Slope Propagation in Static Timing Analysis

David Blaauw, Vladimir Zolotov, Savithri Sundareswaran*, Chanhee Oh and Rajendran Panda
Motorola Inc. Austin, TX, *Motorola India Electronics Ltd., Bangalore, India

1 Abstract

Static timing analysis has traditionally used the PERT method for identifying the critical path of a digital circuit. Due to the influence of the slope of a signal at a particular node on the subsequent path delay, an earlier signal with a signal slope greater than the slope of the later signal may result in a greater delay. Therefore, the traditional method for timing analysis may identify the incorrect critical path and report an optimistic delay for the circuit. We show that the circuit delay calculated using the traditional method is a discontinuous function with respect to transistor and gate sizes, posing a severe problem for circuit optimization methods. We propose a new timing analysis algorithm which resolves both these issues. 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. The algorithm for propagating the corresponding required times is also presented. We prove that the proposed algorithm identifies a circuit's true critical path, where the traditional timing analysis method may not. We also show that under this method circuit delay and node slack are continuous functions with respect to a circuit's transistor and gate sizes. In addition, we present a heuristic method which reduces the number of signals to be propagated at the expense of a slight loss in accuracy. Finally, we show how the proposed algorithm was efficiently implemented in an industrial static timing analysis and optimization tool, and present results for a number of industrial circuits. Our results show that the traditional timing analysis method underestimates the circuit delay by as much as 38%, while that the proposed method efficiently finds the correct circuit delay with only a slight increase in run time.

2 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. It is therefore applicable only to small circuits and tends to be error prone. 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] and logic synthesis tools.

In static timing analysis, the so-called arrival times or *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, the so-called *required times*, which represent the latest time a signal can transition at a node in order to meet performance constraints, are propagated from circuit outputs to inputs. An arriving signal consists of both the *crossing time,* when the signal reaches the 1/2 Vdd point, and the slope of the signal. As signals are propagated across a gate, their crossing times and slopes are updated.

In recent years, extensive research has focused on how to efficiently and accurately calculate propagation delays and slopes for gates in a circuit [1], as well as on methods to eliminate false paths, which are unrealizable due to logic and timing correlations in a circuit [7], [8]. However, the essential principle of static timing analysis has remained largely unchanged since it was proposed in the eighties by [5], [4] and is still based on two fundamental assumptions:

The first assumption is that, when calculating the delay of a gate, only one input of the gate is switching at a time. This results in a calculated gate delay that may be smaller than the actual delay when multiple inputs switch simultaneously, and may therefore yield an overly optimistic timing analysis report. In [9], this problem was discussed and a solution was proposed.

The second assumption is that at each node, the arriving signal with the latest crossing time results in the longest path delay and is therefore propagated forward, while all earlier arriving signals are not. This assumption is the topic of this paper. The traditional implementation of propagating only the latest arriving signal is referred to as the *latest propagation algorithm (LPA).*

LPA selects one signal for forward propagation out of all signals arriving at a node. The basic problem with LPA is that it makes this selection based only on the crossing time of the arriving signals without regard to their slopes. The slope 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 crossing time might well have a larger overall path delay if it has a significantly larger signal slope. Such a signal would not be propagated under LPA and its path would remain undetected, resulting in an underestimation of the worst circuit delay. To illustrate this problem, we have shown in Figure 1(a) a simple two-input circuit with two possi-

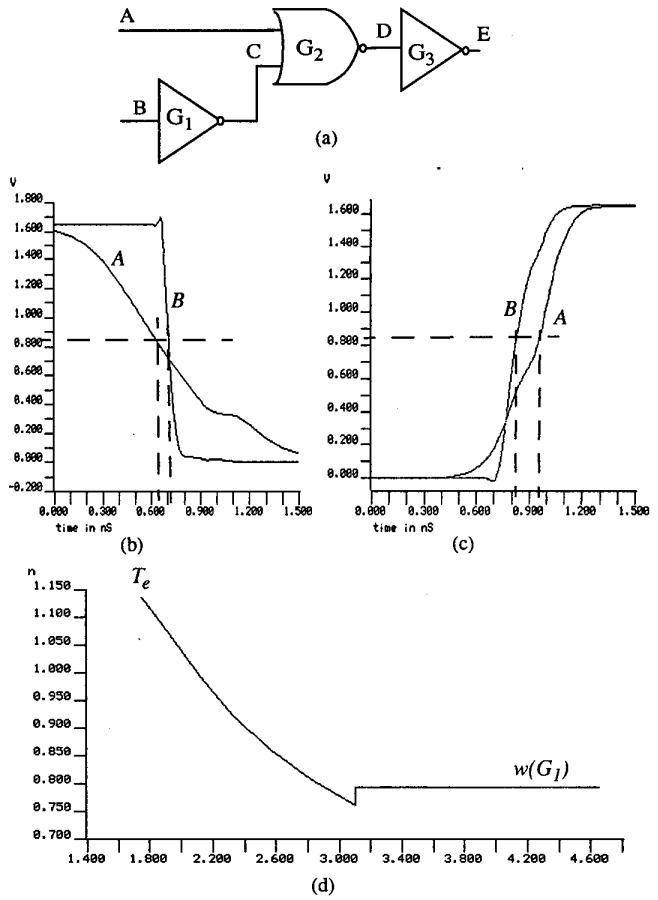

Figure 1. Error in calculated circuit delay with LPA method

ble signal paths, one originating from input A (signal A) and one originating from input B (signal B). Figure 1(b) shows the Spice waveforms and the associated crossing time and slope of the two signals at node D. Since the crossing time of signal B (0.7ns) is later than that of signal A (0.64ns), LPA propagates signal B through gate G3 resulting in a total path delay of 0.82ns as shown in Figure 1(c). However, the slope of arrival signal A (1.36ns) is larger than that of signal B (0.1ns), and would result in a significantly larger delay of gate G3 if signal A was propagated instead of signal B. The total path delay of signal A would therefore be 0.95ns, as shown in Figure 1(c). For this simple circuit, traditional timing analysis using LPA reports a worst circuit delay of 0.82ns, while the actual worst circuit delay is 0.95ns. Although LPA correctly calculated the path delay of signal B, it did not detect that signal A resulted in a longer path delay than signal B and therefore identified the wrong critical path and underestimated the total circuit delay. It should be noted that this error is independent of the delay model provided the model accounts for the influence of signal slope on gate delay, as is the case for all modern delay models. In the above example, the circuit was simulated with Spice.

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 in the circuit. This is illustrated in Figure 1(d) where the worst circuit delay 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 crossing time of signal B at node D becomes earlier than that of signal A. At this point, the signal propagated by LPA switches from signal A to signal B and the slope used to calculate the delay of gate G3 changes abruptly from 0.1ns to 1.36ns. 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]. Discontinuities tend to trap such optimization methods far from an optimal circuit solution. To address this problem, a recently proposed optimization method [2] propagates the latest arriving signal, but modifies its input slope to be the maximum slope of all signals arriving at a node. This guarantees the continuity of the objective function, but can significantly overestimate circuit delay. In the example of Figure 1, the propagated arrival signal would have a crossing time of 0.7ns and a slope of 1.36ns, resulting in an overestimation of circuit delay and a sub-optimal optimization result.

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 crossing times very close to one another. However, they may have dramatically different slopes, and LPA is therefore more likely to select the wrong signal for forward propagation and report an optimistic worst circuit delay. Hence, there is a critical need to address this issue in static timing analysis.

In this paper, we propose a new signal propagation method which is guaranteed to identify the worst circuit 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. An associated algorithm for backward propagation of required times is also presented to allow the calculation of slacks for all nodes in the circuit. We shall prove that the proposed algorithm identifies the correct worst delay of the circuit and also that the calculated worst circuit delay is continuous with respect to gate/transistor sizes. Since the proposed algorithm increases the number of propagated signals and required times, it increases the run time of the analysis compared to the LPA method. However, we show that for digital circuits an upper bound can be calculated for the added delay due to differences in the slope of signals, and we use this proposed bound to reduce the number of propagated signals. With this bound, the increase in the run time over LPA proves to be small in practice. Based on experiments on several industrial circuits, we show that LPA can underestimate the worst path delay by as much as 38%. We also demonstrate the occurrence of discontinuities in the worst circuit delay when sizing using LPA and show how these discontinuities are removed with the proposed propagation algorithm.

The remainder of this paper is organized as follows: In Section 2 we present a formal formulation of the timing analysis problem. In Section 3 we present the newly proposed propagation algorithms, and prove that they correctly identify the worst circuit delay. We also present a delay bound for reducing the run time for digital circuits. In Section 4 we present our results, and in Section 5 our conclusions.

3 Problem Formulation

In this section, we present a formal definition of a timing graph and the latest propagation algorithm. For the purpose of our 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 to these methods as well.

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 $N=\{n_1,n_2,...,n_k\}$ is a set of nodes, $E=\{e_1,e_2,...,e_l\}$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 between gate inputs and 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 also assume without loss of generality that signal crossing times are measured at 50% of the signal level.

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 slope function $s_e=s_e(s_I)$, which represents the slope of the signal at the gate's output. Both are functions of the gate input slope s_I 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 slope $s_e(s_a)<s_e(s_b)$.

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 $P=(n_a,n_b,...,n_z)$ such that each pair of adjacent nodes n_g and n_h has an edge $e_{gh}=(n_g,n_h)$.

A path $P=(n_a,n_b,...,n_z)$ defines a sequence of edges $(e_{ab},e_{bc},...,e_{yz})$. 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.

Finally, among all paths terminating at a node, we define the path with the maximum crossing time as the critical path up to that node:

Definition 4. A path having the maximum delay among all paths with the same ending node is called critical.

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. The main objective of timing analysis is to find the correct critical path in a timing graph and to compute its

delay. It is clear that this critical path is the limiting factor for the performance of a circuit, and that its delay must be decreased in order to increase circuit performance. We will now show that the actual delay of a timing graph is a continuous function with respect to gate delays. 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.

The most obvious technique for finding the critical path of a given 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. The traditional approach for finding the critical path in a circuit is based on the PERT algorithm and uses the propagation of signals from the source node to the sink node. We define a signal as follows:

Definition 5. A signal S_n at a node n is a quadruplet $S_n=\{n, T_A, s, P\}$ where n is the node at which the signal is situated, T_A is its crossing time at the node n, s is its slope at the node n, and $P=(n_s, n_a, ..., n)$ is the signal propagation path from the source node n_s to the node of interest n.

The traditional timing analysis algorithm iterates through each node in a timing graph in topological order, selecting the signal with the latest crossing time from among all incident signals for forward propagation. As a signal is propagated forward, its crossing 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 have referred to this algorithm as the latest propagation algorithm (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 Figure 2.

In Section 1, we already presented a small example circuit for which the latest propagation algorithm identifies the wrong critical path in the timing graph. We now show below in Lemma 1 that, given two signals at a particular net, the path delay of any path from this net to the sink node of the timing graph will be greater for the signal with the slower signal slope. We then show in Theorem 2 that if this signal with the slower slope also had an earlier arrival time, this signal 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 graph delay calculated by LPA can be discontinuous with respect to the gate sizes.

1. Assign to the source node n_0 the signal $S_0=\{n_0, T_0, s_0, P_0\}$ where $T_0=0$, $P_0=(n_0)$
2. Visit each node, i, in the graph in topological order doing the following:
   2.1. For each incoming edge e_{ki} to node i from node k with signal $S_k=(n_k, T_k, s_k, P_k)$, create a new signal $S_{ki}=(n_i, T_{ki}, s_{ki}, P_{ki})$ 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 signal S_{ki} select signal $S_{latest}=(n_i, T_{latest}, s_{latest}, P_{latest})$, where $T_{latest} = max(T_{ki})$
   2.3. Assign the computed signal S_{latest} to node n_i.

Figure 2. Traditional arrival signal propagation algorithm.

Lemma 1. Given two signals $S_a=\{n, T_a, s_a, P_a\}$ and $S_b=\{n, T_b, s_b, P_b\}$ at node n, where $s_a<s_b$ then for any signal path Q from node n to node 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.

Proof. Lemma 1 follows directly from Property 1 of delay and slope functions, the recurrent dependence of gate output slopes on gate input slopes, and the additive property of path delay in Definition 3.

Theorem 2. If, for some node n_i of timing graph G, LPA selects the signal $S_{latest}=(n_i, T_{latest}, s_{latest}, P_{latest})$ and the slope s_{latest} is less than the slope s_k of another signal $S_{ik}=(n_i, T_k, s_k, P_k)$ 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 in H S_{ik} is critical but S_{latest} is not.

Proof. 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 $e_k=(n_k, n_f)$ for each node n_k of H that does not have an outgoing edge (including node n_i). To all edges e_k we assign delay functions $d_i(s)=0$, except for $e_i=(n_i, n_f)$. 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 $e_i=(n_i, n_f)$ we assign a delay function $d_i(s)$ such that $d_i(s_{latest})=d_{max}$ and $d_i(s_k)=d_{max}+T_{latest}-T_k+\Delta$, where $\Delta > 0$. Note that this delay function does not violate Property 1. From this construction it is clear that signal S_{ik} will arrive at the sink node of timing graph H later than signal S_{latest}.

From Theorem 2 we obtain the following corollary:

Corollary 1. There exist timing graphs for which LPA computes an incorrect critical path and delay.

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 a 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 S_{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.

4 Proposed Propagation Algorithm

In order to perform a timing analysis which 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. Only if two arrival signals are incident on a node, such that one of the signals has both an earlier crossing time and a faster slope, is this signal pruned from the analysis. We prove that this algorithm finds the correct critical path and 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 circuit delay for some general timing graph. However, given some additional properties of the signal propagation through digital circuits, we can bound the delay added due to the slope difference between two signals. In Section 3.2 we show how this allows us to significantly reduce the number of propagated signals for this specific class of timing graphs. In Section 3.3 we present the algorithm for propagating a required time backward from the sink node to the source node. This is necessary to calculate slacks for all circuit nodes.

4.1 Arrival Signal Propagation

The proposed arrival propagation algorithm is shown below, in Figure 3. Note that in steps 2.2 and 2.3, a set of signals is propagated forward instead of a single signal as in LPA. We now prove in Theorem 4 and 5 that the proposed algorithm identifies the correct critical path, in Corollary 2 that the calculated graph delay is a continuous function of edge

delays, and in Theorem 6 that the number of propagated arrival signals is minimal.

1. Assign to source node n_0 the signal set $C_0=\{S_0\}$, where $S_0=\{n_0, T_0, s_0, P_0\}$, $T_0=0$, $P_0=(n_0)$
2. Visit each node n_i in topological order and compute its 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 $S_{kij}=(n_i,T_{kij},s_{kij},P_{ki})$, 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 the signal set C_i any signal $S_{ij}=\{n_i, T_{ij}, s_{ij}, P_{ij}\}$ if C_i has another signal $S_{ik}=\{n_i, T_{ik}, s_{ik}, P_{ik}\}$ such as $T_{ij}<T_{ik}$ and $s_{ij}<s_{ik}$

Figure 3. Proposed Signal Propagation Algorithm.

Theorem 4. For any timing graph G and for any of its nodes n_i, any signal $S_{ij}=\{n_i, T_{ij}, s_{ij}, P_{ij}\}$ that is pruned by the proposed algorithm does not have the latest crossing time at any node n_l following node n_i.

Proof. If, in the proposed algorithm, we prune signal $S_{ij}=\{n_i, T_{ij}, s_{ij}, P_{ij}\}$, then at this node n_i there exists another signal $S_{ik}=\{n_i, T_{ik}, s_{ik}, P_{ik}\}$ 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 delay along the path of signal S_{lj} from node n_i to node n_l will always be less than the edge delay along the same path for signal S_{lk}. Since the crossing 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 5. The proposed algorithm correctly calculates the critical path and delay of a timing graph.

Proof. From Theorem 4, 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 2. 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 the 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.

Theorem 6. If, for some node n_i of timing graph G, the proposed algorithm selects the signal $S_{ij}=(n_i,T_{ij},s_{ij},P_{ij})$, we can construct a new graph H containing all nodes and edges in G that have already been visited by the algorithm, such that the critical path of H includes the path P_{ij}.

Proof. The proof for this theorem is similar to that of Theorem 2, and is omitted for brevity.

Theorem 6 shows that if, at the time of pruning at node n_i, there is no information available regarding the timing graph beyond node n_i, it is necessary for the algorithm to propagate all selected signals. It is therefore impossible to reduce the number of propagated signals further without possibly incurring a wrong solution. If, however, certain properties (which we discuss in the next section) can be assumed for this portion of the timing graph, the number of propagated signals can be reduced further.

4.2 Reduction in Propagated Signals

If the proposed algorithm is used for the analysis of digital circuits, we can utilize some well-known 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 Figure 4(a). We define the following 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 instance 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 instance along its transition. Note that Property 2 is stronger than Property 1, and that it may not hold for certain analog circuits or for circuits where parasitic inductance and coupling capacitance dominate the signal delay. 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 Figure 4(a) from a typical gate of a 0.13um, 2Ghz digital processor.

We now consider two signals S_{ia} and S_{ib} at a node n_i with the same crossing time but with different slopes, s_{ia} and s_{ib}, signal S_{ib} having the slower slope, as shown in Figure 4(b). We would like to calculate a bound δ on the difference in crossing 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_{ic} has a later crossing time at node n_f than signal S_{fb} and therefore the difference in crossing times of S_{ia} and S_{ic} at node n_f is an upper bound on the difference in the crossing times of S_{ia} and S_{ib} at node n_i. Since S_{ia} and S_{ic} have identical slopes, it is clear that the bound δ is exactly the difference in the crossing times of S_{ia} and S_{ic} at node n_i, which is $(s_{ib} - s_{ia}) / 2$.

Using δ, we can prune from the propagation any signal $S_{ia}=\{n_i, T_{ia}, s_{ia}, P_{ia}\}$ if another signal $S_{ib}=\{n_i, T_{ib}, s_{ib}, P_{ib}\}$ exists such that $T_{ib} - T_{ia} > \delta$, or $T_{ib} - T_{ia} > (s_{ia} - s_{ib}) / 2$. In this case, signal S_{ia} is earlier than signal S_{ib} to such an extent that the added delay of S_{ib} in the path from n_i to n_f will not render it critical. Use of this condition in step 2.3 of the proposed algorithm limits the propagated signals to a small window of crossing times preceding the latest crossing time and significantly reduces the number of signals propagated through the timing graph. This condition guarantees the correct calculation of the critical path and graph delay for circuit where the gates comply with Property 2, which holds for all digital circuits. The proof is omitted for brevity. Note that even if Property 2 does not hold, the obtained timing result will be at least as

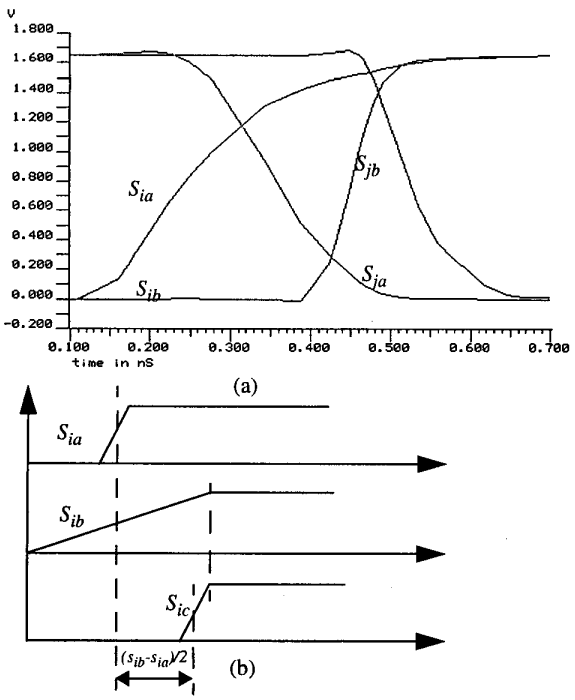

Figure 4. Bound on added path delay

accurate with LPA, since the signal with the latest crossing time is always be propagated.

We now examine the runtime complexity of the proposed algorithm. In step 2.3, a subset of all signals incident on a node is selected to be propagated forward. This operation involves the sorting of all signals according to their crossing 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 in step 2.1 is in the worst case equal to the sum of all path lengths 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 order in terms of decreasing crossing time and increasing signal slope is identical, thereby not allowing any prunning. 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.

4.3 Backward Propagation of Required Times

Many optimization algorithms require the calculation of node slacks for all nodes in a circuit. To accomplish this, required times must be propagated backward from sink node n_f to source node n_0, and the slack at each node must be calculated as the difference between the required time and the crossing time of a signal. As required times are propagated backward across edges, their crossing times are decremented by the edge delay. For each signal that is propagated forward, it is therefore necessary to have an associated required time that is propagated backward such that this required time is decremented by the same delays as the crossing time of its forward propagated signal was incremented. This way, the arrival signal and associated require times are updated with the same edge delays as they are propagated, and remain consistent. A possible implementation involves storing multiple edge delays for each edge in the timing graph, and tagging each edge delay and associated arrival signal and required time with a unique identifier.

For an arrival signal S_{ia} that is pruned at node n_i, a new required time T_{ia} must be created during the backward propagation as shown in Figure 5. For this purpose, we select from among the propagated signals at node

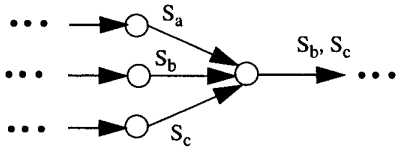

Figure 5. Propagation of required times.

n_i the signal S_{ip} with a slope s_{ip} that has the least slope greater than S_{ia} of all propagated signals. The required time T_{ip} of signal S_{ip} is then propagated backward as T_{ia}. In Figure 5, required time T_{ic} is selected from among the two propagated required times T_{ib} and T_{ic}. Using the above criteria for creating the required time of a pruned signal, the selected signal S_{ip} has a greater slope than the pruned signal S_{ia}. Therefore, the required time T_{ip} will overestimate the delay between node n_i and n_f, meaning that T_{ip} will be earlier than the exact required time of S_{ia}, resulting in an over-estimation of the slacks on the path P_{ai} of signal S_{ia}. This guarantees the important condition that the slack along the sub-critical path of S_{ia} is always higher than the slack of the critical path at node n_i.

5 Results

The proposed algorithms were implemented in an industrial transistor-level static timing analysis and optimization tool. Both the *basic* algorithm presented in Figure 3 and the *extended* 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 1, we show the circuit delay calculated by the traditional LPA

Circuit	# transistors	Total Circuit Delay in nS		
		LPA	New	%Error in LPA
mux	784	0.88	0.99	10%
adder	1,074	0.55	0.87	36%
decoder	1,490	2.11	3.47	38%
contr3	3,190	2.63	3.22	18%
reg	4,902	3.83	4.66	17%
contr2	11,112	8.49	9.26	8%
contr1	12,519	2.07	2.26	8%

Table 1. Effect of slope propagation on estimated delay

method and the proposed exact method from this paper. The table demonstrates that the LPA method underestimates the circuit delay by as much as 38% for the decoder circuit and by 19% on average. It is clear that this is a significant error that can not be ignored.

In Table 2, the run time and the number of propagated signals for the traditional LPA method, the proposed basic method, and the proposed extended method are shown. The exact method has a run time penalty of 4.1 - 17.8% over the traditional LPA method. On the other hand, the extended method reduced the run time penalty to only 1.2 - 9.9% over the LPA method. In all cases, the extended method produced identical results with the basic method, as expected. Finally, the proposed methods were also used in transistor size optimization. In Figure 6, we show the area/delay trade-off produced during the optimization of circuit mux for both the LPA and the proposed method. The discontinuities in the circuit delay are evident towards the end of the trade-off curve produced

by LPA. The trade-off curve produced by the proposed method is free of such discontinuities.

circuit	#signals propagated(% increase over LPA)			Run time in Seconds (% increase over LPA)		
	LPA	New	New, Pruned	LPA	New	New, Pruned
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	16.8(12.3)	15.9(5.3)
contr2	92792	98288(5.9)	93511(0.8)	35.6	41.9(17.8)	39.1(9.9)
contr1	58215	59820(2.8)	58595(0.7)	32.5	34.4(6.1)	33.7(3.9)

Table 2. Performance Comparison

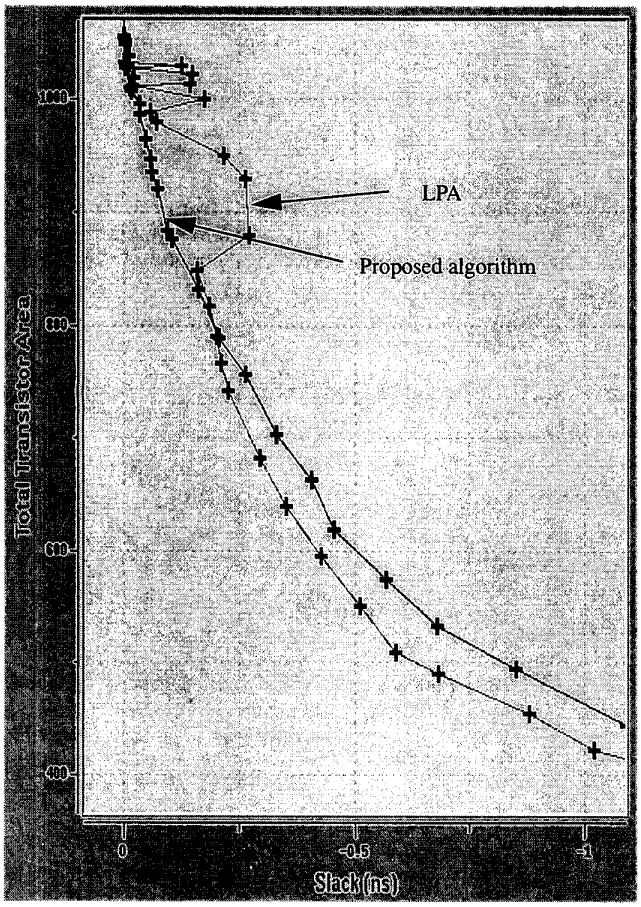

Figure 6. Circuit Optimization with LPA and new method

6 Conclusions

In this paper we have shown that the traditional timing analysis approach can significantly underestimate the delay of a circuit, due to its method of propagating arrival signals. We also showed that this can lead to discontinuities in the circuit delay as a function of its transistor sizes, which creates difficulties for circuit optimization tools. We therefore presented a new algorithm that addresses this problem and is proved to correctly calculate the critical path in a circuit and its circuit delay. We also showed that this algorithm propagates the minimum number of possible arrival signals for a general timing graph. Then, based on the specific properties of digital logic gates, we showed that the number of propagates arrival signals can be further reduced for digital circuits, without incurring an error. Finally, we presented the algorithm for propagating the required times in a manner consistent with the arrival signal propagation to enable the calculation of node slacks at all circuit nodes. The proposed algorithms were implemented in an industrial timing analysis and optimization tool and were tested on a number of processor circuits. The results show that the traditional method can underestimate the actual delay of a circuit by as much as 38%. We also show that the proposed algorithms increase the run time of timing analysis only marginally by 10%.

7 References

[1] Ayman I. Kayssi, Karem A. Sakallah, Trevor N.Mudge The Impact of Signal Transition Time on Path Delay Computation, IEEE Transactions on circuits and systems-II: Analog and digital signal processing, Vol. 40, No. 5, May 1993

[2] Chandu Visweswariah, Andrew R.Conn, Formulation of Static Circuit Optimization with Reduced Size, Degeneracy and Redundancy by Timing Graph Manipulation, Proc. IEEE/ACM ICCAD, 1999, pp.244-251.

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

[4] Hitchcock, R.B. Timing verification and the Timing Analysis program, Proc., IEEE/ACM DAC, 1982, pp.594-604

[5] Jouppi, N.P. Timing analysis for nMOS VLSI, IEEE/ACM Design Automation Conf., 1983, pp. 411-418

[6] J.P.Fishburn, A.Dunlop, "TILOS: A posynomial programming approach to transistor sizing", ICCAD, Nov 1985

[7] S.Devadas, K.Keutzer, S.Malik, "Computation of Floating Mode Delay in Combinational Circuit: Theory and Algorithms", IEEE Trans. on Computer Aided Design, Dec 1993.

[8] Y.Kukimoto, W.Gosti, A.Saldanha, R.Brayton, "Approximate Timing Analysis of Combinatorial Circuits under XBD0 Model", ICCAD, 1997, pp. 176-181

[9] H.Yalcin, J.P.Hayes, "Event propagation conditions in circuit delay computation", ACM Transactions on Design Automation of Electronic Systems, July 1997