop a fast, accurate model of delay evaluation in a circuit. This paper has proposed a semianalytic model for switch-level CMOS delay times that are formulated using a single parameter, namely, the optimally weighted peak current. The expression for the peak current is derived separately for the cases of fast and slow inputs. The maximum error is about 10%, while the average deviation is about 3%, which is quite acceptable in most CMOS delay calculations. As it uses a series of analytic functions to calculate delay time, the semi-analytic model achieves a tradeoff between speed and accuracy.

KCPV uses the pattern-independent switch-level approach, which results in an efficient and accurate tool for the handling of a wide variety of CMOS circuits. Also, KCPV combines the user's tagging with a set of rules developed to determine the transistor signal-flow direction. An algorithm has been presented that computes multiple critical paths with reduced execution times compared with those of the conventional method.

The experimental results show that KCPV is quite accurate and effective in the verification of CMOS digital designs. As a continuing research effort, the authors are going to develop an effective algorithm and program that optimize the transistor sizes in terms of high operating speed and small chip area.

REFERENCES

1 **Nagel, L W** 'SPICE2: a computer program to simulate semiconductor circuits' *Report ERL-M520* University of California, Berkeley, USA (May 1975)

2 **Chen, C F and Subramaniam, P** 'The second generation MOTIS timing simulator – an efficient and accurate approach for general MOS circuits' *Proc. IEEE Int. Symp. Circuits & Systems* (May 1984) pp 538–542

3 **Bryant, R E** 'A switch-level model and simulator for MOS digital systems' *IEEE Trans. Comput.* Vol C-33 (Feb 1984) pp 160–177

4 **Acuna, E L, Dervenis, J P, Pagones, A J, Yang, F L and Saleh, R A** 'Simulation techniques for mixed analog/digital circuits' *IEEE J. Solid-State Circuits* Vol SC-25 (Apr 1990) pp 353–363

5 **Ousterhout, J K** 'A switch-level timing verifier for digital MOS VLSI' *IEEE Trans. Comput. Aided Des.* Vol CAD-4 (Jul 1985) pp 336–349

6 **Jouppi, N P** 'Timing analysis and performance improvement of MOS VLSI designs' *IEEE Trans. Comput. Aided Des.* Vol CAD-6 (Jul 1987) pp 650–665

7 **Cherry, J J** 'Pearl: a CMOS timing analyzer' *Proc. 25th Design Automation Conf.* (1988) pp 148–153

8 **Hodges, D A and Jackson, H G** *Analysis and Design of Digital Integrated Circuits* McGraw–Hill, USA (1983)

9 **Kang, S M** 'A design of CMOS polycells for LSI circuits' *IEEE Trans. Circuits & Syst.* Vol CAS-28 (Aug 1981) pp 838–843

10 **Kim, K H and Park, S B** 'CMOS delay time model based on weighted peak current' *Electron. Lett.* Vol 24 (Sep 1988) pp 1128–1129

11 **Deschacht, D, Robert, M and Auvergne, D** 'Explicit formulation of delays in CMOS data paths' *IEEE J. Solid-State Circuits* Vol SC-23 (Oct 1988) pp 1257–1264

12 **Rubinstein, J, Penfield, P Jr and Horowitz, M A** 'Signal delay in RC tree networks' *IEEE Trans. Comput. Aided Des.* Vol CAD-2 (Jul 1983) pp 202–211

13 **Ousterhout, J K** 'Switch-level delay models for digital MOS VLSI' *Proc. 21st Design Automation Conf.* (1984) pp 542–548

14 **Benkoski, J and Strojwas, A J** 'A new approach to hierarchical and statistical timing simulations' *IEEE Trans. Comput. Aided Des.* Vol CAD-6 (Nov 1987) pp 1039–1052

15 **Hitchcock, R B Sr** 'Timing verification and the timing analysis program' *Proc. 19th Design Automation Conf.* (Jun 1982) pp 594–604

16 **Ruehli, A E** *Circuit Analysis, Simulation and Design – Vol 2* Elsevier Science, Netherlands (1987)

17 **Annaratone, M** *Digital CMOS Circuit Design* Kluwer, Netherlands (1986)

18 **Fletcher, R and Powell, M J D** 'A rapidly convergent descent method for minimization' *BCS Comput. J.* Vol 6 (1963) pp 163–168

19 **Jouppi, N P** 'Derivation of signal flow direction in MOS VLSI' *IEEE Trans. Comput. Aided Des.* Vol CAD-6 (May 1987) pp 480–490

20 **Tarjan, R** 'Depth-first search and linear graph algorithm' *SIAM J. Comput.* Vol 1 (Jun 1972) pp 146–160