Static Timing Analysis using Backward Signal Propagation

Dongwoo Lee, *Vladimir Zolotov, David Blaauw
University of Michigan, Ann Arbor, MI
*Motorola Inc. Austin, TX

Abstract

In this paper, we address the problem of signal pruning in static timing analysis (STA). Traditionally, signals are propagated through the circuit and are pruned, such that only the signal with the latest arrival time at each node is propagated forward. This signal pruning is a key to the linear run time of STA. However, it was previously observed that a signal with the latest arrival time may not be the most critical signal, as an earlier signal with a larger transition time can result in a longer delay in the down-stream logic. Hence, arrival time based pruning can result in an optimistic delay, incorrect critical paths, and discontinuities of the delay during circuit optimization. Although algorithms were proposed to remedy this issue, they rely on propagation of multiple signals and have an exponential worst-case complexity. In this paper, we propose a new timing analysis algorithm, which uses a two pass traversal of the circuit. In the initial backward traversal, we construct delay tables which record the required time at a node as a function of the transition time at that node. This is followed by a forward traversal where signals are pruned not based on arrival times but based on slack. The proposed algorithm corrects the accuracy problems of the arrival time based pruning while at the same time maintaining the linear run time of STA. We implemented our algorithm and demonstrated its accuracy and efficiency.

Categories and Subject Descriptors
B.8.2 [Performance and Reliability]: Performance analysis
General Terms
Algorithms, performance, reliability

1 Introduction and Overview of Approach

Static timing analysis has become the primary tool for performance verification in the semiconductor industry. It is based on the propagation of signals through the circuit, where each signal is comprised of an arrival time and a transition time. During the traversal of the circuit, signals are pruned, such that only the signal with the latest arrival time is selected for further propagation to the down stream gates. Since only a single signal is propagated for each node in the circuit, the run time of STA is linear with the circuit size. It is this linear run time characteristic that has allowed STA to be applied to very large designs efficiently and is responsible in large part for its industrial success. Although the forward propagation of signals is sufficient for computing the maximum delay of the circuit, it has been common to also propagate so-called required time signals in backward direction through the circuit. By subtraction the arrival time from the required time, slack is computed at each node, which is useful for circuit optimization.

In the last two decades since the inception of STA [1][2], extensive work has focussed on improving the accuracy of the analysis. These works have focussed on different aspects, such as identification of false paths [3][4], delay model accuracy [5], simultaneous input switching effects [6], and different noise effects [7][8][9]. Hence, the accuracy of STA has improved significantly making it feasible to use STA as a sign-off tool for high-performance designs. However, despite these improvements, the basic arrival time based signal pruning algorithm that is employed in STA has remained relatively unchanged.

It was previously noted that the arrival time based pruning may select the incorrect signal for forward propagation [10]. The objective of signal pruning is to select for each node the signal that will result in the overall longest delay at the primary outputs of the circuit. We refer to this signal as the "critical signal". However, the critical signal may not be the signal with the latest arrival time. This is illustrated below in Figure 1, where two different signals are propagated through a small circuit, representing two possible paths from the inputs to the output (A_1-A_3-A_4 or B_2-B_3-B_4). Signal A originates from input n_1 and signal B originates from input n_2. When both signals reach node n_3, the pruning algorithm selects one of them for forward propagation. Since the arrival time of signal B_3 is greater then the arrival time of A_3, the traditional arrival time based pruning algorithm will select signal B_3, discarding signal A_3. However, the transition time of signal A_3 is greater than that of signal B_3. Hence, the delay of gates g_1 through g_2 will be larger for signal A than that for signal B, and signal A has an overall larger delay then signal B.

It is therefore clear that the traditional arrival time pruning method may select the incorrect arrival time for forward propaga-

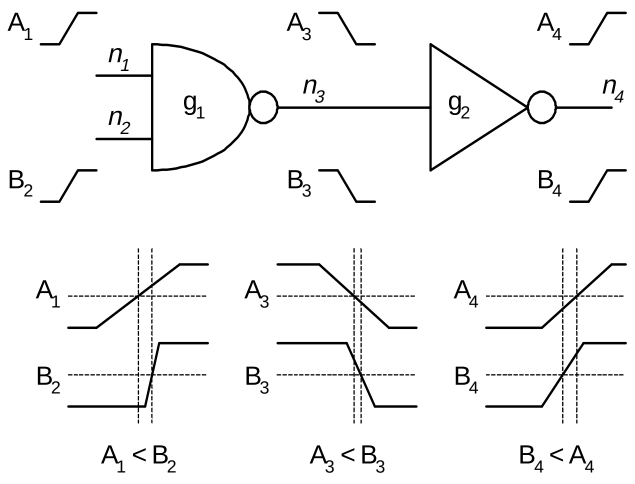


Figure 1. Circuit example with incorrect signal pruning.

tion. This can result in the computation of an optimistic circuit delay in STA since critical signals are lost. In this case, the computed critical path will also be incorrect, leading to possibly erroneous circuit optimization. Finally, the arrival time based pruning can results in discontinuities of the circuit delay as a function of the gate delays. This discontinuity arises when the delay of the critical path changes suddenly as the pruning algorithm switches from propagating one arrival time to another during circuit optimization. Such discontinuities can cause severe convergence problems for circuit optimization algorithms.

To address this problem in STA, several approaches have been proposed. In [10], an STA algorithm was proposed using multiple signal propagation, thereby guaranteeing that the critical signal is propagated. A pruning criteria was presented to reduce the number of propagated signals, however, the worst-case complexity of the approach remains exponential with circuit size. Also, the algorithm requires substantial change to the implementation of STA. In [11], an algorithm is proposed where a new signal is constructed for propagation by combining the latest arrival time among all signals with the largest transition time among all signals. This constructed signal is guaranteed to result in a worse overall circuit delay than any of the true signals at a node. This approach therefore results in the computation of a pessimistic circuit delay, but removes the discontinuity problem present in arrival time based pruning. An additional draw back of this approach is that the critical path can no longer be clearly identified in a circuit since the propagated signals are constructed by combining aspects of multiple signals. More recently, this method was extended in [12] to reduce the conservatism of the approach.

In this paper, we present a new STA algorithm. The approach is unusual in that it first performs the backward traversal of the circuit, followed by a forward traversal, unlike traditional STA which first performs a forward traversal followed by a backward traversal. During the backward traversal, the required time is propagated from the outputs of the circuit to the inputs. However, the gate delays, and hence the required time, is a function of the transition time at a node. Therefore, we construct a table of the required times at each node as a function of the transition time at that node. We show how such a required time table can be constructed in a single traversal of the circuit.

After the backward traversal, we perform a forward propagation of signals in the circuit, pruning signals at each node. However, counter to arrival time pruning, we select the signal with the worst slack for propagation. Since for each signal at a node the transition time is available, we can compute the required time for that signal using the required time table, and hence the slack of the signal. The slack of the signal represents its delay both leading up to that node, as well as from that node to any of the primary outputs of the ciru-cit. Therefore, the signal with the worst slack is guaranteed to be the critical signal. We show that our approach eliminates the inaccuracy of arrival time based pruning while at the same time maintain the linear run time of timing analysis. To our knowledge, this is the first linear run time solution to this problem. We implemented our proposed approach and demonstrated its accuracy in comparison to exhaustive path traversal and compare its efficiency to that of traditional STA.

The remainder of this paper is organized as follows. In section 2 we define basic concepts and the traditional STA approach. In Section 3, we present our proposed STA algorithm, including its properties and applications. In Section 4, we present experimental results and in Section 5, we conclude the paper.

2 Traditional STA Model and Definitions

In this section, we present a formal definition of a timing graph and the traditional timing analysis algorithm using arrival time based pruning. 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 to these methods as well.

We first define the so-called timing graph, on which a timing analysis algorithm is performed.

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. Without loss of generality, we also assume 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 transition time function $s_e=s_e(s_i)$, which represents the transition time of the signal at the gate's output, where s_i is the transition time at the gate's input.

We now define a path in timing graph G, and its path delay as follows:

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 transition time s_a at the first node n_a of path P, we can determine the signal transition times 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 transition time at node n_j, s_i is the transition time at the predecessor node n_i, and s_{ij} is a transition time function of the edge e_{ij}. After the signal transition time 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 transition time 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 arrival 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 from the source to 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 delay must be decreased in order to increase circuit performance. Also, delay of a timing graph is a continuous function with respect to its gate delays [10]. This property is important for circuit optimization methods, since many of such methods rely on their objective function being continuous.

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. A signal is defined as follows:

Definition 5. A signal S_n at a node n is a triplet $S_n=\{T_A, s, P\}$ where n is the node at which the signal is situated, T_A is its arrival time at the node n, s is its transition time 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 is shown in Figure 2. The algorithm iterates through nodes in a timing graph in topological order. As a signal is propagated forward through a timing edge e_{ki}, its arrival time is increased by the edge delay $d_{ki}(s_k)$ and its transition time is replaced with $s_{ki}(s_k)$, where s_k is the transition of the signal at the fanin node of the edge. The algorithm then selects the signal with the latest arrival time and assigns it to node i. 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. Since only a single signal is propagated through each edge, arrival time based STA has a run time complexity that is linear with the number of edges in the timing graph.

3 Proposed STA Approach

Instead of traversing the timing graph in direct topological order, the proposed approach performs a traversal in reverse topological order. Also, instead of computing arrival times at nodes the proposed slack based STA algorithm computes delays from each node to the sink node. The delay from a node to the sink node is dependent on the transition time at that node. Hence, we compute delays from a node to the sink node as functions of transition time. Although it makes the algorithm slightly more complex, knowledge of the delays as functions of transition time is critical for selecting the correct signal for forward propagation in the second phase of the algorithm. Also, this information can be very useful for circuit optimization since it can be used to predict the variation of the circuit timing in response to different transformations that change the transition time at a node.

```
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. Visit each node, $i$, in the graph in topological order:
   2.1. For each incoming edge $e_{ki}$ from node $k$ to node $i$ 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 $S_l=(T_l,s_l,P_l)$, where $T_l = max(T_{ki})$
   2.3. Assign the selected signal $S_l$ to node $n_i$.
```

Figure 2. Traditional timing analysis algorithm.

3.1 Circuit delay as function of transition time

Initially, we examine the dependence of the timing graph delay on the transition time at the source node s_n. We then extend our analysis to the delay from an arbitrary node in the circuit to the sink node.

Since the delay and output transition time of individual gates are a function of their input transition time, the delay of the timing graph as a whole, as well as the transition time at the sink node are a function of the transition time at the source node. Furthermore, since the delay and output transition time are continuous functions of input transition time, it is easy to prove that the delay and sink node transition time of a timing graph are continuous functions of the transition time at the source node, as stated in the following property:

Property 1: If the timing graph edges have delays and transition times that are continuous functions $d_e(s_i)$ and $s_e(s_i)$ of edge input transition times s_i, then the timing graph delay D and the sink signal transition time s_f are continuous functions $D(s_s)$ and $s_f(s_s)$ of the transition time s_s at the source node of the graph.

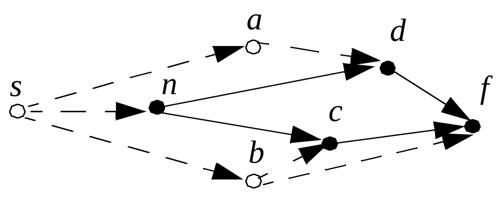
It is useful to note that in a real circuit any transition time s has a positive, non-zero lower bound s_{min} defined by the maximum speed of logic gates switching. On the other hand, the transition time is upper bounded by value s_{max} defined by the maximum reasonable delay of the circuit. From this, it is clear that both the delay and transition time functions $D_i(s_j)$ and $s_i(s_j)$ can be efficiently approximated with a finite table at any required accuracy.

We now extend our analysis to the delay from an arbitrary node in the graph to the sink node, and introduce the following useful definition of a dependence graph.

Definition 6: For a node n in timing graph $G=(N,E,n_s,n_f)$ with sink node f, we define a node n dependence graph $G_n= (N_n,E_n,n,n_f)$ as the subgraph of G such that it consists of all nodes and edges of graph G belonging to at least one path from node n to the sink node n_f.

In Figure 3, we show the dependency graph of node n for an example of timing graph. Solid edges are dependent edges of node n. We refer to the delay of a dependence graph G_n as D_n. Note that delay D_n is the delay of the slowest path from node n to the sink node f of the timing graph G. The delay of the dependence graph at n is dependent on the transition time at node n, meaning that $D_n = D_n(s_n)$. Also, it is easy to see that if the sink node has a required time t^r_f and node n has a required time t^r_n, then the delay of the dependence graph at node n can be expressed as the difference of these require times:

$$D_n = t^r_f - t^r_n. \quad \text{(EQ 1)}$$



node *n* dependency graph consists of nodes *n, c, d* and *f* and, edges (*n,d*), (*n,c*), (*d,f*) and (*c,f*)

Figure 3. Dependency graph for node *n*

In addition to the delay from a node to the sink node, we also define the delay D_e from an edge to the sink node, as follows.

Definition 7: The delay from edge $e_{ab}=(n_a,n_b)$ is the delay of the longest path starting from this edge and ending at the sink node n_f.

3.2 Backward propagation of signals

We now discuss how the delay from node *n* to the sink node n_f can be computed using backward propagation of signals. We express the delay from node *n* to node n_f recursively as a function of the delay from its fanout nodes and its fanout edges. We consider the graph fragment shown in Figure 4 where node *n* has fanout nodes *x, y,* ... . The delay $D_{nx}(s)$ from edge (*n, x*) to the sink node n_f can be expressed in terms of the delay $D_x(s)$ and the edge delay function $d_{nx}(s)$ and transition time function $s_{nx}(s)$ as follows:

$$D_{nx}(s) = D_x(s_{nx}(s)) + d_e(s) \qquad \text{(EQ 2)}$$

The delay from node *n* can now be computed as the maximum of the delays from each of its fanout edges (*n, x*), (*n, y*), ... as follows:

$$D_n(s) = max(D_{nx}(s), D_{ny}(s), ...) \qquad \text{(EQ 3)}$$

The algorithm for computing delays $D_n(s)$ from nodes *n* in timing graph $G=\{N,E,n_s,n_f\}$ as a function of the signal transition times at those nodes is shown in Figure 5. It traverses the timing graph in reverse topological order starting from the sink node n_f and computes the delays from each node *n* using the values of the delays from its fanout nodes. For simplicity, the function $D_f(s) = 0$ although any other constant value could be used. Note that in practice, the function $D_n(s)$ is represented by a linearly interpolated table and that tables for all nodes in the graph are constructed in a single backward traversal.

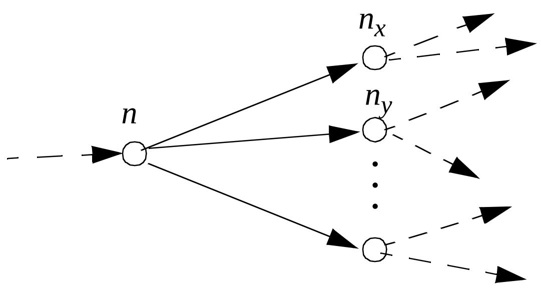


Figure 4. Fragment of a timing graph

```
1. Assign to the sink node $n_f$ the delay function $D_f(s)=0$
2. Visit each node $n$ in the graph in reverse topological order and perform the following:
   2.1. For each outgoing (fanout) edge $e_x=(n,x)$ from node $n$ to node $x$ propagate the delay $D_{nx}(s)$ from node $x$ through the edge $e_x$ using formula (EQ2).
   2.2. Compute the delay $D_n(s)$ from node $n$ to $n_f$ using formula (EQ3)
   2.3. Assign the computed delay $D_n(s)$ to node $n$
```

Figure 5. Backward computation of dependence graph delay

This algorithm computes delays from each node of the timing graph including the source node. The delay from the source node is the delay of the timing graph itself. Each delay is computed for all possible values of signal transition times. The complexity of the algorithm is $O(NM)$ where *N* is number of nodes in the timing graph $G=\{N,E,n_s,n_f\}$ and *M* is number of discretization points of delays and transition times as a function of transition time.

The above algorithm does not compute signal transition time at the sink node of the graph as a function of the transition time at its source node. We therefore extend the algorithm such that a transition time function $S_n(s)$ is computed, which expresses the transition time of the sink node n_f as a function of the transition time of a signal at node *n*. Again, we can define the edge transition time function $S_{nx}(s)$ for edge (*n, x*) as the transition time at the sink node n_f as a function of the transition time of a signal that passes through (*n, x*). Similar to EQ 2, $S_{nx}(s)$ is computed as follows:

$$S_{nx}(s) = S_x(s_{nx}(s)) \qquad \text{(EQ 4)}$$

Note that for the sink node itself, $S_f(s) = s$.

The computation of the node transition time function $S_n(s)$ is more complex and is tightly coupled to the computation of the delay function $D_n(s)$ shown in EQ 3. Both are computed simultaneously, as shown in Figure 6. The algorithm iterates through the complete range of possible transition times. For each transition time s_i, the edge delay functions $D_{nx}(s_i)$, $D_{ny}(s_i)$, ... are evaluated for each fanout edge and the maximum delay is selected. The edge

```
1. For each transition time value $s_i$ compute values of the delay from node $n$ to the sink node and transition time functions $D_n(s_i)$ by doing the following:
   1.1. From the set of edge delay functions $\{D_{nx}(s_i), D_{ny}(s_i),...\}$ select the function $D_{nx}(s_i)$ such that $D_{nx}(s_i)=max(D_{nx}(s_i), D_{ny}(s_i),...)$.
   1.2. From the set of edge transition time functions $\{S_{nx}(s_i), S_{ny}(s_i),...\}$ select the function $S_{nx}(s)$ corresponding to the same edge $(n,x)$ that the delay function $D_{nx}(s_i)$ corresponds to.
   1.3. Assign the values of the selected functions for input transition time $s_i$ to the delay and transition time functions at node $n$: $D_n(s_i)=D_{nx}(s_i)$ and $S_n(s_i)=S_{nx}(s_i)$.
```

Figure 6. Computation of signal slew at a node.

transition time functions $S_{nx}(s_i)$, $S_{ny}(s_i)$, ... are also evaluated for all fanout edges with input transition time s_i. Since the transition time of the sink node is determined by the latest arriving signal, the algorithm selects the transition time of the fanout edge, corresponding to the selected maximum delay. In this manner, the delay from node n to the sink node n_f and its associated transition time are computed for each value of s_i, until the complete delay function $D_n(s_i)$ and slope function $S_n(s_i)$ are defined for node n.

One of the results of this algorithm is the transition time function of the source node $S_s(s)$. This function gives values of the transition time at the sink node as a function of the transition time at the source node of the timing graph. Hence, if only the delay and transition time of the sink node are required in the static timing analysis, the proposed backward propagation of signal is sufficient for its computation. However, for many optimization tasks, such as gate sizing and logic optimization, it is advantageous to also compute the critical path and the slack at each node. We now show how these can be computed using an additional forward propagation of signals.

3.3 Arrival time, slack and critical path computation

To compute the arrival time, slack and critical path for each node of the timing graph, we propagate signals in topological order through the circuit. We first consider a single propagated signal. During its forward propagation, we compute its arrival time T_n and transition time s_n at node n. During the backward traversal, described in the previous section, we obtained the delay function $D_n(s)$ representing the maximum delay from node n to the sink node n_f, as a function of the transition time at node n. Using the transition time s_n of a forward propagated signal at node n, it is therefore possible to compute the required time $t^r{}_n$ for this signal, using EQ 1 as follows:

$$t^r{}_n = t^r{}_f - D_n(s_n) \qquad \text{(EQ 5)}$$

Using this required time, it is now possible to compute the slack Sl_n of the propagated signal by subtracting its signal arrival time from its signal required time:

$$Sl_n = t^r{}_n - T_n \qquad \text{(EQ 6)}$$

In the same way, it is possible to compute the slack of all signals that are propagated to node n. Since for each signal we can compute its slack, we determine which signal will result in the latest arrival time at the sink node (and hence the largest overall circuit delay from) by selecting the signal with the most critical slack. Note that a smaller or more negative slack value corresponds to a higher criticality of a signal. Using the slack of the signals at node n, it is possible to predict exactly which signal will be critical and to only propagate this signal in the forward traversal. In this manner, the arrival time, slack and critical path of all nodes in the circuit can be computed, as shown in Figure 7. Note the algorithm is identical to that of traditional timing analysis shown in Figure 2, except that the pruning criteria for selecting the propagated signal has be modified to use the backward propagated delay functions.

Using the proposed slack based timing analysis, the critical signal at each node is selected for propagation, and hence the inaccuracies of the traditional timing algorithm are eliminated. Also, the discontinuities of circuit delay with gate delays that occur in this algorithm are eliminated making the proposed STA highly suitable for circuit optimization applications.

```
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. Visit each node, $i$, in the graph in topological order:
   2.1. For each incoming edge $e_{ki}$ from node $k$ to node $i$ 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. For signal $T_{ki}$ compute the slack $Sl_{ki} = t^r{}_f - D_i(s_{ki}) - T_{ki}$,
        where $D_i(s_i)$ is the delay function at node $i$ computed in
        the backward traversal
   2.3. From all signals $S_{ki}$ select signal $S_l=(T_l,s_l,P_l)$, where
        $Sl_{li}= min(Sl_{ki})$
   2.3. Assign the selected signal $S_l$ to node $n_i$.
```

Figure 7. Slack based timing analysis algorithm.

In the forward traversal of the proposed algorithm only a single arrival time is propagated at each node and hence the linear run time of STA is preserved. The overall run time is dominated by the backward traversal, which has a complexity $O(NM)$ where N is the number of nodes in the timing graph and M is the number of discretization points of the delay functions. Note however that only a single backward traversal is performed and hence, the traversal overhead is incurred only once, regardless of the discretization size. Similarly, a number of other operations, such as the computation of the output loading condition are shared for all discretization points. Therefore, it was found that in practice, the run time overhead due to the backward traversal of the proposed approach was relatively minor.

Also, the number of discretization M represents a run time/accuracy trade-off. As the number of discretizations is reduced, the run time reduces, but the accuracy of the delay functions reduces due to interpolation error. Hence, if insufficient discretization points are used, it is possible that the wrong signal is selected for propagation in the forward traversal due to the approximation of the delay functions. However, it can be easily shown that the error in the computed delay of the circuit is bounded by the error in the delay function representation for each level of the circuit. In practice, as shown in the next Section, a discretization of 7 points was sufficient to obtain an exact match between the proposed algorithm and exhaustive timing analysis.

4 Experimental Results

The proposed timing analysis algorithm was implemented and tested on the MCNC benchmark circuits [13]. For each tested circuit, three timing analysis approaches were compared: the traditional timing analysis approach using arrival time based pruning, the proposed timing analysis approach using slack based pruning, and an exhaustive timing analysis approach which traverses all paths in the circuit using a depth-first-search. The last approach has a very high run time and was implemented only to provide a comparison with an “exact” approach. For the proposed slack based STA approach, the delay functions were represented using 7 piecewise linear discretizations.

The benchmark circuits were synthesized using a commercial 0.18μm technology library. The timing analysis algorithms operated on the delay characterization tables provided with the library. Arrival times at the circuit inputs were chosen such that the impact of slope propagation on the circuit delay was observable, thereby providing a more stringent test of the correctness of the proposed algorithm. Out of 10 test circuits, it was found that 9 show a difference between the arrival time based STA and the exact STA. The results for these circuits are shown in Table 1. For each circuit, the exact, arrival time based, and the proposed slack based STA timing results are shown in columns 5, 6, and 7, respectively. The run time for arrival time STA and slack based STA are shown in columns 8 and 9, respectively. Note that for all circuits, the timing analysis results of the proposed backward STA approach matched that of the exact STA approach. Also, the run time of the proposed STA approach is increased over that of the traditional STA by only 33% on average.

5 Conclusions

In this paper, we have presented a new STA approach which uses slack based pruning of propagated signals and thereby removes the inaccuracies present in the traditional STA approach. Since only a single arrival time is propagated for each node in the circuit, the linear run time of STA is preserved. We demonstrate that the proposed approach computes circuit delays that match those computed with an exact STA algorithm, while incurring only a modest run time increase over the traditional STA approach. Since the proposed method removes the discontinuities in circuit delay present in traditional STA, the approach is particularly useful for circuit optimization methods.

Acknowledgements

The authors would like to thank Youngmin Kim for his work and help. The work has been supported by NSF, SRC, GSRC/DARPA, IBM, and Intel.

References

[1] Hitchcock, R.B., “Timing verification and the Timing Analysis program”, Proc., IEEE/ACM Design Automation Conference, 1982, pp.594-604.

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

[3] S.Devadas, K.Keutzer, S.Malik, “Computation of Floating Mode Delay in Combinational Circuit: Theory and Algorithms”, IEEE Trans. on Computer Aided Design, Vol. 12, No. 12, pp. 1913-1923, Dec. 1993.

[4] D. H. C. Du, S. H. C. Yen, S. Ghanta, “On the general False path problem in timing analysis”, Proc., IEEE/ACM Design Automation Conference, 1989, pp. 555-560.

[5] 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, pp. 302-309, May 1993.

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

[7] K. Shepard, V. Narayanan, “Noise in Deep Submicron Digital Design,” Proc. ICCAD 1996, pp. 524-531.

[8] S. Sapatnekar, “Capturing the Effect of Crosstalk on Delay,” Proc. VLSI Design 2000, pp. 364-369, January 2000.

[9] G. Bai, S. Bobba, I. Hajj, “Static timing analysis including power supply noise effect on propagation delay in VLSI circuits,” Proc. DAC, 2001, pp. 295 -300.

[10] D. Blaauw, V. Zolotov, and S. Sundareswaran, “Slope Propagation in Static Timing Analysis,” IEEE Transactions on Computer-Aided Design, Vol. 21, No. 10, pp. 1180-1195, October 2002.

[11] Chandu Visweswariah, Andrew R.Conn, “Formulation of Static Circuit Optimization with Reduced Size, Degeneracy and Redundancy by Timing Graph Manipulation”, ICCAD, 1999, pp.244-251.

[12] J.F. Lee, D.L. Ostapko, J.Soreff, C.K. Wong, “On Signal bounding problem in timing analysis”, ICCAD 2001, pp. 507- 514.

[13] http://www.cbl.ncsu.edu.

Table 1. Timing and run time comparisons

Circuit				STA timing results (ns)			Run time (msec)	
Name	Gates	Inputs	Max. level	Exact	Arrival time based (difference % vs. exact)	Proposed slack based (difference % vs. exact)	Arrival time based	Proposed slack based (increased % vs. arrival time based)
i1	41	25	5	4.16	3.95 (5.1%)	4.16 (0%)	2	3 (50%)
i2	189	201	10	4.18	3.87 (7.4%)	4.18 (0%)	13	16 (23%)
i3	120	132	6	2.34	2.34 (0%)	2.34 (0%)	9	11 (22%)
i4	160	192	12	4.39	4.33 (1.4%)	4.39 (0%)	12	14 (17%)
i5	198	133	18	5.84	5.77 (1.3%)	5.84 (0%)	11	14 (27%)
i6	390	138	6	6.73	6.63 (1.5%)	6.73 (0%)	18	24 (33%)
i7	510	199	7	6.75	6.66 (1.3%)	6.75 (0%)	24	33 (38%)
i8	749	133	13	7.26	6.38 (12.0%)	7.26 (0%)	32	46 (44%)
i9	473	88	12	4.44	4.31 (2.8%)	4.44 (0%)	19	29 (53%)
i10	1912	257	50	12.5	11.6 (7.5%)	12.5 (0%)	89	109 (22%)