Slope Propagation in Static Timing Analysis

David Blaauw, *Member, IEEE*, Vladimir Zolotov, *Member, IEEE*, and Savithri Sundareswaran, *Member, IEEE*

***Abstract*—Static timing analysis has traditionally used the PERT method for identifying the critical path of a circuit. The authors show in this paper that due to the influence of the transition time of a signal on the subsequent path delay, the traditional timing analysis approach can report an optimistic circuit delay and may identify the wrong critical path. Also, the calculated circuit delay is a discontinuous function with respect to transistor and gate sizes, posing a severe problem for circuit optimization methods. The authors also examine an alternate approach where the propagated signal is constructed by combining the latest arrival time and the slowest transition time from all signals incident on a node. While this approach remedies the problem of discontinuity, it can significantly overestimate the circuit delay and can also identify the wrong critical path. In this paper, they therefore propose a new timing analysis algorithm and prove that it computes the correct and continuous timing graph delay and the proper critical path. The proposed algorithm selectively propagates multiple signals through each timing edge in cases where there exists ambiguity regarding which arriving signal represents the critical path. They show that the algorithm propagates the sufficient and necessary set of signals for computing the delay of a general timing graph. The authors also introduce a new property of digital gates, referred to as the *transition shift property*, and, using this property, show that the number of propagated signals can be significantly reduced for timing graphs of digital circuits. Finally, they discuss the computation of required times and node slacks for the traditional approaches and propose corresponding algorithms for the new approaches. They show that while the traditional approach can incur both a positive or negative error in the computed slack, the proposed algorithms compute a conservative slack for off-critical nodes and the correct and continuous slack for the critical path. The proposed algorithms were implemented in an industrial static timing analysis and optimization tool, and the authors present results for a number of industrial circuits. Their results show that the traditional timing analysis method underestimates the circuit delay by as much as 39%, while the discussed alternate approach can overestimate circuit delay by as much as 17%. The proposed method computes the correct delay, while incurring only a small run time overhead in all cases.**

***Index Terms*—Delay computation, performance verification, static timing analysis.**

I. Introduction

TWO APPROACHES are commonly used to verify the timing of a digital circuit: dynamic simulation and static timing analysis. A disadvantage of dynamic simulation is that it requires the user to generate a set of input vectors which exhaustively exercise all possible paths in a circuit. For large designs, static timing analysis has become the predominant method for timing verification. Static timing analysis also has become the core engine used inside circuit optimization tools such as transistor and gate sizing tools [2], [6], [25] and logic synthesis tools [27]. In static timing analysis, so-called *arrival signals*, which represent the latest time a signal can transition at a node due to a signal change at a circuit input, are propagated forward through a circuit from inputs to outputs. Similarly, so-called *required signals*, which represent the latest time a signal can transition at a node in order to meet the performance constraints, are propagated from circuit outputs to inputs. An arrival signal consists of both the *arrival time*, when the signal reaches a predetermined voltage point, such as the 1/2 of Vdd, and the transition time of the signal, measured for instance from the 10% to the 90% supply voltage crossing times. As signals are propagated across a gate, their arrival times and transition times are updated.

Gate delay is a complex function of the arrival times and slopes of signals at the gate inputs, and over the last decade, extensive research has focused on how to efficiently and accurately calculate propagation delays and slopes for gates in a circuit, addressing issues such as the state dependence of gate delays [10], the impact of slopes on the gate delay [1], [11]–[15] and, more recently, the impact of cross-coupled noise on delay [28]–[34]. Extensive research was also focused on methods to eliminate false paths, which are unrealizable due to logic and timing correlations in a circuit [7], [8], [16]–[20], [24]. However, the essential principle of static timing analysis has remained largely unchanged since it was proposed in 1982 by [4], [5] and is still based on two fundamental assumptions.

1) When calculating the delay of a gate, only one input of the gate is assumed to be switching at a time, ignoring the effects of (near-) simultaneous switching of the gate input signals. This so-called *single-input-switching* assumption greatly simplifies the analysis and enhances its computational efficiency. However, when the single-input-switching assumption is violated in the actual circuit, delay estimates become inaccurate and may underestimate the actual gate delay, therefore yielding an optimistic timing analysis report. In [9] and [21]–[23], this problem was discussed and solutions proposed at the expense of additional run time.

2) Given the single switching assumption, the signal arriving with the latest arrival time at a particular node is assumed to result in the longest path delay and is, therefore, propagated forward, while all other signals with earlier arrival times are pruned. This second assumption is the main topic of this paper, and the traditional algorithm of propagating only the latest arriving signal is referred to as the *latest propagation algorithm (LPA)*.

While the single-switching assumption and the latest propagation algorithm enable efficient timing analysis algorithms,

Manuscript received May 9, 2001; revised November 12, 2001 and April 10, 2002. This paper was recommended by Associate Editor L. Stok.

D. Blaauw is with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48104 USA.

V. Zolotov and S. Sundareswaran are with Motorola Inc., Austin, TX 78709 USA.

Digital Object Identifier 10.1109/TCAD.2002.802274.

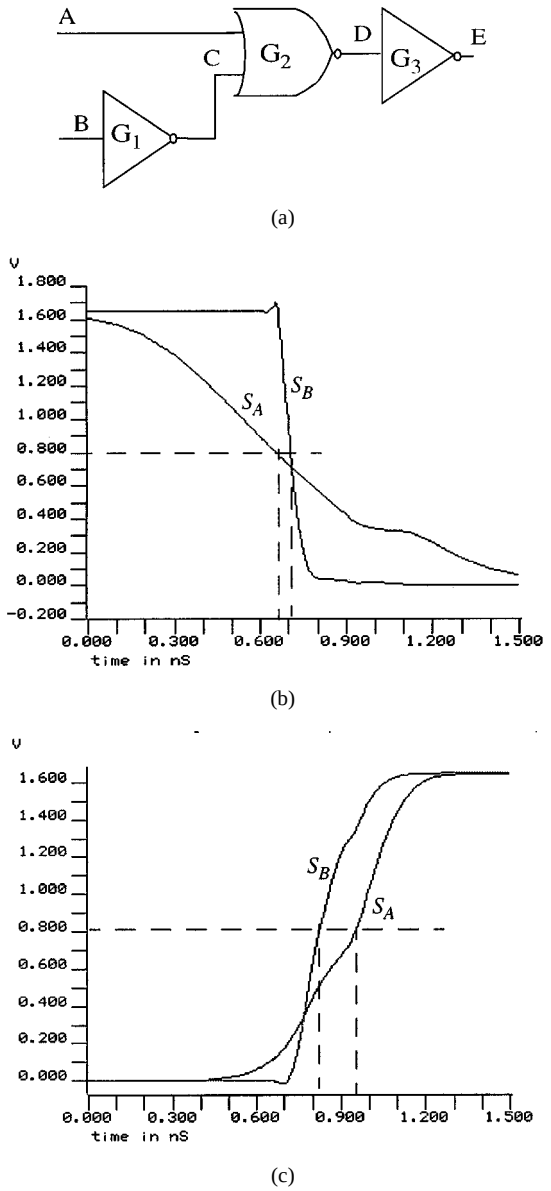


Fig. 1. Error in calculated circuit delay with LPA method. (a) Circuit with two signal propagation paths: A, D, E and B, D, E. (b) Waveforms of two signals at node D. Signal S_A originating from node A and signal S_B originating from node B. (c) Signals S_A and S_B after they propagate through gate G_3.

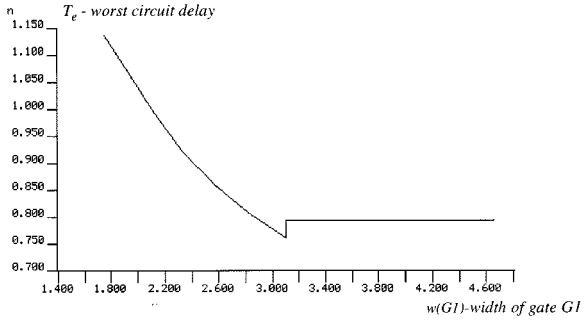


Fig. 2. Discontinuity in delay calculated by LPA method.

they often result in the underestimation of the circuit delay and the identification of the wrong critical path. What is less obvious is that these assumption also lead to undesirable discontinuities of the circuit delay as a function of the underlying gate delays, thereby significantly complicating the task of circuit optimization algorithms. In this paper, we propose a new signal propagation algorithm that preserves the simplicity of the single-input-switching assumption while at the same time improving the accuracy of the analysis and eliminating the discontinuities present in traditional timing analysis algorithms.

The basic problem with LPA is that, out of all signals arriving at a node, it selects one signal for forward propagation based only on the arrival time of the arriving signals and without regard for their transition times. The transition time of a signal at a node, however, has a direct impact on the delay of subsequent gates in its path and therefore affects the overall path delay of the signal. Given two signals, the signal with an earlier arrival time might well have a larger overall path delay if it has a significantly larger signal transition time than that of the signal with the later arrival time. However, LPA will propagate the signal with the later arrival time, and the signal with the longest path delay will remain undetected, resulting in an underestimation of the worst circuit delay. To illustrate this problem, we have shown in Fig. 1(a) a simple two-input circuit with two possible signal paths, one originating from input A (signal S_A) and one originating from input B (signal S_B). Either of the two signals is applied, such that there is only one input switching at any one point in time. Delay computations when simultaneous inputs are switching have been studied elsewhere [21]–[23]. Fig. 1(b) shows the waveforms for signals S_A and S_B at node D where signal S_A originates from node A and signal S_B originates from node B. Fig. 1(c) shows the waveforms of the same two signals at node E. Since the arrival time of signal S_B (0.7 ns) is later than that of signal S_A (0.64 ns), LPA propagates signal S_B through gate $G3$ resulting in a total path delay of 0.82 ns as shown in Fig. 1(c). However, the transition time of arrival signal S_A (1.36 ns) is larger than that of signal S_B (0.1 ns) and would result in a significantly larger delay of gate $G3$ if signal S_A were propagated instead of signal S_B. The total path delay of signal S_A would therefore be 0.95 ns, as shown in Fig. 1(c). For this circuit, traditional timing analysis using LPA reports a worst circuit delay of 0.82 ns, while the actual worst circuit delay is 0.95 ns. Although LPA correctly calculated the path delay of signal S_B, it did not detect that signal S_A resulted in a longer path delay than signal S_B and therefore identified the wrong critical path and underestimated the total circuit delay. This error is independent of the delay model, provided the model accounts for the influence of signal transition time on gate delay.

Besides underestimating the total circuit delay, LPA poses problems to circuit optimization algorithms since it results in discontinuities in the calculated worst circuit delay with respect to transistor and gate sizes. This is illustrated in Fig. 2 where the worst circuit delay computed by LPA is shown as a function of the size of gate $G1$. A sudden change in the calculated circuit delay occurs when the size of $G1$ is increased such that the arrival time of signal S_B at node D becomes earlier than that of signal S_A. At this point, the signal propagated by LPA switches from signal S_B to signal S_A and the slope used to calculate the delay of gate $G3$ changes abruptly. This results in a sudden increase in the delay of $G3$ and hence in the worst circuit

delay. Of course, the actual delay of the circuit is a continuous and smooth function of gate sizes and the observed discontinuity is purely an artifact of LPA. Such discontinuities pose a severe problem for efficient gradient-based optimization methods, which rely on the continuity and smoothness of their objective function [3], [25], [26]. Discontinuities tend to trap such optimization methods far from an optimal circuit solution.

Increasingly, designers are using automated sizing and logic synthesis tools which result in optimized circuits with highly balanced path delays. In such balanced circuits, the signals converging at a particular node are likely to have arrival times that are very close to one another. However, they may have dramatically different slopes, and LPA is therefore more likely to select the wrong signal for propagation and report an optimistic worst circuit delay. One approach to address this problem is to propagate the latest arriving signal, but modify its transition time to be the maximum transition time of all signals arriving at a node. We will refer to this method as the *slowest propagation algorithm (SPA)*. This approach guarantees the continuity of the objective function and has therefore been used in certain optimization approaches [2]. However, this approach can significantly overestimate circuit delay. In the example in Fig. 1, the propagated arrival signal would have a arrival time of 0.7 ns and a transition time of 1.36 ns, resulting in an overestimation of circuit delay and a suboptimal optimization result. Hence, there is a critical need to address the issue of slope propagation in static timing analysis.

In this paper, we propose a new signal propagation method which, given the delay and slope functions for the gates in the circuit, computes the worst timing graph delay correctly. The algorithm uses the propagation of multiple signals in cases where there is ambiguity regarding which signal results in the longest path delay. We refer to the proposed approach as the *multiple propagation algorithm* (MPA) and prove that, given the delay and slope function for all gates in the circuit, it identifies the correct delay of the timing graph and that this calculated delay is continuous with respect to gate/transistor sizes. We show that MPA propagates the necessary and sufficient set of signals for correct calculation of the timing graph delay if no further information about downstream gates is known. Since the proposed MPA approach increases the number of propagated signals, it increases the run time of the analysis. However, for digital circuits, we construct an upper bound for the added delay due to differences in the transition time of signals, based on a newly introduced *transition shift property* of digital gates. Using this proposed property, we can significantly reduce the number of propagated signals for digital circuits. We also discuss in some detail the calculation of required times and node slacks for the traditional LPA and SPA approaches and present the required time and slack computation for MPA. For MPA, the slack computed for the critical path is shown to be correct and continuous, while for the off-critical path nodes, the computed slack is conservative. This is in contrast to LPA which can compute incorrect and discontinuous slack for the critical path and can have either a positive or negative error for the off-critical path nodes. We implemented the proposed MPA approach and show through experimental results that the increase in the run time over LPA is minor in practice. Our experiments also show that LPA can underestimate the worst path delay by as much as 39%, whereas SPA can overestimate the worst circuit delay by as much as 17%. We also demonstrate the occurrence of discontinuities in the worst circuit delay when sizing using LPA and demonstrate how these discontinuities are removed with the proposed algorithm.

The remainder of this paper is organized as follows. In Section II we present a formal formulation of the timing analysis problem. In Section III, we discuss the traditional approaches using LPA and SPA. In Section IV we propose our new MPA, including the signal reduction technique using the proposed transition shift property for digital gates. In Section V, we discuss the propagation of required times for the two traditional approaches and the proposed MPA approach. In Section VI, we present our results and in Section VII our conclusion.

II. Problem Formulation

In this section, we present a formal definition of a timing graph and its properties. For the purpose of discussion, we do not include the elimination of false paths due to logic or timing correlations in a circuit in our formulation. The problem addressed in this paper is orthogonal to the problem of false path elimination and our proposed solution can be applied in conjunction with these methods.

Definition 1: A *timing graph* is defined as a directed graph having exactly one source and one sink node: $G = \{N, E, n_s, n_f\}$, where $N = \{n_1, n_2, \ldots, n_k\}$ is a set of nodes, $E = \{e_1, e_2, \ldots, e_l\}$ is a set of edges, $n_s \in N$ is a source node, and $n_f \in N$ is a sink node. Each edge $e \in E$ is simply an ordered pair $e = (n_i, n_j)$ of nodes.

The nodes in the timing graph correspond to nets in the circuit, and the edges in the graph correspond to the connections from gate inputs to gate outputs. Although circuits in general have multiple inputs and outputs, we can trivially transform them to graphs with a single source and sink by adding a virtual source and virtual sink. We assume without loss of generality that signal arrival times are measured at 50% of the signal level. Also, for convenience of notation, we will refer to a signal transition time as a signal *slope* denoted by s and refer to a slower (faster) slope as a longer (shorter) transition time.

Each edge E is assigned two functions: a delay function $d_e = d_e(s_I)$, which represents the signal propagation delay from a gate's input to its output, and a transition time or slope function $s_e = s_e(s_I)$, which represents the transition time of the signal at the gate output. Both are functions of the gate input slope and have the following property which reflects the fact that for a logic gate, a faster input slope produces a lesser gate delay and faster output slope.

Property 1: If slope $s_a < s_b$ then delay $d_e(s_a) < d_e(s_b)$ and output slope $s_e(s_a) < s_e(s_b)$.

Note that the validity of Property 1 will depend on the selected voltage at which the arrival times are measured. If arrival times are measured at $1/2Vdd$, it is possible that for slow input slopes, the delay of a gate is negative and $d_e(s_a) > d_e(s_b)$. However, it is always possible to select a set of voltage levels for rising and falling transitions, such that Property 1 is valid, for instance V_{tn} for rising transitions and $Vdd - V_{tp}$ for falling transitions.

1. Assign to the source node n_0 the signal $S_0=\{T_0, s_0, P_0\}$ where $T_0=0$, $P_0=(n_0)$
2. For each node n_i in the graph in topological order:

{

2.1. For each incoming edge e_{ki} to node i from node k with signal $S_k=(T_k, s_k, P_k)$
create a new signal $S_{ki}=(T_{ki}, s_{ki}, P_{ki})$ where $T_{ki}=T_k+d_{ki}(s_k)$, $s_{ki}=s_{ki}(s_k)$, $P_{ki}=(P_k, n_i)$

2.2. From all signals S_{ki}
select signal $S_{latest}=(T_{latest}, s_{latest}, P_{latest})$,
where $T_{latest} = max(T_{ki})$ and s_{latest} is the slope of the signal with arrival time T_{latest}

2.3 Assign to node n_i signal S_{latest}.

}

Fig. 3. Traditional latest propagation algorithm.

Below we give a definition of a path in the timing graph G and of its path delay.

Definition 2: A path P of timing graph $G = \{N, E, n_s, n_f\}$ is a sequence of its nodes $P = (n_a, n_b, \ldots, n_z)$ such that each pair of adjacent nodes n_g and n_h has an edge $e_{gh} = (n_g, n_h)$.

A path $P = (n_a, n_b, \ldots, n_z)$ defines a sequence of edges $(e_{ab}, e_{bc}, \ldots, e_{yz})$. Given the slope s_a at the first node n_a of path P, we can determine the signal slopes for all the nodes on the path using the equation $s_j = s_{ij}(s_i)$ recursively, where s_j is the to-be-determined slope at node n_j, s_i is the slope at the predecessor node n_j, and s_{ij} is a slope function of the edge e_{ij}. After the signal slope at each node of a path is determined, the delay of the path is determined using the following definition.

Definition 3: The path delay d_P of path P is defined as $\sum_{e_{ij} \in P} d_{ij}(s_i)$, where $d_{ij}(s_i)$ is a delay of an edge e_{ij} on path P with input slope s_i, and the summation is over all edges belonging to path P.

Among all paths terminating at a node, we define the path with the maximum arrival time as *the critical path* up to that node. The critical path of the sink node n_f of a timing graph is referred to as *the* critical path of the timing graph, and its path delay, $d(G)$, is referred to as the delay of the timing graph. For convenience, we will at times refer to the delay of a timing graph as the *circuit delay*. The main objective of timing analysis is to find the correct critical path and to compute its delay. We will now show that the actual delay of a timing graph is a continuous function with respect to transistor and gate sizes. This property is important for circuit optimization methods, since many of such methods rely on their objective function being continuous.

Theorem 1: If the edge delay and slope functions are continuous with respect to some parameters, then the timing graph delay is also continuous with respect to these parameters.

Proof: The delay of each path in a graph is a finite sum of finite compositions of delay and slope functions of individual edges. Hence, the path delay is a continuous function with respect to the parameters of the slope and delay functions. The total graph delay is the maximum of all path delays in the timing graph and is therefore also continuous, since the maximum operation is a continuous function. □

From Property 1 of edge delay and slope functions, it is clear that the path delay for slower signals is always more than for faster ones along the same path, as stated in the following.

Lemma 1: Given two signals S_a and S_b at node n with slopes s_a and s_b, respectively, and such that $s_a < s_b$, then for any signal path $Q = (n, l, \ldots, z)$ the path delay $d_a(Q)$ of signal S_a is always less then the path delay $d_b(Q)$ of signal S_b.

III. Traditional Propagation Techniques

The most obvious technique for finding the critical path of a timing graph is to simply enumerate all paths from its source to sink, compute their delays, and select the path with the worst delay. However, since the worst case number of paths in a circuit is exponential with circuit size, this approach is infeasible for modern circuits. Below in Sections III-A and B we discuss two common approaches for finding the critical path in a circuit based on the PERT approach. Both approaches uses the propagation of signals from the source node to the sink node, which we define more formally as follows.

Definition 4: A signal S_n at a node n is a triplet $S_n = \{T_A, s, P\}$ where T_A is its arrival time at the node n, s is its slope at the node n, and $P = (n_s, n_a, \ldots, n)$ is the signal propagation path from the source node n_s to the node of interest n.

A. Latest Propagation Approach

The traditional timing analysis algorithm iterates through each node in a timing graph in topological order, selecting the signal with the latest arrival time from among all incident signals for forward propagation. As a signal is propagated forward, its arrival time is increased by gate delay $d_{ij}(s_i)$ and its slope is replaced with $s_{ij}(s_i)$, where s_i is the slope of the selected signal. We refer to this algorithm as the LPA to reflect its selection criteria. Note that in LPA, only one signal is propagated across each edge, and each node in the timing graph is visited exactly once. Although in our notation a signal at a particular node records its entire path to that node, in practice a signal only needs to record its predecessor node. So, traditional timing analysis has a run time complexity that is linear with the number of edges in the timing graph. The latest propagation algorithm is presented below in Fig. 3.

In Section I, we already presented a small example circuit for which LPA identifies wrong critical path. We show in Theorem 2 below that the presence of a signal with a slower slope and an earlier arrival time than the latest arriving signal at a node

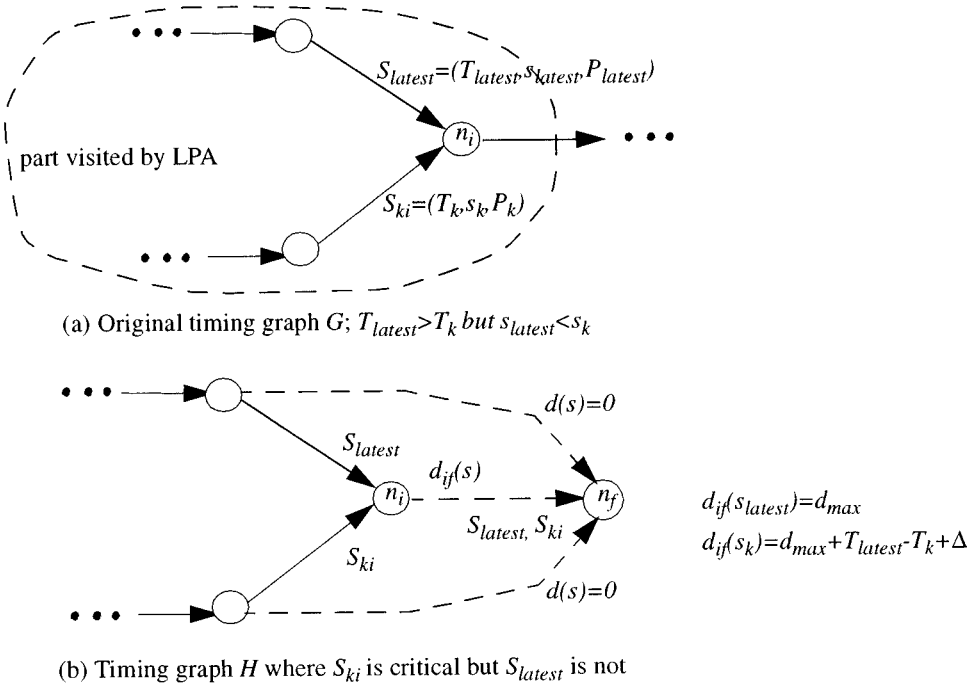


Fig. 4. Construction of a timing graph with a critical path different from the one identified by LPA.

can cause LPA to fail. We prove that the existence of such a signal is a necessary and sufficient condition for the existence of a graph with this signal on which LPA fails. From this we show, in Theorem 3, that the circuit delay calculated by LPA can be discontinuous with respect to the gate sizes.

Theorem 2: If, for some node n_i of timing graph G, LPA selects the signal $S_{\text{latest}} = (T_{\text{latest}}, s_{\text{latest}}, P_{\text{latest}})$ and the slope s_{latest} is faster than the slope s_k of another signal $S_{ki} = (T_k, s_k, P_k)$ propagated to n_i, then we can construct a new graph H, containing all nodes and edges in G that have already been visited by LPA, such that H S_{ki} is critical but S_{latest} is not.

Proof: The construction is demonstrated in Fig. 4. We first construct H such that it contains only the nodes from G that have been visited by LPA. To complete H, we then add a sink node n_f and an edge $e_{kf} = (n_k, n_f)$ for each node n_k of H that does not have an outgoing edge (including node n_i). To all edges e_{kf} we assign delay functions $d_{kf}(s) = 0$, except for $e_{if} = (n_i, n_f)$. We now calculate the maximum path delay of the set of all paths that do not pass through node n_i and denote it as $d_{\max}$. For edge $e_{if} = (n_i, n_f)$ we assign a delay function $d_{if}(s)$ such that $d_{if}(s_{\text{latest}}) = d_{\max}$ and $d_{if}(s_k) = d_{\max} + T_{\text{latest}} - T_k + \Delta$, where $\Delta > 0$. Note that this delay function does not violate Property 1. From this construction it is clear that signal S_{ki} will arrive at the sink node of timing graph H later than signal S_{latest}. □

We now show that the delay calculated by LPA can be a discontinuous function of the gate and transistor sizes.

Theorem 3: It is possible to construct a timing graph G with edge delay functions d_e depending continuously on a certain edge parameter x_e, but such that the timing graph delay $d(x_e)$ computed by LPA is discontinuous with respect to x_e.

Proof: We use the timing graph H constructed in the proof of Theorem 2. Assuming that the edge delay function d_e of subpath P_{latest} depends strongly and monotonically on parameter x_e, we set $x = x_0$ such that the path delays of subpath P_{latest} and P_{ki} are equal. Then, a small variation of x around x_0 will result in a variation of the graph delay by Δ. □

It is clear that LPA always propagates real signals that occur in the actual circuit. Therefore, it never overestimates the arrival times at a circuit node and hence will never overestimate the circuit delay. However, it can underestimate arrival times and the circuit delay as is shown in Theorem 2. For LPA to compute the correct latest arrival times for each circuit node, it is sufficient if at each node, except the sink node, the latest signal that arrives at that node has the slowest slope.

B. Slowest Propagation Approach

The SPA is a variation of LPA where, instead of propagating the signal with the latest arrival time, a new signal is created by combining the latest arrival time and the slowest slope of all the signals arriving at a node. The algorithm is presented in Fig. 5. Contrary to LPA, the slowest propagation algorithm is conservative, meaning it always computes a delay that is greater than the actual delay in the circuit, and the computed delay is a continuous function of transistor and gate sizes in the circuit. We show that SPA is conservative and continuous in Theorems 4 and 5 below. We then show in Lemma 2 that SPA correctly computes the slowest signal slope at a node.

Theorem 4: If for some node n_i of a timing graph SPA assigns signal $(T_{\text{latest}}, s_{\text{slowest}}, P_{\text{latest}})$ for propagation, then its arrival time T_{latest} and slope s_{slowest} are not less than the ar-

1. Assign to the source node n_0 the signal $S_0=\{T_0, s_0, P_0\}$ where $T_0=0$, $P_0=(n_0)$
2. For each node, i, in the graph in topological order:

{

2.1. For each incoming edge e_{ki} to node i from node k with signal $S_k=(T_k, s_k, P_k)$,

create a new signal $S_{ki}=(T_{ki}, s_{ki}, P_{ki})$ where $T_{ki}=T_k+d_{ki}(s_k)$, $s_{ki}=s_{ki}(s_k)$, $P_{ki}=(P_k, n_i)$

2.2. Create a new signal $S_{slowest}=(T_{latest}, s_{slowest}, P_{latest})$ where $T_{latest} = max(T_{ki})$ and $s_{slowest}= max(s_{ki})$

2.3. Assign the computed signal S_{latest} to node n_i.

}

Fig. 5. Slowest signal propagation algorithm.

rival time T_{ki} and slope s_{ki} of any signal S_{ki} that may arrive at this node n_i.

Proof: This property can be proved by induction over timing graph nodes ordered topologically. For the source node it is obvious. Assume that it is true for all nodes n_i where $i < m$. Denote $S_{m,\text{slowest}} = (T_{m,\text{latest}}, s_{m,\text{slowest}}, P_{m,\text{latest}})$ the signal assigned to node m. Consider an arbitrary edge e_{km} from node k to node m and any signal $S_k = (T_k, s_k, P_k)$ that may arrive at node k. Denote $S_{k,\text{latest}} = (T_{k,\text{latest}}, s_{k,\text{latest}}, P_{k,\text{latest}})$ the signal assigned to node k. By the induction assumption $T_{k,\text{latest}} \geq T_k$ and $s_{k,\text{latest}} \geq s_k$. From Property 1, we have inequalities $d_{km}(s_{k,\text{latest}}) \geq d_{km}(s_k)$ and $s_{km}(s_{k,\text{latest}}) \geq s_{km}(s_k)$ for delay and slope functions of the edge e_{km}. Combining these we obtain $T_{k,\text{latest}} + d_{km}(s_{k,\text{latest}}) \geq T_k + d_{km}(s_k)$. From the SPA description it follows that $T_{m,\text{latest}} \geq T_{k,\text{latest}} + d_{km}(s_{k,\text{latest}}) \geq T_k + d_{km}(s_k)$. However, $T_k + d_{km}(s_k)$ and $s_{km}(s_k)$ are the arrival time and slope of the arbitrary signal coming to node m through arbitrary edge e_{ki} which proves the theorem. □

Theorem 5: If the edge delay and slope functions are continuous with respect to some parameters, then the arrival times, delays, and slopes computed by SPA are also continuous with respect to these parameters.

Proof: We can prove it by induction over timing graph nodes ordered topologically in the same way as SPA. For the source node it is obvious. Assume that it is correct for any node with a number less than i, then the arrival time and slope of the signal at node i is computed as $T_i = \max(T_k + d_{ki}(s_k))$, and $s_i = \max(s_{ki}(s_k))$ are continuous functions too. □

Lemma 2: If SPA assigns to node n a signal with slope s, then the signal with the slowest slope that can arrive at this node has exactly this slope s.

For SPA to compute the correct arrival times at each node, it is a sufficient condition that at each timing graph node n, except the sink node, the latest signal among all the signals that arrives at n has the slowest slope. It is interesting to see that this is the same condition necessary for the correctness of LPA, which results from the fact that a timing graph satisfying this condition has no ambiguities in its signal propagation.

From Theorem 4 and Lemma 2 it follows that SPA will make a conservative error at each node where the slope of the latest signal is not the slowest. This error propagates and accumulates along the critical path in the circuit, resulting in an overestimation of the circuit delay. Also, it may result in the identification of the wrong critical path. On the other hand, SPA has the desirable characteristic that the delay is a continuous function of the transistor and gate sizes. This allows circuit optimization problems to be solved with the use of general purpose optimization programs [2]. However, SPA will often compute incorrect dependencies of gate delays on transistor width and therefore can lead to suboptimal solutions. Also, the created signals in SPA do not correspond to real signals in the circuit and can make understanding of the circuit behavior difficult for designers. This, therefore, raises the importance for a timing analysis algorithm that, given delay and slope functions for all gates in the circuit, computes the correct timing graph delay.

IV. Proposed Propagation Algorithm

In order to perform timing analysis that correctly identifies the critical path and delay of a timing graph, we propose a new propagation algorithm. The algorithm propagates multiple signals forward in cases where there is ambiguity regarding which signal results in the longest path delay and is referred to as the MPA. If two arrival signals are incident on a node, such that one of the signals has both an earlier arrival time and a faster slope, the earlier signal is pruned from the analysis. We prove that this algorithm finds the correct critical path and timing graph delay for any timing graph. Also, we show that the algorithm propagates the minimal set of necessary arrival signals. It is not possible to propagate fewer signals without incurring an incorrect critical path and worst timing graph delay for some general timing graph. However, by introducing a new property of signal propagation through digital gates, we can bound the delay added due to the slope difference between two signals. In Section IV-B, we therefore show how this allows us to significantly reduce the number of propagated signals for this pertinent class of timing graphs.

A. Multiple Propagation Algorithm

Instead of propagating a single signal across each edge, MPA propagates sets of signals across the edges of a timing graph. At each node, the union of the propagated signals is taken and is then pruned. A signal is pruned from the set if there exists another signal in the set that has both a not earlier arrival time and

1. Assign to source node n_0 the signal set $C_0=\{S_0\}$, where $S_0=\{T_0, s_0, P_0\}$, $T_0=0$, $P_0=(n_0)$
2. For each node n_i in topological order compute the signal set $C_k=(S_{k1},S_{k2},...)$ as follows:

{

2.1. For each incoming edge $e_{ki}=(n_k,n_i)$ from node k with signal set $C_k=\{S_{k1}, S_{k2},...\}$
create a new signal set $C_{ki}=(S_{ki1},S_{ki2},S_{ki3},...)$, with $S_{kij}=(T_{kij},s_{kij},P_{ki})$,
where $T_{kij}=T_{kj}+d_{ki}(s_{kj})$, $s_{kij}=s_{ki}(s_{kj})$, $P_{ki}=(P_k, n_i)$.

2.2. Assign to node n_i signal set C_i, consisting of the union of signal sets C_{ki}.

2.3. Remove from C_i any signal $S_{ij}=\{T_{ij}, s_{ij}, P_{ij}\}$
if C_i has another signal $S_{ik}=\{T_{ik}, s_{ik}, P_{ik}\}$ such as $T_{ij} \le T_{ik}$ and $s_{ij} \le s_{ik}$

}

Fig. 6. Multiple propagation algorithm.

a not faster slope. The multiple propagation algorithm is shown in Fig. 6. We now prove in Theorems 6 and 7 that the proposed algorithm identifies the correct critical path and in Corollary 1 that the calculated graph delay is a continuous function with respect to transistor and gate sizes.

Theorem 6: For any timing graph G and for any of its nodes n_i, any signal $S_{ij} = \{T_{ij}, s_{ij}, P_{ij}\}$ that is pruned by MPA does not have the latest arrival time at any node n_l following node n_i.

Proof: If, in the proposed algorithm, we prune signal $S_{ij} = \{T_{ij}, s_{ij}, P_{ij}\}$, then at this node n_i there exists another signal $S_{ik} = \{T_{ik}, s_{ik}, P_{ik}\}$ such that $T_{ij} < T_{ik}$ and $s_{ij} < s_{ik}$. Since both S_{ij} and S_{ik} propagate through the same edges after node n_i, and from the property of the slope function, it follows that at any node n_l after node n_i the slope s_{lj} of signal S_{lj} will be less than the slope s_{lk} of S_{lk}. From this and the property of the edge delay function it follows that the edge delays along the path of signal S_{lj} from node n_i to node n_l will always be less than the edge delays along the same path for signal S_{lk}. Since the arrival time is the summation of edge delays from node n_i to node n_l, and since $T_{ij} < T_{ik}$ at node n_i, it follows that S_{ij} will always be earlier than S_{ik}. □

Theorem 7: The multiple propagation algorithm correctly calculates the critical path and delay of a timing graph.

Proof: From Theorem 6, it follows that the proposed algorithm never prunes a signal that could be critical. Hence, all potentially critical signals are propagated to the sink node, where the critical path and graph delay are determined by identifying the latest signal from the set of propagated potentially critical signals. □

Corollary 1: The timing graph delay computed by the proposed algorithm is a continuous function of any parameters of the edge delay or slope function if these functions are, in turn, continuous functions of their chosen parameters.

Proof: It follows from the fact that the algorithm correctly computes the timing graph delay and from Theorem 1 that the calculated timing graph delay is a continuous function of the parameters of the edge delay and slope functions. □

The proposed algorithm propagates the minimum number of necessary signals to compute the correct delay of a timing graph without further knowledge of the subsequent gates in the path of a signal. It is impossible to safely prune additional signals from the propagated set of signals at a node without some further knowledge of the gates that lie topologically after this node. In the next section, we introduce a property of digital gates which, if assumed for this portion of the timing graph, allows the number of propagated signals to be reduced further.

B. Reduction in Propagated Signals

If MPA is used for the analysis of digital circuits, we can utilize some fundamental properties of such circuits to significantly reduce the number of propagated signals. Let us consider two rising signals S_{ia} and S_{ib} that are applied to a digital gate resulting in two falling output transitions S_{ja} and S_{jb}, as shown in Fig. 7(a). We define the following property, which we refer to as the transition shift property.

Property 2: For a digital gate connecting input node n_i with output node n_j, if two input signal waveforms S_{ia} and S_{ib} are related such that at any point along their transition S_{ia} is earlier than S_{ib}, then for all time points along the output waveforms S_{ja} and S_{jb}, waveform S_{ja} will be earlier than S_{jb}.

If signal S_{ib} is later than S_{ia} at all points along its transition, it follows that at every point in time, the voltage of signal waveform S_{ib} will be less than the voltage of waveform S_{ia} (assuming a rising input transition). Digital gates have the property that at any instant in time, a lesser input voltage results in a lesser instantaneous drive current that charges the output load of the gate. Since the output voltage waveform of a gate is simply the integral of this drive current divided by the load capacitance, it is clear that a gate with a lesser driving current at all time points will also have a less complete transition and therefore a later waveform at all points. In other words, a digital gate can only produce an output signal waveform S_{jb} that is earlier than signal S_{ja}, if the input signal S_{ib} is earlier than signal S_{ia} on at least one time instant along its transition. Note that Property 2 may not hold for certain analog circuits. However, Property 2 holds for all standard digital circuits for which static timing analysis is performed, including very high performance and deep-submicron designs as illustrated by the waveforms in Fig. 7(a) from a typical gate of a 0.13-μm 2-Ghz digital processor.

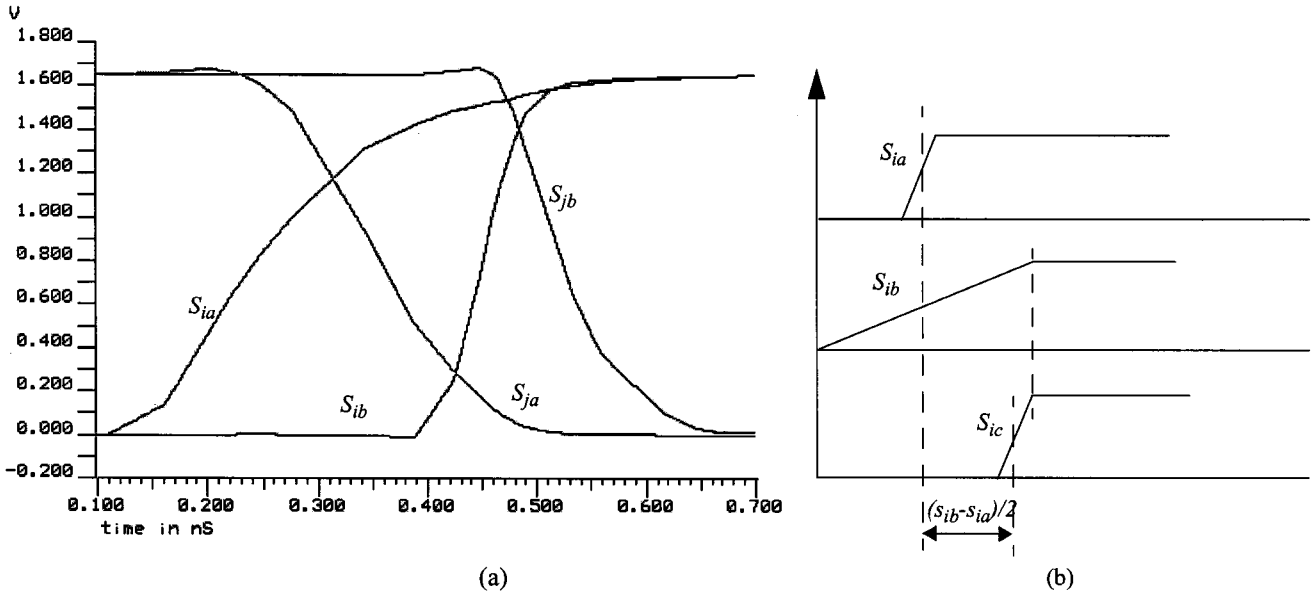


Fig. 7. Bound on added path delay.

We now consider two signals S_{ia} and S_{ib} at a node n_i with the same arrival time but with different slopes, s_{ia} and s_{ib}, signal S_{ib} having the slower slope, as shown in Fig. 7(b). We would like to calculate a bound δ on the difference in arrival times of these two signals at the sink node n_f. To do this, we first replace signal S_{ib} with a signal S_{ic}, such that signal S_{ic} has the same slope as signal S_{ia} and completes its transition at the same point in time as signal S_{ib}. Note that signal S_{ic} is later than signal S_{ib} at all points along its transition. Based on Property 2, signal S_{ic} will be later than signal S_{ib} at all points along its transition at the next node n_j, and by recursion, also at node n_f. Therefore, S_{fc} has a later arrival time at node n_f than signal S_{fb} and therefore the difference in arrival times of S_{fa} and S_{fc} at node n_f is an upper bound on the difference in the arrival times of S_{fa} and S_{fb} at node n_f. Since S_{ia} and S_{ic} have identical slopes, it is clear that the bound δ is exactly the difference in the arrival times of S_{ia} and S_{ic} at node n_i, which is $(s_{ib} - s_{ia})/2$. We define and prove this bound more formally below.

Theorem 8: Given two signals S_a and S_b that propagate along a path P and have at the start point slopes s_a and s_b such that $s_b > s_a$ then, at any node along P, the delay D_b of signal S_b can be bounded by the inequality $D_b \leq D_a + (s_b - s_a)/2$.

Proof: We use auxiliary signal S_c having slope s_a and completing its transition at the same time as signal S_b. Denote the signal S_a, S_b, S_c arrival times to the end of the path as T_a, T_b, T_c and, their delays as D_a, D_b, D_c. Denote as t_0 the arrival time of signals S_a and S_b at the start of the path. Obviously, $D_a = T_a - t_0$ and $D_b = T_b - t_0$. From the construction of signal S_c it follows $D_c = D_a$, $T_c = T_a + (s_b - s_a)/2$. From digital gate property to preserve signal order it follows that $T_b < T_c$. Substituting $T_a + (s_b - s_a)/2$ for T_c and subtracting t_0 from the both sides of the inequality, we obtain the target inequality $D_b \leq D_a + (s_b - s_a)/2$. □

Using the bound from Theorem 8, we propose a *reduced multiple propagation algorithm* (rMPA) where a signal $S_{ia} = \{T_{ia}, s_{ia}, P_{ia}\}$ is pruned from the propagation if another later signal $S_{ib} = \{T_{ib}, s_{ib}, P_{ib}\}$ exists such that $T_{ib} - T_{ia} > (s_{ia} - s_{ib})/2$. In this case, signal S_{ia} is earlier than signal S_{ib} to such an extent that the added delay of S_{ia} in the path from n_i to n_f will not render it critical. Use of this condition in the MPA algorithm limits the propagated signals to a small window of arrival times preceding the latest arrival time and significantly reduces the number of signals propagated through the timing graph. Given the correct delay and slope functions, and given that the gates comply with the transition shift property 2, this condition guarantees the correct calculation of the critical path and graph delay for a circuit. The proof is omitted for brevity. Note that even if transistion shift property does not hold, the obtained timing result will be at least as accurate as with LPA, since the signal with the latest arrival time is always propagated. Also, the proposed reduction in propagated signals is only intended for reducing the run time. For circuits where the transition shift property cannot be guaranteed, the original MPA algorithm can be used, at a modest increase in run time.

We now examine the runtime complexity of the proposed algorithms. In MPA and rMPA, a subset of all signals incident on a node is selected for forward propagation. This operation involves the sorting of all signals according to their arrival time and is thus $O(N \log N)$, where N is the number of signals incident on the node. The sum of all of signals propagated across an edge is in the worst case equal to the number of paths in a circuit. Therefore, the overall complexity of the proposed algorithm is exponential in the worst case, since the number of paths in a circuit is exponential with the size of the graph. This, however, would require that the signal ordering is identical in terms of decreasing arrival time and increasing signal slope, thereby not allowing any pruning. In practice, this is unlikely, and we find that for industrial circuits the number of signals that are propagated in the proposed algorithm is increased only slightly when compared with LPA or SPA.

V. Required Time Propagation and Timing Slack Computation

Up to this point we have only concerned ourselves with the correct identification of the critical path and the circuit delay. However, for many circuit optimization applications, it is necessary to compute the so-called *slack* and *required times* T_R of nodes in the timing graph. For the sink node n_f of the timing graph, the required time $T_{R,\,\mathrm{Sink}}$ is the latest time that a signal can arrive such that the circuit will operate correctly. Required times are defined for a signal at an internal node as the latest time when the signal may arrive at that node in order to arrive at the sink node no later than at the required time. Given the signal required time at a node, the slack of the signal is defined as the difference between the required time and the arrival time of the signal $Sl = T_R - T_A$. The slack of a node is the minimum slack of all the signals at the node. The slack of a node is positive if all signals that arrive at the node have an arrival time that is earlier than their required times, otherwise it is negative.

Required times for internal nodes of a timing graph can be uniquely and easily defined if the edge delays do not depend on signal slopes. In this case, the algorithm is similar to computing arrival times. We propagate the required times backward from the sink node to the source node, decreasing the required times by edge delays as they are propagated. At each node, we can then compute the required time as the minimum of all the required times incident on the node through its fanout edges. Given the node required time, the node slack is simply computed as the difference between the node required time and its latest arrival time.

For realistic timing graphs, the algorithm becomes more complicated since the edge delays and gate output slopes depend on the input signal slopes as expressed, for instance, in Property 1. In this case, the edge delays will vary depending on which signal is propagated through the edge. Therefore, the backward propagation of required times is a function of the signal slopes computed in the forward propagation and the backward and forward propagation are interdependent. A required time at a node n is therefore only properly defined for a particular arrival signal at node n. The signal slack is defined as the difference between the signal arrival time and its corresponding required time and the node slack is the minimum slack of all signal slacks at the node. The node slack represents the maximum amount of time by which any signal at the node can be delayed such that it will still arrive at the sink node before the sink required time $T_{R,\,\mathrm{Sink}}$.

The difficulty in computing signal slacks results from the fact that arrival signals are pruned during forward propagation. Therefore, during backward propagation, a required signal will not exist for a pruned arrival signal. In this case, a new required signal must be constructed based on the available required signals, introducing inevitable error in the computation of slacks. In this section, we will present slack computation algorithms for the presented propagation algorithms and discuss the accuracy of each.

For future consideration, we first define the required time in a formal and algorithm independent way as follows.

Definition 5: Given signal $S_{ik} = (T_{ik}, s_{ik}, P_{ik})$ at node n_i, let $T_{ik,\,\mathrm{SinkLatest}}$ be the latest time when the signal S_{ik} can arrive at the sink node n_f. Let $T_{R,\,\mathrm{Sink}}$ be the required time at the sink node. We define the required time for the signal S_{ik} at node n_i as $T_{R,\,\mathrm{Sink}} - T_{ik,\,\mathrm{SinkLatest}} + T_{ik}$.

For a signal that propagates from the source node to the sink node of a timing graph, the *signal slack* is the same at all the nodes along the propagation path. Of course, this does not mean that the *node* slack is the same for all nodes along the path, because the node slack is the minimum signal slack over all signals that arrive at the node. However, all critical path nodes have the same node slack, which is the least (most negative) node slack value in the circuit. Note that the critical path slack is equal to the sink required time minus the circuit delay. Also, it can be shown that it is impossible to compute the correct slack for every node in a general timing graph where edge delay and gate output slopes are a function of the input slopes, without propagating all arrival signals and their corresponding required times. Since all practical timing analysis algorithms (including the proposed MPA and rMPA approaches) prune signals at nodes in the timing graph, they will incur errors in the computed slacks for some nodes and signals in the timing graph.

In the following four sections, we will discuss the required time computation in more detail for the traditional LPA and SPA approaches as well for the proposed MPA and rMPA approaches. We will show that LPA and SPA can compute incorrect slacks for any node in a general timing graph, including nodes along the critical path. LPA can compute slacks with either a positive or negative error, while SPA will always compute slacks with a negative error, meaning that it will report a smaller, more negative slack than the actual slack. We will then show that, given delay and slope functions for all gates, MPA and rMPA compute slack for the critical path of a timing graph correctly and that this slack is a continuous function of the gate and transistor sizes in the circuit, which is an important property for optimization algorithms. For nodes not on the critical path, MPA and rMPA may compute slacks with a negative error, meaning that the approach is conservative. Finally, we will show that MPA and rMPA adhere to the important property that the computed slack of a noncritical path node is never smaller than the slack of the critical path. In other words, the error in the computed slack is never so great that the slack of a node not on the critical path is less than the slack of a node on the critical path, which of course is an important criteria for effective analysis and optimization of a circuit.

A. Required Time Propagation for Latest Propagation Algorithm

For LPA, the required time computation exploits the fact that only a single signal is propagated across each edge and therefore each edge has a unique edge delay. Hence, the required time computation for LPA is similar to the required time computation for timing graphs with fixed edge delays and is shown in Fig. 8. From the fact that LPA underestimates the circuit delay, it follows that LPA generates a worst slack larger than or equal to the actual worst slack in the timing graph, i.e., it is optimistic.

We will now show that LPA can compute a slack for some nodes with either a positive or negative error using the example shown in Fig. 9. Let us assume that signals $w_1 = S(T_1, s_1, P_1)$

1. Assign to the sink node the required time T_{RN} equal to specified required time.
2.For each node, *i*, in the graph in reverse topological order:
{
2.1 For each outgoing edge e_{ik} from node *i* to node *k*
compute $T_{ik}=T_k-d_{ik}$
2.2 From all computed T_{ik} select the minimal value $T_i=min(T_{ik})$
2.3 Assign the computed value T_i as a required time to node n_i
}

Fig. 8. Required time propagation algorithm for LPA.

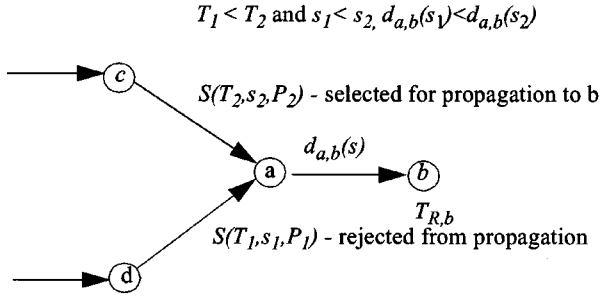


Fig. 9. Example of LPA underestimating node timing slack.

and $w_2 = S(T_2, s_2, P_2)$ arrive at node a from nodes d and c at time T_1 and T_2 such that $T_1 < T_2$ and $s_1 < s_2$. LPA at node a propagates to node b signal w_2. The delays of edge (a, b) for these signals are such that $d_{a,b}(s_1) < d_{a,b}(s_2)$. The required time propagation starting from node b uses the results of the forward propagation and therefore the required time $T_{R,a}$ for node a is computed using delay $d_{a,b}(s_2)$ of the edge (a, b) and is used for computing the required time for node d as $T_{R,d,computed} = T_{R,b} - d_{a,b}(s_2) - d_{d,a}$. Therefore, it underestimates the required time of node d because in reality the required time of node d is determined by propagation of signal w_1 as $T_{R,d,real} = T_{R,b} - d_{a,b}(s_1) - d_{d,a}$, and $d_{a,b}(s_1) < d_{a,b}(s_1)$. Conversely, if we assume $s_1 > s_2$, the delay of edge (a, b) is such that $d_{a,b}(s_1) > d_{a,b}(s_2)$ and LPA will overestimate the required time of node d. Hence, we can conclude that LPA can make an error in the required time and slack computation of any node that is not on the critical path, with either a positive or negative sign, while for nodes on the critical path, LPA will always have a positive error.

B. Required Time Propagation for Slowest Propagation Algorithm

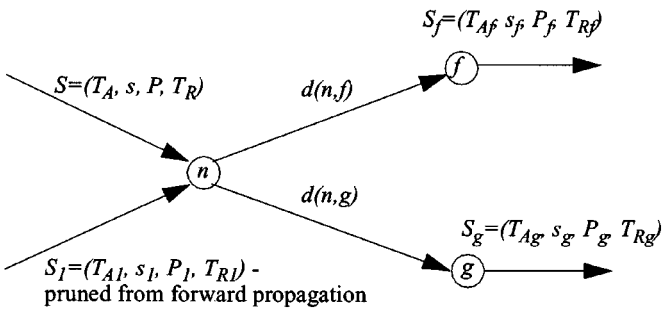
In SPA, a single edge delay is again computed for each edge in the timing graph, and the required time calculation is the same as that shown in Fig. 8 for LPA. After SPA assigns delays to edges during forward signal propagation, the required time propagation is uniquely determined by these delays. The SPA approach differs from the LPA approach in that it uses the maximum possible slope at a node for computing the edge delays. Therefore, LPA will compute edge delays that are greater than or equal to the actual delays of an edge and it will compute a required time and slack that are less than or equal to the required time and slack of the actual signals, i.e., it is pessimistic.

C. Required Time Propagation for Multiple Propagation Algorithm

In order to present the backward propagation algorithm for MPA, we extend the definition of a signal by including a required time as follows. Signal S_n at node n is a quadruplet (T_A, s, P, T_R), where T_A and s are the signal arrival time and slope at node n, P is the path along which the signal propagates from the source node to node n, and T_R is signal required time at node n. The forward propagation of MPA remains unchanged and leaves the required time of the signals undefined. During the forward propagation, sets of signals are assigned to each node, and the edge delay corresponding to each signal that passes through an edge is defined.

The backward propagation visits each node in reverse topological order and defines the required time for each signal. For a signal s_n at node n, the corresponding signals s_f at the fanout nodes n_f of n that resulted from the propagation of s_n during the forward propagation are identified. The required time of a signal s_f is then propagated backward through the edge (n, n_f) and is decreased by the edge delay of edge (n, n_f) corresponding to signal s_n. Among all backward-propagated required times, the earliest required time is assigned to signal s_n at node n. If signal s_n at node n was pruned during the forward propagation of MPA, then no corresponding signal will be found at the fanout nodes n_f. Hence, a required time for such a pruned signal s_p is chosen from the required times of the nonpruned signals at node n. The required time of the nonpruned signal that has the least slower slope than the slope of the pruned signal s_p is used. This operation is illustrated in Fig. 10. It is easy to see from the MPA forward-propagation algorithm in Fig. 6 that such a signal will always be available, since its existence is a required condition for pruning signal s_p. After the signal required time is computed for all signals at node n, their slacks can be computed and the node slack for node n is set to the minimum slack of all its signals. The algorithm for backward propagation of required times under MPA is shown in more detail in Fig. 11.

Since the required time for the pruned signals is selected from a propagated signal with a slower slope, the computed required time is always less than the actual required time. In other words, MPA will always report a pessimistic required time. However, by selecting the signal with the *least* slower slope, we minimize the error in the computed slack as much as possible. For the *critical* signal, propagated along the critical path, MPA always computes the correct slack. Hence, the slack of all critical nodes will be correct. From this, it also follows that the slack computed



- required time of non pruned signal S: $T_R=min[(T_{Rf}-d(n,f), T_{Rg}-d(n,g)]$ - minimum of all required times

- required time of pruned signal S_1: $T_{R1}=T_R$ is recovered from required time of signal S

Fig. 10. Required time propagation for MPA.

1.Update the required time of all the signals at sink node with the sink node required time T_{RSink}.
2.For each node n_i in the timing graph in reversed topological order:
{
2.1. Denote $N_{Succ}=(n_{i,1}, n_{i,2},..., n_{iNi})$ a set of the fanout nodes of node n_i.
2.2. Consider each signal $S_{ik}=(T_{Aik}, s_{ik}, P_{ik}, T_{Rik})$ at node n_i doing the following:
{
2.2.1. For each fanout node n_j that has the signal $S_{jl}=(T_{Ajl}, s_{jl}, P_{jl}, T_{Ril})$
such that path P_{jl} completely includes path P_{ik}
compute value $T_{ijk}=T_{Ril}-d_{ij}(s_{ik})$
where $d_{ij}(s_{ik})$ is delay function of edge e_{ij} between nodes n_i and n_j.
2.2.2. If the set of computed values T_{ijk} is not empty
update the required time of signal $S_{ik}=(T_{Aik}, s_{ik}, P_{ik}, T_{Rik})$ with value $T_{Rik}=min(T_{ijk})$
}
2.3. For each signal $S_{ik}=(T_{Aik}, s_{ik}, P_{ik}, T_{Rik})$ at node n_i that has undefined required time:
{
2.3.1. Among all the signals at node n_i with defined values of required time
find signal $S_{il}=(T_{Ail}, s_{il}, P_{il}, T_{Ril})$ with the least slope s_{il} such as $s_{il} > s_{ik}$
2.3.2. Update the required time of signal $S_{ik}=(T_{Aik}, s_{ik}, P_{ik}, T_{Rik})$
with the required time of the found signal T_{Ril}
}
2.4. Assign to the node n_i the required time equal to the minimum of the required times of all its signals.
}

Fig. 11. MPA required time propagation algorithm.

by MPA for the critical nodes is a continuous function of the transistor and gate sizes.

We show below in Theorem 9 that the computed slack for noncritical signals is never less than the slack of the critical signal. This is an important and necessary property for the required time computation to be useful. If the slack of noncritical nodes were computed to be less than that of the critical path nodes, the optimization of the circuit would certainly fail.

Theorem 9: The slack computed by MPA for noncritical signals is never less than the slack of the critical signal.

Proof: We prove this by contradiction. The theorem is true for the sink node. Assume that it is also true for all signals at any node with index more than i and is wrong for node n_i. Clearly, this would be impossible since the required time computed by steps 2.2, 2.2.1, and 2.2.2 in Fig. 11 decreases the successor required time by the same delay as the arrival time was increased during forward signal propagation. In the forward-propagation steps 2.3, 2.3.2, and 2.3.2 in Fig. 11, the required times for pruned signals are computed. Consider an arbitrary pruned signal $S_{pr} = (T_{pr}, s_{pr}, P_{pr}, T_{R,pr})$. There exists a nonpruned signal $S_{np} = (T_{np}, s_{np}, P_{np}, T_{R,np})$ such that $s_{pr} < s_{np}$ and $T_{pr} < T_{np}$. From such signals, the algorithm selects a signal $S_{np1} = (T_{np1}, s_{np1}, P_{np1}, T_{R,np1})$ such as $s_{np1} > s_{pr}$, $S_{np1} \leq S_{np}$ and sets $T_{R,pr} = T_{R,np1}$. Therefore, $T_{R,pr} = T_{R,np1} \geq T_{R,np}$ because the delay of signal S_{np1} from node n_i to the sink is less than the same delay of the

signal S_{np}. Subtracting the inequalities and $T_{pr} < T_{np}$ we obtain $Sl_{pr} > Sl_{np}$ and therefore the slack computed for a pruned signal is never less than the slack of a nonpruned signal at the same node. □

In summary, MPA computes slack values that are exact and continuous for the critical path nodes. For other noncritical nodes, MPA will compute a conservative slack that is less than the actual slack. However, this computed slack will never be less than the slack of the critical path. The required time of pruned signals is estimated from the required time of nonpruned signals. This estimate may be a discontinuous function with respect to transistor and gate sizes since they can cause an abrupt change in the set of nonpruned signals. Therefore, the slacks computed by MPA for noncritical nodes can be a discontinuous function too. It is clear that the only way to compute the exact required times for all nodes involves computing exact delays of each signal from each internal node to the sink node which requires the propagation of all the signals. This would lead to an exponential complexity of the slack computation algorithm. Barring this possibility, the proposed MPA approach satisfies the best attainable characteristics of slack computation while maintaining a good run time performance.

D. Required Time Propagation for Reduced Multiple Propagation Algorithm

In the reduced multiple propagation algorithm, additional signals are pruned during the forward propagation based on the transition shift property, Property 2. The backward-propagation algorithm for rMPA is therefore the same as that for MPA, except that the required time for these additional pruned signals needs to be computed. Computation of these required times is shown in steps 2.4, 2.4.1, and 2.4.2 in Fig. 12. Pruning of these additional signals does not require the presence of a propagated signal with a slower slope but only the presence of a propagated signal that has a sufficiently greater arrival time. Therefore, a signal with a slower slope may not be found for these additional pruned signals in step 2.3.1. In step 2.4.1, we therefore select the signal with the least faster slope than the slope of the pruned signal and utilize its required time for the pruned signal. Note that in step 2.4.2, the required time of the pruned signal is computed by subtracting from the required time of the signal with the least faster slope the bound $(s_{ik} - s_{il})/2$. This ensures that the computed required time and signal slack of the pruned signal are conservative. The delay bound $(s_{ik} - s_{il})/2$ is guaranteed to overestimate the added delay of the downstream gates due to the slower slope of the pruned signal. Therefore, the required time and slack of the pruned signal will be conservative.

We now show that the computed slack in rMPA is never worse than the slack of the critical signal.

Theorem 10: rMPA computes a required time for a signal such that the signal slack is never less than the slack of the critical signal.

Proof: We need to prove this only for the additional steps 2.4, 2.4.1, 2.4.2 in Fig. 12. Assume a signal S with slope s, delay D from the node of interest to the sink node, and required time T_R. Assume also that signal S is pruned during the forward propagation according to digital gate Property 2. Let S_1 be the signal used for estimating the required time of the signal S with required time T_{R1} and delay D_1 from the node of interest to the sink node. Let S_p be the signal that pruned signal S, using the digital gate property, with slope s_p and arrival time T_p. $T_{R,\mathrm{Sink}}$ denotes the required time at the sink node.

The estimated slack computed by the algorithm for signal S is equal to $Sl_{est} = T_{R1} - ((s - s_1)/2) - T$. The slack of the signal S_p is equal to $Sl_p = T_{Rp} - T_p$. As the signal S is pruned by signal S_p, we have inequalities $s > s_p$ and $T \leq T_p - (s - s_p)/2$. From the definition of the algorithm, it follows that $s_1 \geq s_p$. Then by combining these inequalities with the above expression of the slack estimation, we can obtain $Sl_{est} \geq T_{R1} - ((s - s_p)/2) - T_p + (s - s_p)/2 = T_{R1} - T_p$. It is obvious that $D_1 \geq D_p$ because $s_1 \geq s_p$. Therefore, $T_{R1} = T_{R\,\mathrm{Sink}} - D_1 \leq T_{R\,\mathrm{Sink}} - D_p = T_{Rp}$. Combining this inequality with the inequality of the slack estimation, we obtain $Sl_{est} \geq T_{Rp} - T_p = Sl_p$. This means that the estimated slack at a node is greater than or equal to the slack of the critical path. □

VI. Results

The proposed algorithms were implemented in an industrial transistor-level static timing analysis and optimization tool. Both the MPA algorithm presented in Fig. 6 and the rMPA algorithm which reduces the number of propagated arrival signals for digital circuits were implemented. The algorithms were tested on a number of industrial designs ranging in size from 780 to 12 500 transistors. These included circuit blocks from high-performance microprocessors and DSP chips. In Table I, we show the circuit delay calculated using the traditional LPA method, the conservative SPA method, and the proposed MPA method from this paper. The table demonstrates that the LPA method underestimates the circuit delay by as much as 39% for the decoder circuit and by 21% on average. On the other hand, the SPA method overestimates the circuit delay by as much as 17% for the decoder circuit and by 8% on average over all test cases. In all cases, the circuit delay computed with the MPA approach lies between the results from the LPA and SPA approach, as expected.

Table I shows the difference between the circuit delays computed by the three algorithms using identical delay and slope functions. However, these underlying delay and slope functions have their own inherent inaccuracies due to the limitations of the underlying transistor-level delay models. Hence, the computed circuit delay by all three methods, including the proposed MPA method, will in practice differ from SPICE-base simulation as well as from silicon measurements. To improve the estimate of the computed circuit delay, it is therefore common for designers to run SPICE simulations on the critical paths identified by the timing-analysis algorithm. We preformed such an analysis and show the results in Table II. In column 3 through 5 we show the delay estimate obtained through SPICE simulation of the critical path identified by the different timing analysis algorithms and report the difference between this SPICE delay and the delay computed by timing analysis algorithms using more approximate transistor-level delay models (% *error*). The error in the transistor-level delay models can results in either an under or overestimate of the compute circuit delay, as compared with the

1.Update the required time of all the signals at sink node with the sink node required time T_{RSink}.

2.For each node n_i i n the timing graph in reverse topological order:

{

2 .1. Denote $N_{Succ}=(n_{i,1}, n_{i,2},..., n_{iN})$ a set of the fanout nodes of node n_i.

2.2. Consider each signal $S_{ik}=(T_{Aik}, s_{ik}, P_{ik}, T_{Rik})$ at node n_i d oing the following:

{

2 .2.1. For each fanout node n_j that has the signal $S_{jl}=(T_{Ajl}, s_{jl}, P_{jl}, T_{Ril})$
such that path P_{jl} completely includes path P_{ik}
compute value $T_{ijk}=T_{Ril}-d_{ij}(s_{ik})$
where $d_{ij}(s_{ik})$ is the delay function of edge e_{ij} between nodes n_i and n_j.

2.2.2. If the set of computed values T_{ijk} is not empty
update the required time of signal $S_{ik}=(T_{Aik}, s_{ik}, P_{ik}, T_{Rik})$ with value $T_{Rik}=min(T_{ijk})$

}

2.3. For each signal $S_{ik}=(T_{Aik}, s_{ik}, P_{ik}, T_{Rik})$ at node n_i t hat has undefined required time:

{

2 .3.1. Among all the signals at node n_i with values of required time computed at steps 2.2, 2.2.1,
find signal $S_{il}=(T_{Ail}, s_{il}, P_{il}, T_{Ril})$ with the least slope s_{il} such as $s_{il} > s_{ik}$

2.3.2. Update the required time of signal $S_{ik}=(T_{Aik}, s_{ik}, P_{ik}, T_{Rik})$
with the required time of the found signal T_{Ril}

}

2.4. For each signal $S_{ik}=(T_{Aik}, s_{ik}, P_{ik}, T_{Rik})$ at node n_i t hat has undefined of required time:

{

2.4.1. Among all the signals at node n_i with values of required time computed at steps 2.2, 2.2.1,
find signal $S_{il}=(T_{Ail}, s_{il}, P_{il}, T_{Ril})$ with the longest s_{il} such as $s_{il} < s_{ik}$

2.4.2. Update the required time of signal $S_{ik}=(T_{Aik}, s_{ik}, P_{ik}, T_{Rik})$ with the value $T_{Ril} - (s_{ik}-s_{il})/2$

}

2.5. Assign to the node n_i the required time equal to the minimum of the required times of all the signals.

}

Fig. 12. rMPA required time propagation algorithm.

TABLE I
EFFECT OF SLOPE PROPAGATION ON CIRCUIT DELAY

Circuit	# transistors	Total Circuit Delay in nS				
		LPA	SPA	MPA	% Diff with LPA	% Diff with SPA
mux	784	0.88	1.27	1.16	24%	9%
adder	1,074	0.55	0.93	0.83	34%	12%
decoder	1,490	2.11	4.05	3.47	39%	17%
contr3	3,190	2.63	3.61	3.22	18%	12%
reg	4,902	3.83	4.77	4.66	17%	2%
contr2	11,112	8.49	9.48	9.26	8%	2%
contr1	12,519	2.07	2.29	2.26	8%	1%

SPICE based simulation. In columns 6 and 7, the difference between the SPICE delay of critical paths for LPA and SPA are compared with that of MPA. In all cases, MPA identifies a critical path that has a SPICE delay that is greater or equal to the SPICE delay of the critical paths identified by LPA and SPA. Despite the inaccuracies in the underlying delay model, MPA is therefore able to better identify the critical path than the LPA and SPA algorithms.

Note that even though the circuit delay computed by SPA in Table I is consistently greater than that computed by MPA, the SPICE delay of the critical path identified by SPA is consistently less than or equal to that of MPA. This is due to the fact

TABLE II
SPICE DELAY OF CRITICAL PATHS IDENTIFIED BY LPA, SPA, AND MPA ALGORITHMS

Circuit	# transistors	Delay of critical paths simulated with SPICE (nS)				
		LPA (% error)	SPA (% error)	MPA (% error)	% Diff with LPA	% Diff with SPA
mux	784	1.03 (-14.6%)	1.03 (23.3%)	1.35 (-14.1%)	23.7%	23.7%
adder	1,074	0.604 (-8.9%)	0.612 (52.0%)	0.649 (27.9%)	6.9%	5.7%
decoder	1,490	2.27 (-7.0%)	3.29 (23.1%)	3.41 (1.8%)	33.4%	3.5%
contr3	3,190	2.40 (9.6%)	2.71 (33.2%)	2.77 (16.2%)	13.4%	2.2%
reg	4,902	4.47 (-14.3%)	5.36 (-11.0%)	6.16 (-24.4%)	27.4%	13.0%
contr2	11,112	8.67 (-2.3%)	8.67 (9.3%)	9.56 (-3.1%)	9.3%	9.3%
contr1	12,519	2.50 (-17.2%)	2.50 (-8.4%)	2.50 (-9.6%)	0%	0%

TABLE III
PERFORMANCE COMPARISON

circuit	#signals propagated(% increase over LPA)			Run time in Seconds (% increase over LPA)		
	LPA	MPA	rMPA	LPA	MPA	rMPA
mux	1614	1836 (13.8)	1744 (8.1)	1.7	1.9 (11.8)	1.8 (5.9)
adder	1524	1597 (4.8)	1541 (1.1)	4.9	5.1 (4.1)	5.0 (2.0)
decoder	2636	2732 (3.6)	2638 (0.1)	3.25	3.37 (3.7)	3.29 (1.2)
contr3	4863	5476 (12.6)	5276 (8.5)	8.3	8.6 (3.6)	8.4 (1.2)
reg	32130	35150 (9.4)	32584 (1.4)	15.1	16.85 (12.3)	15.9 (5.3)
contr2	92792	98288 (5.9)	93511 (0.8)	35.6	41.93 (17.8)	39.14 (9.9)
contr1	58215	59820 (2.8)	58595 (0.7)	32.46	34.44 (6.1)	33.72 (3.9)

that while SPA has the property of making conservative approximations of the propagated signal shapes, these approximations will result in incorrect identification of the critical path, which therefore has a lesser SPICE delay than the true critical path. Accordingly, the discrepancy between circuit delay in Table I and the SPICE delay in Table II can be very large for SPA.

In Table III, the run time and the number of propagated signals for the traditional LPA method, the proposed MPA method, and the proposed rMPA methods are shown. The run time of SPA is not shown, since it is the same as that of LPA. The MPA method has a run time penalty of 8.5% on average over the traditional LPA method. On the other hand, the rMPA method diminishes the run time penalty to only 4% on average over the LPA method. In all cases, the rMPA method produced identical results with the MPA method, as expected.

Finally, the proposed methods were also used in transistor-sized optimization. In Fig. 13, we show the area/delay tradeoff produced during the optimization of circuit *mux* for both LPA and MPA. In the upper left corner of the tradeoff curve, generated by LPA toward the end of the optimization, saw teeth in the tradeoff curve are evident. These are the results of two paths that join at an internal node and have nearly equal arrival times at this node, while one path has a significantly slower slope at this node than the other path. As the optimization improves the arrival time at this node for each path in turn, the LPA algorithm switches back and forth between the two paths, creating abrupt changes in the computed circuit delay. The tradeoff curve produced by the proposed MPA method is smooth and free of such discontinuities, as shown in Fig. 13.

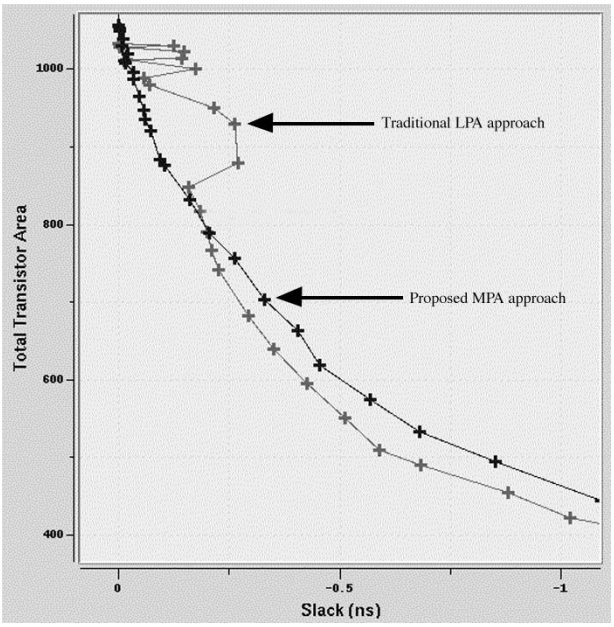


Fig. 13. Circuit optimization with LPA and proposed MPA methods.

VII. CONCLUSION

In this paper, we have shown that the traditional timing analysis approach, referred to as the latest propagation approach, can significantly underestimate the delay of a circuit due to its method of propagating arrival signals and slopes.

We also showed that this can lead to discontinuities in the circuit delay as a function of its transistor and gate sizes, which creates difficulties for circuit optimization tools. We then discussed an alternate propagation approach, referred to as the SPA, which combines the latest arrival time with the slowest slope of all signals at a node. We show that while this approach eliminates the discontinuity of timing graph delay, it significantly overestimates the delay of the circuit. In this paper, we presented a new algorithm that, given delay and slope functions for all gates in a circuit, computes the correct critical path in the timing graph and the correct delay of the timing graph. We also showed that this algorithm propagates the minimum number of possible arrival signals for a general timing graph. Then, based on a newly introduced property of digital gates, we showed that the number of propagated arrival signals can be reduced for digital circuits, without incurring an error. Finally, we discussed the computation of required times and slacks and presented associated algorithms for the proposed methods. We show that the traditional algorithms can compute an incorrect slack for the critical path, while the proposed algorithms will compute a slack for the critical path that is both correct and continuous with the transistor and gate sizes in the circuit. We also show that the traditional latest propagation algorithm computes an error for the noncritical nodes that can be both positive or negative, while the proposed algorithms will always incur a conservative (negative) error. The proposed algorithms were implemented in an industrial timing analysis and optimization tool and were tested on a number of industrial circuits. The results show that the traditional method can underestimate the delay of a circuit by as much as 39%, while the slowest propagation approach can overestimate the circuit delay by as much as 17%. We also show that the proposed algorithms increase the run time of timing analysis only marginally by 4% on average.

REFERENCES

[1] A. I. Kayssi, K. A. Sakallah, and T. N. Mudge, "The impact of signal transition time on path delay computation," *IEEE Trans. Circuits and Systems*, vol. 40, May 1993.

[2] C. Visweswariah and A. R. Conn, "Formulation of static circuit optimization with reduced size, degeneracy and redundancy by timing graph manipulation," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, 1999, pp. 244–251.

[3] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*. New York: Academic, 1983.

[4] R. B. Hitchcock, "Timing verification and the timing analysis program," in *Proc. IEEE/ACM Design Automation Conf.*, 1982, pp. 594–604.

[5] N. P. Jouppi, "Timing analysis for nMOS VLSI," in *Proc. IEEE/ACM Design Automation Conf.*, 1983, pp. 411–418.

[6] J. P. Fishburn and A. Dunlop, "TILOS: A posynomial programming approach to transistor sizing," in *Int. Conf. Computer-Aided Design*, Nov. 1985.

[7] S. Devadas, K. Keutzer, and S. Malik, "Computation of floating mode delay in combinational circuit: Theory and algorithms," *IEEE Trans. Computer-Aided Design*, Dec. 1993.

[8] Y. Kukimoto, W. Gosti, A. Saldanha, and R. Brayton, "Approximate timing analysis of combinatorial circuits under XBD0 model," in *IEEE Int. Conf. Computer-Aided Design*, 1997, pp. 176–181.

[9] H. Yalcin and J. P. Hayes, "Event propagation conditions in circuit delay computation," *ACM Trans. Design Automat. Electronic Syst.*, July 1997.

[10] S.-Z. Sun, D. H. C. Du, and H.-C. Chen, "Efficient timing analysis for CMOS circuits considering data dependent delays," in *Proc. Int. Conf. Computer Design*, 1994.

[11] N. Nedenstierna and K. O. Jeppson, "CMOS circuit speed and buffer optimization," *IEEE Trans. Computer-Aided Design Integrated Circuits Syst.*, vol. 6, pp. 270–281, Feb. 1987.

[12] T. Sakurai and A. R. Newton, "Alpha—Power law MOSFET model and its applications to CMOS inverter delay and other formulas," *IEEE J. Solid-State Circuits*, vol. 25, pp. 584–594, Apr. 1990.

[13] H.-Y. Chen and S. Dutta, "A timing model for static CMOS gates," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1989.

[14] J. T. Kong and D. Overhauser, "Methods to improve digital MOS macromodel accuracy," *IEEE Trans. Computer-Aided Design*, vol. 14, pp. 868–881, July 1995.

[15] T. Sakurai and A. R. Newton, "Delay analysis of series connected MOSFETs," *IEEE J. Solid-State Circuits*, vol. 26, pp. 122–131, Feb. 1991.

[16] D. H. C. Du, S. H. C. Yen, and S. Ghanta, *On the General False Path Problem in Timing Analysis*: DAC, 1989, pp. 555–560.

[17] P. C. McGeer and R. K. Brayton, "Efficient algorithms for computing the longest viable path in a combinational network," in *Design Automation Conf.*, 1989, pp. 561–567.

[18] H. Yalcin, J. P. Hayes, and K. A. Sakallah, "An approximate timing analysis of combinational circuits under XBD0 model," in *Int. Conf. Computer-Aided Design*, 1997, pp. 176–181.

[19] K. P. Belkhale and A. J. Suess, "Timing analysis with known false subgraphs," in *Int. Conf. Computer-Aided Design*, 1995, pp. 736–739.

[20] E. Goldberg and A. Saldanha, "Timing analysis with implicitly specified false path," in *Proc. Int. Workshop Timing Issues Specification and Synthesis of Digital Designs*, 1999, Paper TAU99.

[21] V. Chandramouli and K. A. Sakallah, "Modeling the effects of temporal proximity of input transitions on gate propagation delay and transition time," in *Proc. Design Automation Conf.*, 1996.

[22] Y.-H. Jun, K. Jun, and S.-B. Park, "An accurate and efficient delay time modeling for MOS logic circuits using polynomial approximation," *IEEE Trans. Computer-Aided Design*, vol. 9, no. 6, pp. 1027–1032, 1989.

[23] A. Nabavi-Lishi and N. C. Rumin, "Inverter models of CMOS gates for supply current and delay evaluation," *IEEE Trans. Computer-Aided Design*, vol. 13, no. 10, pp. 1271–1279, 1994.

[24] D. Blaauw, R. Panda, and A. Das, "Removing user-specified false paths from timing graphs," in *Proc. Design Automation Conf.*, 2000, pp. 270–273.

[25] S .S. Sapatnekar, V. B. Rao, P. M. Vaidya, and S.-M. Kang, "An exact solution to the transistor sizing problem for CMOS circuits using convex optimization," *IEEE Trans. Computer-Aided Design*, vol. 12, no. 11, pp. 1621–1634.

[26] A. Dharchoudhury, D. Blaauw, J. Norton, S. Pullela, and J. Dunning, "Transistor-level sizing and timing verification of domino circuits in the power PC™ microprocessor," in *Proc. Int. Conf. Computer Design*, 1997, pp. 143–148.

[27] G. De Micheli, *Synthesis and Optimization of Digital Circuits*. New York: McGraw Hill, 1994.

[28] A. Vittal, L. H. Chen, M. Marek-Sadowska, K-P. Wang, and S. Yang, "Crosstalk in VLSI interconnections," *Trans. on Computer-Aided Design*, vol. 18, pp. 1817–1824, Dec. 1999.

[29] A. B. Kahng, S. Muddu, and E. Sarto, "On switching factor based analysis of coupled RC interconnects," in *Proc. IEEE/ACM Design Automation Conf.*, 2000, pp. 79–84.

[30] P. D. Gross, R. Arunachalam, K. Rajagopal, and L. T. Pileggi, "Determination of worst-case aggressor alignment for delay calculation," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, Nov. 1998, pp. 212–219.

[31] S. Sapatnekar, "Capturing the effect of crosstalk on delay," in *Proc. VLSI Design 2000*, Jan. 2000, pp. 364–369.

[32] Y. Sasaki and G. De Micheli, "Crosstalk delay analysis using relative window method," in *IEEE Int. ASIC/SOC Conf.*, 1999, pp. 9–13.

[33] R. Levy, D. Blaauw, G. Braca, A. Dasgupta, A. Grinshpon, C. Oh, B. Orshav, S. Sirichotiyakul, and V. Zolotov, "Clarinet: A noise analysis tool for deep submicron design," in *Proc. IEEE/ACM Design Automation Conf.*, June 2000, pp. 233–238.

[34] S. Sirichotiyakul, D. Blaauw, C. Oh, R. Levy, V. Zolotov, and J. Zuo, "Driver modeling and alignment for worst-case delay noise," in *IEEE/ACM Design Automation Conf.*, 2001.

David Blaauw (M'94) received the B.S. degree in physics and computer science from Duke University, Chapel Hill, NC, in 1986 and the M.S. and Ph.D. degrees, both in computer science, from the University of Illinois, Urbana, in 1988 and 1991, respectively.

He worked at the Engineering Accelerator Technology Division, IBM Corporation, Endicott, until August 1993. From 1993 until August 2001, he worked for Motorola, Inc., Austin, TX, where he was the Manager of the High Performance Design Technology group. Since August 2001, he has been on the faculty at the University of Michigan as an Associate Professor. His work has focused on VLSI design and CAD with particular emphasis on circuit analysis and optimization problems for high performance and low power designs.

Dr. Blaauw received the Best Paper Award at the ACM/IEEE Design Automation Conference in 2000. He was the Technical Program Chair and General Chair for the International Symposium on Low Power Electronic and Design in 1999 and 2000, respectively, and was the Technical Program Co-Chair and member of the Executive Committee for the ACM/IEEE Design Automation Conference in 2000–2002.

Vladimir Zolotov (M'96) received the M.S. degree in electrical engineering from the Moscow Institute of Electronics, Moscow, Russia, and the Ph.D. degree in electrical engineering from the Scientific Research Institute of Micro Devices, Moscow, Russia.

He has been with the Advanced Tools group, Motorola, Inc., Austin, TX, since 1998, where he is currently a Senior Staff Engineer/Scientist involved in the development of EDA tools and methodology for high-performance VLSI designs. Previously, he was with the Scientific Research Institute of Micro Devices, and then with the Moscow Research Laboratory, Motorola, Inc., Moscow, Russia. His research interests include signal integrity, reliability, on-chip inductance, SOI circuits, timing analysis, and optimization of VLSI.

Savithri Sundareswaran (M'01) received the B.E. and M.S. degrees in electrical enginering and physics from Birla Institute of Technology and Sciences, India.

She has been with Motorola, Inc., since 1995, working in the areas of timing analysis, optimization, and CAD algorithms for VLSI. She is currently leading the development of tools and methodology for power grid analysis at Advanced Tools Group, Motorola, Austin, TX. Prior to Motorola, she was a Scientist with Central Electronics Engineering Research Labs in India.