

Using Temporal and Functional Information in Crosstalk Aware Static Timing Analysis

TONG XIAO^{a,*} and MALGORZATA MAREK-SADOWSKA^{b,†}

^a*Sun Microsystems, Inc., 410 North Mary Avenue, USUN02-203, Sunnyvale, CA 94085, USA;* ^b*Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106, USA*

(Received 15 March 2001; Revised 30 January 2002)

Crosstalk-induced delay in deep sub-micron digital circuits can be quite significant and difficult to determine because of dependency on neighboring signals. In this paper we study the problem of incorporating temporal and functional information to improve the accuracy of crosstalk aware static timing analysis. We propose an efficient method to compute a signal's earliest and latest arrival times when timing windows and slew rate ranges are known for its inputs and its coupling neighbors' inputs. We show that iteratively updating timing windows is necessary when signals on the same path are mutually coupled. The accuracy of static timing analysis can be further improved by our functional correlation analysis. The proposed techniques have been applied in crosstalk aware static timing analysis, which can guide timing-driven layout synthesis and quick timing verification in deep submicron technologies. Experimental results demonstrate that the proposed methods significantly reduce the pessimism in predicting circuit performance.

Keywords: Capacitive coupling; Crosstalk; Delay; Timing analysis; Timing window; Functional correlation

INTRODUCTION

With aggressive technology scaling, wire aspect ratios increases and interconnect spacing decreases. Thus, in deep sub-micron circuits, coupling capacitance between interconnects may be comparable to interconnect intrinsic capacitance. This makes delay of a switching signal depend on switching times and switching directions of neighboring signals. Our simulation results show that, compared to the case when coupling neighbors are quiet, delay of a single stage may be speed up by 50% when coupling neighbors are switching in the same direction, and may be slowed down by 90% when coupling neighbors are switching in opposite directions in 0.35 μm technology. Delay of a circuit is determined by the longest path from primary inputs to primary outputs. A long path may have a large number of coupling neighbors and the interdependencies among them are

difficult to determine. If delay of a stage and a path are computed based on the worst-case estimation without analyzing the temporal and functional information of coupling neighbors, crosstalk aware static timers may give very loose bounds on circuit performance, which will result in unnecessary design effort and waste of chip resources to meet timing specifications.

Static timing analyzers are widely used to quickly estimate a circuit's timing behavior at various stages of a design. More accurate timing estimation is feasible by incorporating functional or sensitization information. The papers complementing or improving static timing analysis include false path identification [3–5] and data dependent delay analysis [29]. Recently, several papers have been published [1,10,25,27] addressing various aspects of static timing analysis in the presence of crosstalk delay effects. However, most methods basically follow (though perhaps implicitly) the assumption in traditional static timing

*Corresponding author. Tel.: +1-408-524-6413. Fax: +1-408-733-6587. E-mail: xiaotong@ieee.org

†E-mail: mms@ece.ucsb.edu

analysis, i.e. only the latest arrival time at a signal is propagated [2]. Thus, some methods are too pessimistic and some are overly optimistic when estimating circuit delay. Moreover, no efficient methods have been proposed to analyze the functional dependency of coupling neighbors to improve the accuracy of crosstalk aware static timers.

Our goal in this paper is to decrease pessimism while maintaining efficiency in crosstalk aware static timing analysis, so it may be applied in fast timing verification and crosstalk aware layout synthesis.

We begin our discussion with computing delay through a single logic stage. Next, we describe an efficient method to estimate a signal's timing window in presence of coupling. We explain how to compute a signal's earliest and latest arrival times when timing windows and slew rate ranges are known for its inputs and its coupling neighbors' inputs. We model coupling delay effects as worst-case speedup or slowdown in an *effective skew window*. We show how to find the effective skew window by analyzing the simple worst case when coupling neighbors can switch at any time. After obtaining the effective skew window, we make a pessimistic assumption that maximum slowdown occurs in the entire effective skew window, and propose an efficient method to estimate the earliest and latest arrival time.

We then show that iterative topological analysis is necessary to compute timing windows of all signals in the circuit when signals on the same path are coupled to each other. The uncertainty of delay estimation can be decreased by performing iterations based on two different assumptions: simple worst assumption and nominal case assumption. In the simple worst case we first assume all coupling neighbors cause maximum coupling delay and then iteratively reduce the pessimism in delay estimation. In the nominal case assumption we first assume there is no coupling effect from coupling neighbors, then iteratively reduce the optimism of delay estimation.

Finally functional correlations are analyzed based on path sensitization criteria. Path delay can be more accurately estimated based on the correlation information. Thus there are many fewer critical paths.

The techniques proposed in this paper are efficient and suitable for fast timing verification. They can be used to guide timing-driven layout synthesis and optimization. As technology scaling causes more severe coupling effects, we believe that the proposed techniques can be applied in current and new technologies.

The rest of the paper is organized as follows: in the second section we give an overview of related work. In the third section we explain concepts of delay calculation in the presence of crosstalk. In the fourth section we discuss the problem of computing a signal's timing window and propose an efficient method to determine it. In the fifth section we explain the basics of crosstalk aware static timing analysis and show how the proposed timing window estimation is used in this procedure. In the sixth section we propose an efficient functional correlation

analysis to improve accuracy in critical path delay estimation and critical path identification. Experimental data in the seventh section demonstrate the effectiveness and efficiency of our crosstalk aware static timing analysis integrating temporal and functional information. The eighth section concludes the paper.

PREVIOUS WORK

Recently, several papers addressing the problem of computing worst-case delay in presence of crosstalk-delay effect have been published. They fall into two categories:

1. Compute the worst-case coupling delay without timing information of coupling neighbors:

The Elmore delay model is widely used in static timing analysis and optimization. It cannot be applied directly when coupling capacitance is present. In Ref. [22] it has been proposed that coupling capacitance C_c is replaced with effective capacitance-to-ground, which is 0, C_c , or $2C_c$ for different switching conditions, but this simple approximation doesn't provide upper bounds on delays for certain combinations of input rise times and skews [33].

Gate delay look-up tables are used as an efficient way to estimate delay, but they are inherently incompatible with RC(L) interconnects. Lookup tables usually only list cell delays for different load capacitances. When wire resistances are not negligible, effective capacitance needs to be computed [23]. If coupling capacitance is significant, finding effective capacitance corresponding to worst case coupling delay requires several Newton–Raphson iterations, and its complexity is greater [7]. A lookup table considering coupling has been proposed in Ref. [33], but the number of characterized coupling structures is limited. Coupling delay depends not only on the coupling length, but also on aggressors' and victim's driver strength. It seems there are too many combinations of variables to characterize.

Waveform iteration method [14] can be used to find a victim's worst-case delay when aggressors can switch at any time. However, there is no discussion on how to apply the proposed approach when input timing windows are restricted rather than of arbitrary duration, and when inputs have variable slew rates.

2. Compute coupling delay with partial timing information:

In traditional static timing analysis, it is assumed that only the latest arrival time at each signal may result in the longest path delay, and the latest signal is propagated forward. This is referred to as latest propagation algorithm in Ref. [2]. This assumption makes computing the earliest and latest arrival times of a signal a straightforward task in traditional static timing analysis. However, this is not a valid assumption in the presence of coupling. This assumption has been adopted in various published works when computing the latest arrival time of a signal, and may result in optimistic or pessimistic estimation.

The authors of Ref. [1] improved waveform iteration method [14] by checking whether an aggressor is temporally isolated from a victim when the victim switches at its latest time. This may result in optimistic estimation of the latest arrival time.

A relative window method is used in Ref. [26] to estimate the coupling delay. The relative window is the arrival time difference of victim and aggressor inputs. In our paper this arrival time difference is referred to as a skew. In Ref. [26], a lookup table of delay degradation (slowdown) vs. relative window is obtained for each victim and aggressor pair. Later this information is used to find worst delay degradation considering timing windows of victim and aggressor inputs. The case of multiple aggressors is discussed in Ref. [27]. Building a lookup table for each victim and aggressor pair uses a large amount of run time and consumes excessive memory.

In Ref. [10], multiple aggressor alignment is proposed to find the worst-case delay on a victim when timing windows of aggressors are known. The methods in Refs. [27,10] may be very pessimistic in estimating the latest arrival time.

In contrast to the above two groups of methods, in Ref. [24], timing windows of all signals are first computed by assuming all adjacent wires are quiet. Then coupling effects are computed by exhaustively checking all possible combinations of victim and aggressor input switching times. This process is expensive when the timing window is a large interval, and when there are multiple coupling neighbors.

In Ref. [25], a nice iterative procedure for capturing the dependency of coupling delay on neighbors' switching times has been proposed and incorporated into the inner loop of a timing optimizer. The procedure is based on the assumption that maximum slowdown occurs when output waveforms switch at the same time in opposite directions. This assumption may result in underestimation of coupling delay, as significant slowdown occurs not only when victim's and aggressor's outputs switch at the same time, but also when the aggressor is earlier or later by up to 400 ps for typical situations in 0.35 μm technology.

Traditional static timing analyzers visit the circuit's nodes in topological order [8]. Without coupling, combinational circuits can be represented as acyclic graphs. Any topological order can be used to compute arrival times of all signals in the circuit in one sweep and in linear time, as long as all inputs of a signal are computed before it is processed. However, similar conditions may not be satisfied in the presence of interconnect coupling; thus iterative updating timing windows may be necessary. The condition of iteration has not been discussed before.

Besides the temporal information, functional isolation has also been explored to reduce pessimism of coupling delay analysis. Two signals are functionally isolated if they can never switch simultaneously or if the switching effect cannot be observed at the circuit's primary outputs [16]. In Ref. [16], BDD based approach is proposed to

identify pairs of signals in a circuit which are functionally isolated. For such signals, there is no need to consider slowdown (or speedup) effects between them. These data are helpful in constraint-driven layout syntheses. Satisfiability formulation [9] is proposed to decide the functional correlation between neighbors of a wire and to compute crosstalk noise when delay of each gate is a fixed number. We believe this formulation is suitable only for small circuits, as the satisfiability problem is NP-complete [13]. The above functional analysis methods are not applicable to static timing analysis because of complexity issues.

COMPUTING STAGE DELAY IN PRESENCE OF CROSSTALK

In this section, we explain techniques used to compute delay of a stage when exact switching times and exact slew rates are known for the stage's inputs. We consider a general network, where a net may be coupled to more than one line, as shown in Fig. 1.

Solving differential equations using numerical methods as it is done in SPICE can provide accurate results at the expense of a long run time. It is usually applied to the last stages of the design process, and sometimes only to the critical parts of the circuit which have been identified by static analysis or estimation. Moment-based methods [21] greatly increase the simulation speed by performing transform domain analysis. With proper gate modeling [6], transform domain analysis can provide good accuracy compared to SPICE simulation. SPICE simulation and general moment matching methods are still too expensive to use in fast timing verification and timing driven layout synthesis.

In Ref. [32], we proposed a simpler yet still efficient way to find an approximate transfer function for RC circuits, and we used it to compute bounds of delay in the presence of crosstalk. We briefly explain that approach below.

We replace each driving gate in Fig. 1 with its equivalent resistance, each receiver gate with input capacitance, and each wire segment by a π model. The delay from START NODE to END NODE can be computed by solving the following Eqs. (1)–(8).

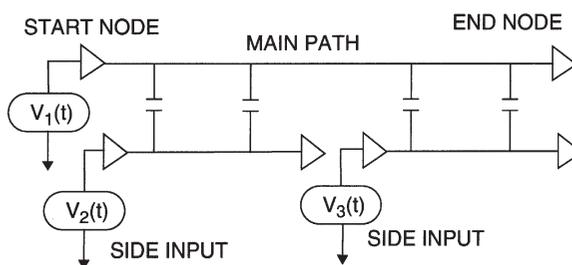


FIGURE 1 Illustration of the MAIN PATH.

The s -domain output voltage at the END NODE can be expressed as:

$$V_o(s) = \sum_{i=1}^N H_i(s)V_i(s) \quad (1)$$

where $H_i(s)$ is the transfer function from the input i to the END NODE when all other inputs are held at 0 and where $V_i(s)$ is a voltage in transform domain at input i . We approximate each transfer function using expressions in the following form:

$$H_i(s) = \frac{a_{0i} + a_{1i}s}{1 + b_1s + b_2s^2} \quad (2)$$

The coefficients are computed from the formulae [20]:

$$b_1 = \sum_{C_i \in C} R_{ii}^0 C_i, b_2 = \sum_{C_i, C_j \in C, j < i} R_{ii}^0 C_i R_{jj}^i C_j, \quad (3)$$

R_{ii}^0 is the resistor seen from two ends of C_i while keeping all other capacitors open. It is computed by a tree traversal. R_{jj}^i is a resistor seen from two ends of C_j when C_i is shortened and all other capacitors are open.

If input i is at the START NODE, a_0 and a_1 can be computed as follows:

$$a_{0i} = 1 \quad (4)$$

$$a_{1i} = a_{0i}b_1 - \sum R_j(C_j + X_j) \quad (5)$$

For each node j on the MAIN PATH, R_j is the driving resistor seen from node j , X_j is coupling capacitance connected at node j .

If input i is a SIDE INPUT, a_0 and a_1 can be computed as follows:

$$a_{0i} = 1 \quad (6)$$

$$a_{1i} = a_{0i}b_1 - \sum R_k X_k \quad (7)$$

For each node k from i th SIDE INPUT, R_k is the driving resistor seen from node k . X_k is coupling capacitance connected at node k .

The time domain expression $v_o(t)$ of the output at the END NODE can be obtained by performing explicit inverse Laplace transformation. Analytical expression for poles and residues are given in Ref. [32]. If we want to compute delay at any point d on the waveform of the END NODE (for example, $d = 0.5$ for 50% point delay) it can be computed by solving the following equation:

$$V_o(t_d) = d \times V_{DD} \quad (8)$$

One Newton–Raphson iteration written in analytical form can provide a good approximation. Equations (1)–(7) above give a way to compute waveform at any node in the circuit, and can also be used to compute coupling

noise. Similar and even simpler formulae have been derived to estimate peak crosstalk noise in Refs. [30,31].

In delay calculation considering crosstalk, more waveform information is needed than before. We need not only delay at 50% and other points of output, but also peak noise and noise arrival time. In this paper, we will use the above delay calculation method as an example. However, other delay calculation that offers all the necessary information can also be used, including simulation based methods.

WORST ARRIVAL TIME ESTIMATION

Delay calculation as explained in the last section can predict delay of a logic stage when exact switching time are known for the victim's input and the aggressors' inputs. However, it cannot be directly applied to find the shortest and longest delay paths in a circuit. To solve this problem, in this section, we consider the problem of computing the earliest and latest arrival times of each signal, which is also referred to as the timing window. The fastest and slowest slew rates of each signal are referred to as the slew rate ranges. In the rest of the paper, we use the following symbols:

T : signal arrival time, T^V : arrival time of a victim's input, T^A : arrival time of an aggressor's input;

S : signal slew rate, S^V : slew rate of a victim's input, S^A : slew rate of an aggressor's input.

Multiple aggressors worst arrival time (MAWAT) problem formulation:

Given the victim's input timing window $[T_1^V, T_2^V]$, and the input slew rate range $[S_1^V, S_2^V]$; the aggressors' input timing windows $[T_{i1}^A, T_{i2}^A]$, and the input slew rate ranges $[S_{i1}^A, S_{i2}^A]$, $i = 1, 2, \dots, n$, where n is the number of aggressors, compute the victim output's timing window.

In traditional static timing analysis, we can estimate the minimum and maximum stage delay t_{\min} , and t_{\max} . t_{\min} is the delay when the fastest input pin switches at slew S_1 . t_{\max} is the delay when the slowest input pin switches at slew S_2 . We assume that the timing window at the gate's input is $[T_1, T_2]$. The timing window at the gate's output is computed as $[T_1 + t_{\min}, T_2 + t_{\max}]$. Computing the earliest and the latest arrival times at a gate's output is complicated when the gate's driven interconnect is coupled to other interconnects. Computing output's latest arrival time by simply adding the worst-case coupling delay to T_2 may yield very pessimistic estimation.

In the following section, we will only discuss how to find the latest arrival time considering coupling. Similar analysis is applicable to finding the earliest arrival time considering coupling. We assume that the slew rate ranges are known on the primary inputs. The slew rates at each signal are estimated by assuming the worst speedup and the worst slowdown effects from the coupling neighbors. We also assume that the recorded slew rate ranges may be associated with both the earliest and the latest arrival time

at each signal. First some experimental data will be shown to motivate the problem of the worst arrival time estimation, some common misconceptions will also be clarified. We will discuss how to estimate the worst delay in a simple coupling structure, and then extend our solution to the general case.

Coupling Effects: Some Experimental Data

First consider the case when an interconnect has only one neighbor, as shown in Fig. 2. When we are computing delay at the node Out1 of wire1, we call the neighbor wire2 an *aggressor*, and wire1 a *victim*. Note that the victim and the aggressor attributes are interchangeable, i.e. wire2 is also a victim of an aggressor wire1. The difference of switching times between the aggressor and the victim is referred to as a *skew*. Different skews we will be considering are shown in Fig. 3. The *input skew* is a difference of switching time at 50% point of the victim's and the aggressor's input waveforms. The *starting point skew* is a difference between the switching starting points of the victim's and the aggressor's input waveforms. The *output skew* is a difference at 50% point of the victim's output (Out1 in Fig. 2) and the aggressor's output (Out2 in Fig. 2) waveform.

When the aggressor and the victim switch in the opposite directions, the victim may be slowed down compared to the case when the aggressor is quiet (nominal case). When the aggressor and the victim switch in the same direction, the victim may be sped up. The rate of slowdown or speedup at the victim's output depends on the skew and on the slew rates of the victim's and the aggressor's inputs.

To illustrate the impact of input skew and slew rates on the delay, typical simulation data are shown in Fig. 4 and in Table I. Two wires which are 1 mm long and run parallel with minimum spacing are driven by the minimum sized drivers. The SPICE level 49 transistor model for 0.35 μm technology is used. The simulation data are obtained by sweeping the starting point skew of the victim's and of the aggressor's inputs from -4 ns to +4 ns, with a step of 100 ps. Figure 4 shows how delay at the victim Out1 changes when the input skew changes. The victim's input time constant is 400 ps, and the aggressor's input time constant is 40 ps. A negative skew means that the aggressor's input switches earlier than the victim's input.

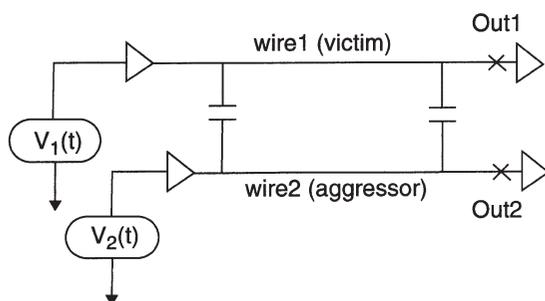


FIGURE 2 Two coupling interconnects.

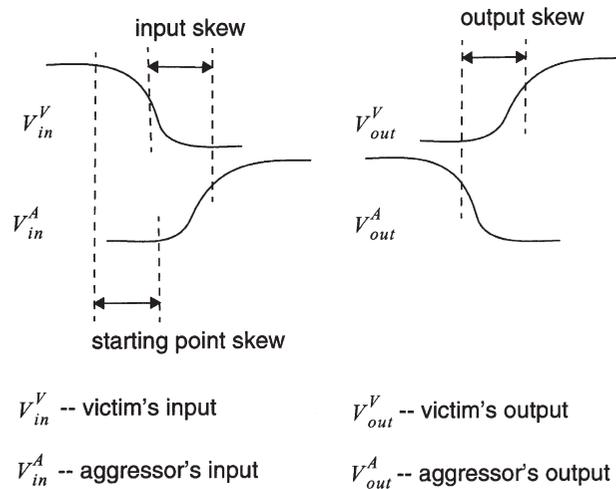


FIGURE 3 Definition of different skews.

As can be seen in Fig. 4, the worst case slowdown at the victim's output occurs only at a certain skew, but significant slowdown occurs within a large range of skews. In this case, the maximum slowdown occurs when the aggressor's input switches later than the victim's input by 454 ps. When the aggressor has a skew between 54.4 and 554 ps, the slowdown at the victim is at least 70% of the maximum slowdown.

Table I shows typical data when the victim's and the aggressor's inputs have different slew rates. The units of delay are ps. Nominal delay is the delay at Out1 when its neighbor is quiet. The data in each column are obtained as follows. For a given slew rate of the aggressor's and the victim's inputs, we sweep starting point skew from -4 to 4 ns, with a step of 100 ps, and find the condition when maximum delay at Out1 occurs. The starting point skew, input skew and output skew found in this case are referred to as the *worst starting point skew*, *worst input skew* and *worst output skew*, respectively. When the input skew is too large or too small, the aggressor's waveform is far away from the victim's waveform, so it does not affect the victim. We refer to the range of input skews that cause at

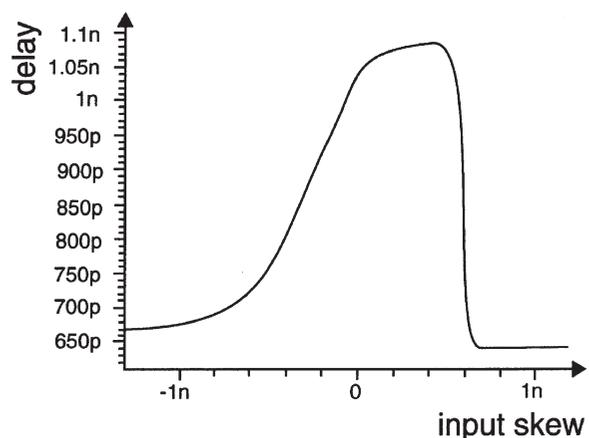


FIGURE 4 The impact of input skew on delay of Out1.

TABLE I The impact of slew rates on coupling delay

Victim's input slew	40	40	400	400
Aggressor's input slew	40	400	400	40
Out1 nominal delay	399	399	641	641
Out1 max. slowdown	395	360	427	441
Worst starting point skew	100	-300	200	700
Worst input skew	103	-54.4	200	454
Worst output skew	-362	-292	-366	-304
Effective skew window	[-1100, 300]	[-1254, 348]	[-1100, 500]	[-945, 554]

least 5% slowdown at the node Out1 as the *effective skew window*. For example, in Fig. 4, the effective skew window is [-954 ps, 554 ps].

Table I shows that for a given slew rate of the victim's input, a faster aggressor causes a larger worst-case slowdown. The maximum delay at the victim's Out1 occurs when the victim is the slowest and the aggressor is the fastest.

The bounds of the effective skew window depend on the slew rate of the victim's and aggressor's input. However, the lower bound occurs in the case when the victim is the fastest, and the aggressor is the slowest. The upper bound occurs in the case when the victim is the slowest and the aggressor is the fastest.

The experimental data in Table I clarify some misconceptions.

Misconception 1 The worst-case delay occurs when the inputs switch at the same time.

As we can see in Table I, this is not the case even when two interconnects are symmetric, i.e. have the same lengths, their input drivers are of the same size, and they switch at the same slew rate. The reason is that the rising and falling delays can be different, so even for the inputs switching at the same time, the waveforms at the output Out1 and Out2 may be far away and not affecting each other.

Misconception 2 Worst case delay occurs when outputs switch at the same time.

Referring to Table I's *worst output skew*, we note that the worst-case delay occurs when aggressor's Out2 waveform arrives earlier than the victim's Out1 waveform.

Latest Arrival Time Estimation for a simple Case

Single aggressor latest arrival time (SALAT) problem: Consider a victim and its neighbor as shown in Fig. 2. Given the victim's input timing window $[T_1^V, T_2^V]$ and its slew rate range $[S_1^V, S_2^V]$, the aggressor's input timing window $[T_1^A, T_2^A]$ and its slew rate range $[S_1^A, S_2^A]$ compute the latest arrival time T_{latest} at the victim's output Out1.

We will first discuss certain special cases of the SALAT problem. In Case 1 and Case 2 to be discussed below, the aggressor may switch at any time. In these cases we need only to find the worst case stage delay in order to directly compute the latest arrival time. If the aggressor's input

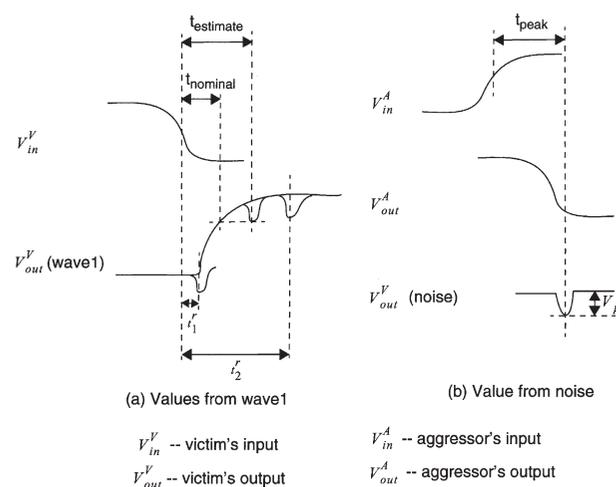
switching time is bounded by the timing window, we determine first when the coupling effects may occur and, using this information, we estimate the latest arrival time. Solution to Case 1 is the foundation for the more complicated cases, as it provides the key parameters like the effective skew window and the worst-case slowdown. These parameters will be used to solve the general SALAT problem.

Case 1 Fixed slew rate: The victim's input switches at T_1^V at a slew S_1^V , the aggressor's input arrival time is unbounded, and the aggressor's input slew is S_1^A , compute T_{latest} .

We solve this case using the procedure shown in Fig. 6, which is inspired by Ref. [14]. The procedure determines the effective skew window, the worst possible slowdown from an aggressor, and T_{latest} . In Fig. 5 we show an example of a falling victim's input and a rising aggressor's input. The outputs have inverted phases, and we measure the delay at 50% waveform points. The procedure below, even though it is stated for a particular switching condition, is applicable to all other cases.

Several aspects in the above procedure require discussion:

1. The above procedure is a simplified version of the waveform iteration method in Ref. [14]. Here we have ignored the impact of the victim on the aggressor. Our experiments show that t_{worst} computed using this

FIGURE 5 Get data for *simple_worst_delay*.

Procedure $simple_worst_delay(S_1^V, S_2^V)$

(1) Compute nominal waveform:

Input: aggressor's input is grounded, and victim's input falls;
 Output: compute the victim's output Out1 waveform $wave1$, as shown in Fig. 5(a);
 Find the following parameters using equations (1) to (8):

- $t_{nominal}$: delay from victim's input 50% point to victim's output 50% point;
- t_1^r : delay from victim's input 50% point to the time victim's output begins to switch;
- t_2^r : delay from victim's input 50% point to the time victim's output stabilizes.

(2) Compute noise:

Input: aggressor's input rises, and victim's input is grounded;
 Output: compute noise waveform on victim's output, as shown in Fig. 5(b);
 Find the following parameters using equations (1) to (8):

- V_p : peak noise, t_{peak} : delay from aggressor's input 50% point to the time peak noise occurs.

(3) Find the worst input skew by aligning peak noise:

From $wave1$ computed in step (1), find the following parameters:

- $t_{estimate}$: the time when victim's output nominal waveform reaches $0.5 * V_{DD} + V_p$;
 this is the time when peak noise should arrive to cause maximum slowdown, as proved in [14].
 $t_{estimate}$ is illustrated in Fig. 5(a).

- w_{worst} : the worst input skew causing the maximum slowdown at the victim;
 as peak noise should arrive at $t_{estimate}$, and its delay is t_{peak} ,
 so aggressor's input should switch at $(t_{estimate} - t_{peak})$.

(4) Compute the worst delay at victim's output:

Input: victim's input falls, aggressor's input rises later than victim's input by w_{worst} ;
 Compute t_{worst} : delay from the victim's input 50% point to the victim's output Out1 50% point;

(5) Compute the latest arrival time at victim's output:

$$T_{latest} = T_1^V + t_{worst};$$

(6) Effective skew window: $[w_1, w_2]$ is estimated to be:

$$[t_1^r - t_{peak}, t_2^r - t_{peak}], \text{ worst slowdown } t_{slowdown} \text{ is } t_{worst} - t_{nominal}.$$

FIGURE 6 Procedure to compute $simple_worst_delay$.

simplified approach has less than 3% error compared to that of exhaustive simulation;

2. $t_{estimate}$ is the estimated delay based on peak noise. Our experiments show that $t_{estimate}$ may be 16% less than the actual worst case delay found by exhaustive simulation;
3. It is important to set the quiet signal correctly. In step (1), if the aggressor's input is held at V_{DD} instead of at ground, delay on the victim may be smaller by about 5%. In step (2), if the victim's input is held at V_{DD} instead of at ground, the peak noise may be smaller by 60%. This will result in an underestimate of the worst-case coupling delay. The observed discrepancy can be attributed to the difference of driving strengths of p- and n-transistor.
4. The effective skew window found by this procedure is an approximation of the one found by simulation. Peak noise arriving between t_1^r and t_2^r will affect the victim's waveform. Because the noise pulse has a non-zero width, even if the peak noise arrives a little earlier, or a little later, we can still see a small slowdown caused by the aggressor. In practice, it is sufficient to use 20% switching time as t_1^r , and 80% switching time as t_2^r .

Case2 Variable slew rates: The victim's input switches at T_2^V at the slew rate between $[S_1^V, S_2^V]$, the aggressor's slew rate is between $[S_1^A, S_2^A]$ and its switching time is unbounded, compute T_{latest} .

This is an easy extension of Case1. The maximum slowdown occurs when the victim is slowest, and the aggressor is fastest. We can use the procedure $simple_worst_delay(S_2^V, S_1^A)$ to find the latest arrival time for this case.

Case3 Given timing windows, and fixed slew rates: The victim's input timing window is $[T_1^V, T_2^V]$ and its slew rate is S_1^V ; the aggressor's input timing window is $[T_1^A, T_2^A]$, and its slew rate is S_1^A , compute T_{latest} .

First we call the procedure $simple_worst_delay(S_1^V, S_1^A)$ to compute the nominal delay $t_{nominal}$, the worst-case delay t_{worst} , and the worst slowdown t_{down} ; the worst input skew w_{worst} , and effective skew window $[w_1, w_2]$.

The most pessimistic estimation of T_{latest} can be obtained by simply adding T_2^V and t_{worst} . The most accurate T_{latest} can be determined by exhaustively checking all possible combinations of the victim's input and the aggressor's input switching times. Here we will present a method which offers a good trade-off between the accuracy and the computation cost. We will check

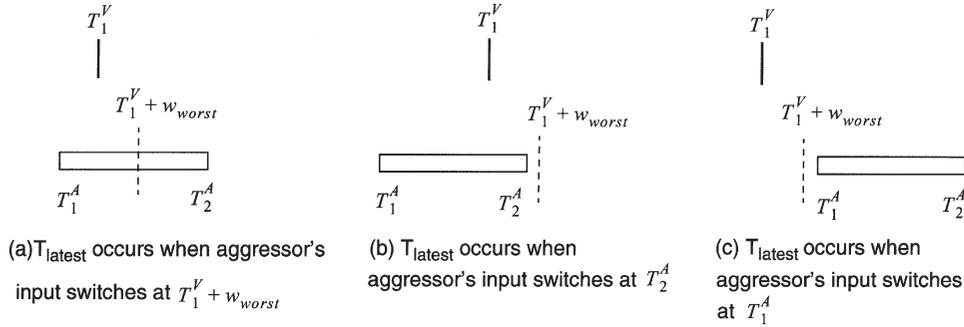


FIGURE 7 Illustration to Lemma 1.

the victim's and the aggressor's input timing windows to identify the conditions when there is a need to consider coupling effects.

For some special cases, i.e. when either the victim's input or the aggressor's input switches only at one specific time point, we have the following Lemma 1 and Lemma 2 to find the condition when T_{latest} occurs. The Lemmas can be proved with the help of Fig. 4. For example, w_{worst} is the input skew causing the worst slowdown, and the slowdown at victim's output decreases if skew changes from w_1 to w_{worst} , or from w_{worst} to w_2 . The proof is omitted.

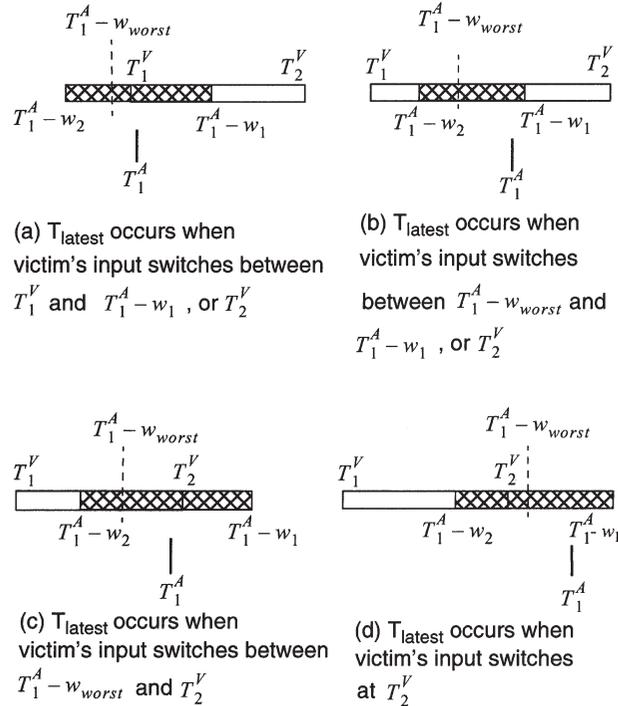


FIGURE 8 Illustration of Lemma 2.

LEMMA 1 Suppose that the victim's input switches at T_1^V , and aggressor's input switches in the interval $[T_1^A, T_2^A]$, T_{latest} occurs when the aggressor's input switches at one of the following times.

- (1) If $T_1^A \leq T_1^V + w_{\text{worst}} \leq T_2^A$, T_{latest} occurs when the aggressor's input switches at $T_1^V + w_{\text{worst}}$, as shown in Fig. 7(a);
- (2) If $T_1^A \leq T_2^A \leq T_1^V + w_{\text{worst}}$, T_{latest} occurs when the aggressor's input switches at T_2^A , as shown in Fig. 7(b);
- (3) If $T_1^V + w_{\text{worst}} \leq T_1^A \leq T_2^A$, T_{latest} occurs when the aggressor's input switches at T_1^A , as shown in Fig. 7(c).

According to the definition of the effective skew window, when aggressor's input switches at T_1^A , it causes slowdown at a victim if the victim's input switches between $[T_1^A - w_2, T_1^A - w_1]$. This leads to the following Lemma.

LEMMA 2 Suppose that the victim's input switches within the interval $[T_1^V, T_2^V]$, the aggressor's input switches at T_1^A , T_{latest} occurs at one of the following times:

- (1) If $T_1^A - w_{\text{worst}} \leq T_1^V \leq T_1^A - w_1 \leq T_2^V$, T_{latest} occurs when the victim's input switches between T_1^V and $T_1^A - w_1$, or T_2^V , as shown in Fig. 8(a);
- (2) If $T_1^V \leq T_1^A - w_{\text{worst}} \leq T_1^A - w_1 \leq T_2^V$, T_{latest} occurs when the victim's input switches between $T_1^A - w_{\text{worst}}$ and $T_1^A - w_1$, or T_2^V , as shown in Fig. 8(b);
- (3) If $T_1^V \leq T_1^A - w_{\text{worst}} \leq T_2^V \leq T_1^A - w_1$, T_{latest} occurs when the victim's input switches between $T_1^A - w_{\text{worst}}$ and T_2^V , as shown in Fig. 8(c);
- (4) Otherwise, T_{latest} occurs when the victim's input switches at T_2^V , as shown in Fig. 8(d).

For the cases shown in Fig. 8(b),(c), we observe that if the slowdown is always large or always small when aggressor's input skew changes from w_{worst} to w_1 , we only need to compute delay at some boundary points. Consider Fig. 9 and an arbitrary time point t_x between $T_1^A - w_{\text{worst}}$ and $T_1^A - w_1$. Let $f(t_x)$ denote the slowdown at the victim's output when its input switches at the time t_x . Figure 4 shows how $f(t_x)$ changes with the change of skews.

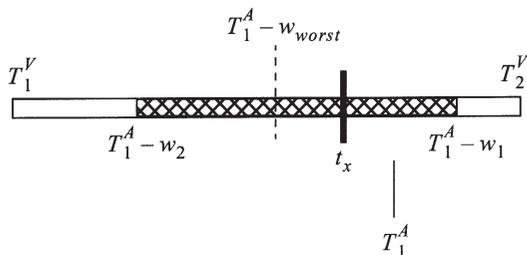


FIGURE 9 Illustration of Corollary 1.

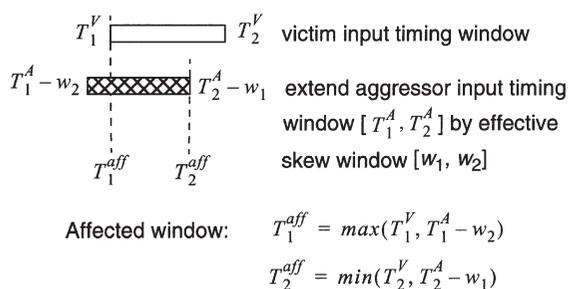


FIGURE 10 Finding the affected window.

COROLLARY 1 If $T_1^V \leq T_1^A - w_{worst} \leq T_2^V$, and if for any given time between $T_1^A - w_{worst}$ and $T_1^A - w_1$, the slope of the function $f(t_x)$ is always larger than -1 , then T_{latest} occurs when the victim's input switches at either $T_1^A - w_1$ or T_2^V . If the slope of the function $f(t_x)$ is always smaller than -1 , then T_{latest} occurs when the victim's input switches at either $T_1^A - w_{worst}$ or T_2^V .

From our experiments, as shown in Fig. 4, we found that slope of $f(t_x)$ is always larger than -1 when input skew changes from w_{worst} to w_1 , and may become smaller than -1 when the input skew changes from w_{worst} to w_2 . This can be explained using Fig. 5(a). When the noise pulse arrives between $[t_1^r, t_{estimate}]$, the aggressor has a gradually increasing impact on the victim's delay, but if the noise pulse arrives after $t_{estimate}$, the impact of the aggressor decreases rapidly. When the noise pulse arrives at a time closer to $t_{estimate}$, the impact of the aggressor on the victim delay is significant. However, more computation is required for identifying the exact point when $1 + f'(t_x)$ changes its sign, and for determining the exact effective skew window.

Based on the above discussion, we assume that the maximum slowdown at the victim's output occurs when the aggressor has a skew within the interval $[t_1^r - t_{peak}, t_2^r - t_{peak}]$ as obtained from the procedure *simple_worst_delay*. This will result in a pessimistic estimation of the coupling delay, but it will greatly simplify the estimation process. We will use this simplification in the first run of coupling delay analysis. A more sophisticated engine can be employed to improve the accuracy if necessary.

Based on the above simplification, we use the procedure *estimate_LAT* shown in Fig. 11 to estimate the worst delay. First we find the sub-interval of victim's input

Procedure estimate_LAT

(1) Compute simple worst case:

Call procedure *simple_worst_delay*(S_1^V, S_1^A) to compute:

the nominal delay $t_{nominal}$,

the worst case slowdown $t_{slowdown}$,

the worst input skew w_{worst} ,

and effective skew window $[w_1, w_2]$ if they are not known.

(2) Compute the victim's input affected interval:

$$T_1^{aff} = \max(T_1^V, T_1^A - w_2), T_2^{aff} = \min(T_2^V, T_2^A - w_1),$$

if $T_1^{aff} > T_2^{aff}$,

then the aggressor does not cause slowdown on victim,

$$T_{latest} = T_2^V + t_{nominal}, \text{ return;}$$

(3) Compute T_{latest} :

$$(a) T_{temp1} = T_2^{aff} + t_{nominal} + t_{slowdown};$$

$$(b) T_{temp2} = T_2^V + t_{nominal};$$

$$T_{latest} = \max(T_{temp1}, T_{temp2}).$$

FIGURE 11 Procedure to estimate the latest arrival time.

switching time that may be affected by the aggressor, and we denote this sub-interval as $[T_1^{aff}, T_2^{aff}]$. It can be found by using effective skew window, as shown in Fig. 10. Then we compare the delays when the victim's input switches at T_2^V and T_2^{aff} .

Case4 SALAT: Given timing windows, and variable slew rates: The victim's input timing window is $[T_1^V, T_2^V]$, and its slew rate range is $[S_1^V, S_2^V]$; the aggressor's input timing window is $[T_1^A, T_2^A]$, and its slew rate range is $[S_1^A, S_2^A]$, compute the latest arrival time T_{latest} at the victim's output.

We can solve this case by calling *estimate_LAT* for four corner cases: the fastest and the slowest aggressor and victim. We prefer to apply the following simpler method. We first find the largest effective skew window by checking two corner cases, i.e. the slowest victim with the fastest aggressor, and the fastest victim with the slowest aggressor. In the procedure *simple_worst_delay* step 6, the effective skew window is determined by the 20% point delay and 80% point delay, and the peak noise delay. The maximum slowdown can be found in the case when the aggressor is fastest, and the victim is slowest. Then, assuming that the maximum slowdown occurring at any point within the effective skew window, which is a conservative assumption, we just call the procedure *estimate_LAT* (only step 2, and step 3) to compute the latest arrival time at the victim's output. This solution can be easily extended to the multiple aggressors case.

Latest Arrival Time Estimation for Multiple Aggressors

When a victim has multiple coupling neighbors, the maximum slowdown caused by one aggressor depends on the switching conditions of the other aggressors.

TABLE II Delay in presence of multiple aggressors

Victim's input slew	400	400	400
Aggressor1 input slew	40	40	40
Aggressor2 input slew	quiet	40	400
Nominal delay	792	1095	1122
Maximum slowdown	438	416	414
Effective skew window	[- 745, 754]	[- 645, 1054]	[- 645, 1054]
Worst input skew at aggressor1	554	854	954
Worst output skew at aggressor1	- 358	- 332	- 285

To illustrate this point, in Table II we show some simulation results, units is ps.

For each column in Table II, aggressor2 has a starting point skew of 400 ps relative to the victim's input. The starting point skew of aggressor1 is swept from -4 ns to 4 ns. We identify the case when the delay at victim's output is at its maximum. The maximum slowdown caused by aggressor 1 occurs in column 1, when aggressor2 is quiet.

Before solving the Multiple Aggressor Latest Arrival estimation problem (MALAT), we first find for each aggressor the maximum slowdown t_{worst} and the largest effective skew window. We can compute t_{worst} by calling for each aggressor the *simple_worst_case* (S_2^V, S_1^A), while all other aggressors are quiet. The effective skew window is determined as $[t_1^r - t_{\text{peak}}, t_2^r - t_{\text{peak}}]$. The largest effective skew window for aggressor i can be found similarly to the *case 4* in the previous section: i.e. by checking the two corner cases: (1) the victim's input switches at the fastest rate S_1^V , the aggressor i switches at its slowest slew S_2^A in opposite direction, and all other aggressors switch at their fastest slews and the worst input skews in same direction as the victim; (2) the victim's input switches at the slowest slew S_2^V , all aggressors switch at their fastest slews and the worst input skews in opposite direction as the victim. The worst input skew is found by aligning the peak noise as in the procedure *simple_worst_delay*. Here we omit details. To ensure conservatism, we assume that the maximum slowdown by the aggressor i occurs in its whole effective skew window.

To compute the latest arrival time at victim's output T_{latest} , we first find, using the same method as shown in Fig. 10, the victim's input interval affected by each aggressor. Then we sort boundaries of all the affected intervals. T_{latest} is computed by sweeping these boundaries in a non increasing order, i.e. from right to left as shown in Fig. 12. If the right boundary of the interval i (affected by the aggressor i) is encountered, the slowdown at victim's output by the aggressor i is added; if the left boundary of

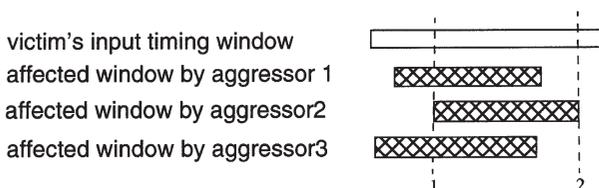


FIGURE 12 Affected intervals to sweep.

interval i is encountered, the slowdown by the aggressor i is removed. The details are described in the procedure *estimate_MALAT* in Fig. 13. The subscript i denotes the parameters of the aggressor i .

In the procedure, at boundary points, we add up the slowdown effects from multiple aggressors. Experimentally, we observe that this summation is pessimistic comparing to the worst case delay when multiple aggressors switch. When multiple aggressors switch, the effective load at the victim becomes larger, victim driver tends to become stronger. As a result, the impact of delay from each aggressor is smaller compared to the case when only the aggressor itself switches. Our static timer can be applied to investigate how severe coupling delay effects are. When higher accuracy is needed, and longer run time is affordable, worst case delay from multiple aggressors can be identified by waveform iteration method as explained in Ref. [14].

Complexity analysis: The complexity of the above worst delay estimation is dominated by the computation of the effective skew window and the worst-case slowdown. It is still a linear time process. The effective skew window can be found from the transition time information and the

Procedure estimate_MALAT

(1) Compute affected interval i for every aggressor i :

$$\text{Let } T_{i_1}^{\text{aff}} = \max(T_1^V, T_1^A - w_{i_2}),$$

$$T_{i_2}^{\text{aff}} = \min(T_2^V, T_2^A - w_{i_1}),$$

$$\text{If } T_{i_1}^{\text{aff}} > T_{i_2}^{\text{aff}}, \text{ set } T_{i_1}^{\text{aff}} = T_{i_2}^{\text{aff}} = \text{infinite},$$

(2) Sort boundaries:

sort all finite $T_{i_1}^{\text{aff}}, T_{i_2}^{\text{aff}}$ in non-decreasing order,
store in an array A;

(3) Sweep the boundaries to compute T_{latest} :

$$\Delta T = t_{\text{nominal}};$$

$$T_{\text{latest}} = T_2^V + \Delta T;$$

Start from the last boundary in the array A,
continue until all right boundaries are processed:

if current boundary is the right boundary $T_{i_2}^{\text{aff}}$ of an interval,

$$\Delta T = \Delta T + t_i^{\text{slowdown}}, T_{\text{temp}} = T_{i_2}^{\text{aff}} + \Delta T,$$

if current boundary is the left boundary $T_{i_1}^{\text{aff}}$ of an interval,

$$T_{\text{temp}} = T_{i_1}^{\text{aff}} + \Delta T, \Delta T = \Delta T - t_i^{\text{slowdown}},$$

if $T_{\text{temp}} > T_{\text{latest}}, T_{\text{latest}} = T_{\text{temp}}$.

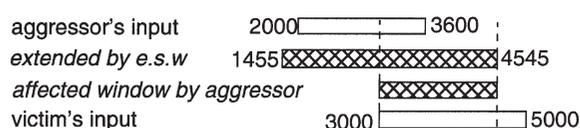
FIGURE 13 Estimate the latest arrival time with multi-aggressors.

peak noise delay. During the static timing analysis, from the known bounds of the slew rates of all the signals, we compute the effective skew windows for each signal once, and we do not recompute them when the slew rate ranges of some signals become smaller during the process. This makes the coupling analysis conservative. The sweeping procedure is efficient: sorting takes $O(n \log n)$, where n is the number of aggressors; sweeping the boundaries takes linear time. Notice that in practice, the number of aggressors acting on a victim may be quite large, with many of them having small coupling capacitances. However, those aggressors with small coupling capacitance would only contribute to small delay variation in different switching conditions. For this reason all of them can be treated as ground capacitance, which greatly reduces the complexity of crosstalk aware static timing analysis. A similar aggressor filtering procedure is also presented in Ref. [12].

Latest Arrival Time Estimation Based on Peak Noise Estimation

The sorting and sweeping method in the procedure *estimate_MALAT* is also applicable to the peak noise estimation. Suppose the aggressor i 's input switches within the interval $[T_{i1}^A, T_{i2}^A]$. Assume that the peak noise from aggressor i takes t_i^{peak} time to reach the victim's output. So the peak noise at victim's output from aggressor i occurs during the interval $[b_{i1}, b_{i2}] = [T_{i1}^A + t_i^{\text{peak}} - p_i, T_{i2}^A + t_i^{\text{peak}} + p_i]$. The effective noise width p_i is used to ensure conservatism in the noise estimation. By sorting and sweeping all the peak noise intervals $[b_{i1}, b_{i2}]$, we can compute the total peak noise in each sub-interval. Thus we can find the maximum peak noise V_{max}^P , and also the sub-interval $[T_1^P, T_2^P]$ during which this peak noise occurs. To further improve the accuracy, we compute the maximum peak noise at the victim's output by making all aggressors switching, with switching times at $T_1^P - t_i^{\text{peak}}$ for the aggressor i .

After identifying the peak noise within each sub-interval, we can use a similar method as the one described in the procedure *simple_worst_delay* to find the impact of the noise signal on the delay in different sub-intervals. To ensure conservatism, we compute the delay at each boundary point, i.e. the aggressor i switches at $b_{i1} - t_i^{\text{peak}}$, or $b_{i2} - t_i^{\text{peak}}$. Once we know delays at all boundary points,



effective skew window (e.s.w) [-945, 545]
 nominal delay of victim = 641, worst delay = 1082
 $T_{\text{latest}} = 5000 + 1082 = 6082$ by worst delay estimation
 $\max(4505 + 1082, 5000 + 641) = 5641$ by our method

FIGURE 14 Comparing worst delay and worst arrival time estimation.

we can easily compute the earliest and the latest arrival times of victim's output.

The above approach can be applied also in simulation based timing analysis. The computation cost would be higher because of the multiple aggressor switching delay computation. The method can be applied to a few critical coupling structures identified by our fast static timing analysis.

Comparing the Worst Delay Estimation and the Latest Arrival Time Estimation

We consider the example in Fig. 14 to explain differences between these two problems. We use some typical delay values in 0.35 μm technology. For a simple coupling structure such as shown in Fig. 2, assume the nominal delay of the victim to be 641 ps, and the worst case delay to be 1082 ps. If the aggressor's and the victim's input skews are within the range [-945, 545] (effective skew window), we observe coupling effects from the aggressor. The aggressor's and victim's input timing windows are shown in Fig. 14.

We show the latest arrival time computed according to the previously published method compared to those computed using our proposed method. The published method, described in Refs. [10,27], will consider the worst case delay to be 1082, and the victim's latest arrival time would be computed as 6082, which yields a pessimistic result. Our method can identify the delay when victim's input switches at different times, thus computing the victim's latest arrival time as 5641.

ITERATIVE TOPOLOGICAL ANALYSIS

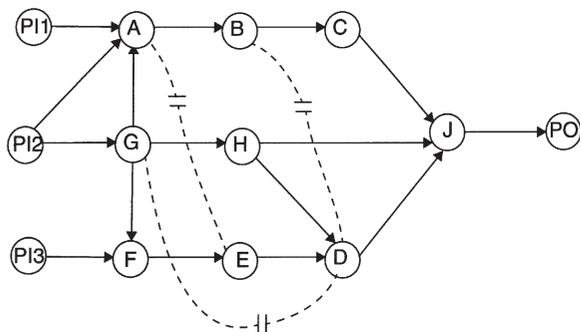
As with the traditional static timing analysis, we can quickly estimate the shortest and longest circuit delays by applying topological analysis. In this section, we will discuss topological analysis in the presence of capacitive coupling.

Iteration to Improve Accuracy

When there is no coupling in a combinational circuit, the static timing analyzer sweeps the circuit in any topological order and computes both the arrival time windows and the slew rate ranges for all the signals. The presence of coupling complicates the computation.

In Fig. 15(a), PI and PO represent I/O pins. Timing information at PIs are given. The other nodes represent logic gates, and the directed edges represent signal flow. A directed edge is from a gate's output to another gate's input. Two nodes are coupled if any two of their outputs are mutually coupled. Couplings are represented by the dashed edges with capacitances. In Fig. 15(a), the coupled nodes are: (A, E), (B, D) and (D, G).

In Fig. 15(b), we show two different topological orders to process nodes. Processing a node means computing



(a) An example circuit with coupling

PI2-->G-->PI2-->F-->E-->H-->D-->PI1-->A-->B-->C-->J-->PO

PI3-->PI2-->PI1-->G-->F-->A-->E-->H-->B-->D-->C-->J-->PO

(b) Two different orders to process nodes

FIGURE 15 An example of different topological orders.

delay at the node's output. Suppose that we process nodes using the first order in Fig. 15(b). When we compute delay at node G's output, we should know the timing window of node D's input. In the first sweep, there is no information about the node D yet, so we have to assume either that the timing window at node D's input is infinite or that the signal is quiet. In the second sweep, we can recompute the delay at node G's output using the available timing window at node D's input. But as node D's input may still change in the second sweep, we may need to do one or more iterations.

According to the assumption we use in the first sweep, there are two different iteration processes.

Iteration Starting from the Simple Worst Case

In the first sweep, we compute the *simple worst case* of each signal, assuming that its neighbors may switch at any time. Therefore, each signal will suffer the maximum slowdown or speedup from its neighbors. The *simple worst case* may be very pessimistic, for each signal it results in a timing window wider than the actual switching timing window. See Fig. 16.

In the next iteration, we use the available timing windows of neighbors' inputs to recompute each signal's timing window and to propagate the changes. This process continues until the timing windows at primary outputs do not change, or when the number of iterations exceeds

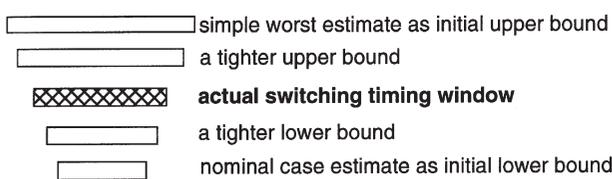


FIGURE 16 Different timing windows.

the specified value. This process will lead to tighter upper bounds of the actual switching timing windows for all signals in the circuit. This is stated in Lemma 3.

LEMMA 3 If the iteration process starts from the simple worst case for all signals, and the speedup and slowdown for each signal are computed using the method described in Fig. 13 (procedure *estimate_MALAT*), the timing windows for all signals gradually shrink or stay the same, and always cover the actual timing windows.

Proof Suppose we first compute the simple worst case for each signal in the circuit, i.e. coupling effects (both speedup and slowdown) are estimated based on the assumption that all neighbors of each signal may switch at any time.

In the first iteration, for each signal, we now recompute its timing window based on the available input timing windows of its neighbors. Coupling effects (speedup and slowdown) will not be greater compared to the case when we assumed all neighbors may switch at any time. So the latest arrival time of each signal will stay the same or become smaller, and the earliest arrival time will stay the same or become greater.

Assume the claim is true in *n*th iteration, i.e. timing window of each signal stays the same or becomes smaller.

Then in iteration *n* + 1 :

From the way we compute affected windows as explained in Fig. 10, since timing windows at its inputs and its neighbors' inputs stay the same or become smaller, the affected windows from each aggressor will stay the same or become smaller. In procedure *estimate_MALAT*, the latest arrival time is computed by sweeping boundary points of the affected windows as shown in Fig. 12, and adding worst slowdown from each aggressor at the right boundary of each affected window. So the latest arrival time of this signal will stay the same or decrease. Similarly, its earliest arrival time will stay the same or increase.

So in each iteration, each signal's timing window will stay the same or shrink.

Assuming that the maximum speedup and slowdown occur within the entire effective skew window, we perform a pessimistic estimation to coupling effects. The timing window of each signal in the iteration process always covers its actual timing window. Hence Lemma 3.

Iteration Starting from Nominal Case

In the first sweep, for each signal when its neighbors are quiet, we first compute the *nominal case* timing windows. The nominal case is an optimistic situation. For each signal, it results in a timing window narrower than the actual switching timing window, as depicted in Fig. 16.

In the next iteration, we use the available timing windows of neighbors' inputs to recompute the timing windows of each signal and to propagate the changes. This process continues until timing windows at the primary outputs do not change, or the number of iteration exceeds

the specified value. Signals are assumed to be uncorrelated. This process leads to tighter lower bounds on the signal switching timing windows for all signals in the circuit. This is stated in Lemma 4, which can be proved by induction. The proof is similar to the one for Lemma 3.

LEMMA 4 Assume that the signals are not correlated. If the iteration process starts from the nominal case for all the signals, and both speedup and slowdown for each signal are computed using the method described in Fig. 13 (procedure *estimate_MALAT*), the timing windows for all signals gradually enlarge or remain unchanged.

For a well tuned design, the coupling effects are not very severe, and in such a case, the iteration may start either from the simple worst case or the nominal case and converges quickly. However, for some designs, especially the designs in early stages, there is no guarantee that either iteration will converge quickly. In the case when fast convergence does not occur, different iteration processes starting from the simple worst case and the nominal case can be performed, with two or three iterations in each process. The difference in the results reflects the uncertainty of the coupling delay analysis.

Non-iteration Condition

Assume that in Fig. 15(a) only the (B, D) coupling exists. To compute the delay at node D’s output, we need to know both node D’s inputs, and node B’s inputs. If we use the first order to process nodes, these conditions are not satisfied, as one of node B’s inputs, node A, is processed after node D. But if we use the second order, these conditions are satisfied and no iteration of coupling analysis is necessary.

Non iteration condition: Assume there is a coupling between two nodes (A, B). If all inputs of node A are processed before node B, and all inputs of node B are processed before node A, then iteration due to coupling (A, B) can be avoided in the coupling delay analysis.

In the second order of Fig. 15(b), the non iteration conditions for coupling (A, E) and (B, D) are satisfied. However, as node D is in node G’s fanout cone, it is not possible to process node D’s input, i.e. nodes E and H, before the node G.

Assume there is a coupling between two nodes (A, B). If node A is not in node B’s fanout cone, and node B is not in node A’s fanout cone, then there exists an order which satisfies the non iteration conditions, i.e. all inputs of node A are processed before those of node B, and all inputs of node B are processed before node A.

Given the order to visit signals when computing the arrival times, if the non iteration condition for a coupling edge cannot be satisfied, we call it a *violation*. The total number of violations reflects the potential iteration cost. For example, in the first order in Fig. 15(b), there are two violations, and there is only one violation in the second order, so the second order is a better order. If node G is

coupled to node D, and node D is in node G’s fanout cone, we call it an *intrinsic violation*. The number of intrinsic violations is the minimum number of violations that a topological order can achieve.

Assume that the signals A and B are mutually coupled, and A_{in} and B_{in} are their inputs, respectively. The task of generating a topological order is equivalent to assigning an ID number to each node. Assume $S(A)$ is the ID of the signal A. V denotes a violation associated with each coupling edge (A, B) in the circuit. The task of generating a topological order with a minimum number of violations can be formulated as follows:

$$\text{Minimize } \sum_{i=1}^m V_i,$$

where m is the number of coupling edges in the circuit,

$$V_i = \begin{cases} 0 & \text{if } A_{in} < B, \text{ and } B_{in} < A; \\ 1 & \text{otherwise} \end{cases}$$

and $S(A_{in}) < S(A)$.

We use a greedy method to solve this problem, i.e. we perform a breadth first search from primary outputs, each time selecting the signal with the smallest cost.

Other Considerations

Multiple Intervals for Slowdown Effect

In solving the MALAT problem as described in section “Worst Arrival Time Estimation”, we find the effective skew windows for all aggressors, assuming that the worst *slowdown* caused by each aggressor will occur if its input skew is anywhere within the whole effective skew window. This is a pessimistic assumption. As can be seen in Fig. 4, a significant slowdown occurs only within a relatively small sub-interval. One possible improvement would be to divide the effective skew window into three consecutive intervals. In the first and last intervals, a slowdown may occur but only 50% at most. In the middle interval the worst *slowdown* may occur. This method may further improve accuracy, but it would require extra effort to characterize and store the effective skew windows. The run time overhead for sorting and sweeping to solve the MALAT problem would increase too. These modifications could be applied to critical nets after the fast estimation.

Multiple Intervals for Timing Window

In our static timing analysis, we record the earliest and latest arrival times for all pins in the circuit. Using this simple timing window to check coupling effects may result in overestimating delay. For example, two signals may have only two different arrival time points. Thus signal A may arrive at 200 or 1000; signal B may arrive at 600 and 1600. Although signal A will never be slowed

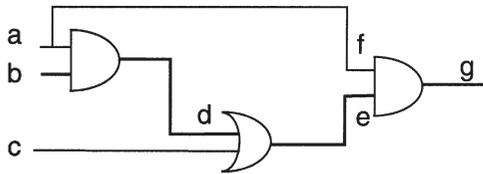


FIGURE 17 An example of a path.

down by signal B , we could misjudge this situation if we were using the simple timing window to represent the signal arrival times.

If we could record all possible arrival time points for a signal, we could more accurately estimate the coupling delays. However, such recording is too memory intensive, and we found it to be computationally infeasible for some circuits. Multiple intervals [17] have been shown to be effective in power estimation. Using multiple intervals will increase the run time when computing the latest arrival times as described in Section “Worst Arrival Time Estimation”. Besides, merging the intervals would impose additional run time overheads. The simple timing windows are more robust and assure conservative timing analysis. Considering all the above factors, we decided to use only the simple windows.

FUNCTIONAL CORRELATION ANALYSIS IN CRITICAL PATHS IDENTIFICATION

In the above topological analysis, we have not considered gates’ functionalities and their relationship in the circuit. Ignoring the functional correlation will result in pessimistic circuit delay estimation. In this section, we describe a method of performing functional correlation analysis for the coupling neighbors. We will apply it to identify the critical paths in the circuit.

Consider a physical path $\{b, d, e, g\}$ as shown in Fig. 17. *On inputs* are the signals on the path. *Side inputs* are the fan-in signals to the gates along the path but are not themselves on the path. For example, c is a side input of signal d . *Neighbors* of an *on input* are signals routed next to it in the layout of the circuit. A logic value is a *controlling value* (cv) for a gate if and only if it determines the gate’s output independently of the other inputs. For example, the controlling value of an AND gate is 0, and the controlling value of an OR gate is 1. The *noncontrolling value* (ncv) for a gate is the complementary value of its cv .

Sensitizable Path

In a circuit, there may be many paths which do not propagate transitions to the primary outputs. These paths are called *false paths*. False paths do not determine the circuit’s performance. A path is sensitizable if there exists test vectors causing a transition to propagate along

the path. Path sensitization has been studied since the late 80s and early 90s [4,11,19,28]. In Ref. [19] it has been shown that the path sensitization problem is equivalent to the SAT problem, and finding the longest sensitizable path in a circuit is NP-hard. The commonly used path sensitization criteria include the floating mode sensitization [4] and the viable path [19].

Floating mode analysis assumes that the previous state of each signal is unknown. A path is considered sensitizable if it satisfies the following conditions [4]:

DEFINITION 1 *Floating mode sensitizable path*: a path is sensitizable if and only if each on input is either the earliest controlling value, or the latest noncontrolling value while all the side inputs have noncontrolling values.

It is computationally expensive to check the above exact condition for each path in a circuit. In Ref. [11] a necessary condition has been proposed:

DEFINITION 2 *Functional irredundant path*: if an on input settles at a noncontrolling value, its corresponding side inputs must also settle at the noncontrolling value.

The above functional irredundant condition is applied in Ref. [11]. When checking to see whether a physical path satisfies a functional irredundancy condition or not, the path is treated as two logical paths: one for the rising transition at the primary input, and one for the falling transition. Starting from a primary input with a logical value, a partial path is extended by one gate at a time. The side inputs of the gate are all set to noncontrolling values if the gate’s on input has a noncontrolling value. The values at the side inputs must be justified. Complete value justification is very time consuming for large circuits. Thus, to speed up the process, only implications are performed [11].

It has been shown that the above procedure usually identifies a large portion of paths as functional redundant, i.e. false paths. We will apply the above procedure before identifying functional correlations between the coupling neighbors.

Signal Correlations

Functional information about a path’s coupling neighbors may be helpful in predicting the path’s delay more accurately. We first introduce some concepts describing functional correlations of signals:

DEFINITION 3 If coupled signals A and B always settle to the same logic value, they are called *completely obedient neighbors*.

DEFINITION 4 If coupled signals A and B always settle to the opposite logic values, they are called *completely disobedient neighbors*.

Figure 18 shows a simple example of this type of functional correlation. Signals a and b , if routed next to each other, are completely obedient neighbors. Signals a and c , if routed next to each other, are completely disobedient neighbors.

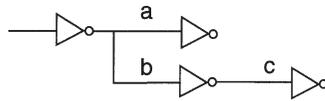


FIGURE 18 Simple correlations.

The completely obedient neighbors have very strong functional correlation. There does not exist an input vector that causes completely obedient (disobedient) signals to switch in opposite (same) directions, or the effect of such a switching cannot be propagated to any primary output. Information about such signals is useful in layout synthesis to reduce crosstalk effect. They can be found using the approach described in Ref. [16] or by applying a recursive learning technique [18].

There exist also other kinds of functional correlations. Suppose that a certain coupling effect between two signals may only be sensitized and propagated along some paths, but not along every path containing these two signals. It is helpful to identify which paths may be affected. If coupling effects propagate only along the short paths, a circuit's performance may not be affected. In such a case no timing optimization is necessary to speed up the circuit. We have the following definitions for path dependent correlations.

DEFINITION 5 If two neighboring signals A and B always settle to the same value when the path P containing signal A propagates a transition, we call the signal B a *path P -dependent obedient neighbor* of A .

DEFINITION 6 If two neighboring signals A and B always settle to the opposite values when the path P containing the signal A propagates a transition, we call the signal B a *path P -dependent disobedient neighbor* of A .

DEFINITION 7 If a coupling neighbor is neither a path dependent obedient neighbor nor a disobedient neighbor, we call it an *active neighbor*.

There are many input vectors that may sensitize a path. Under different input vectors the *active neighbors* may switch in the opposite or the same directions with respect to the on input in the give path P .

Path-dependent obedient (disobedient) neighbors are not necessarily completely obedient (disobedient) neighbors. Identifying path dependent obedient or disobedient neighbors is performed after a path is identified as functional irredundant. When implications are applied to verify if a given path P is sensitizable or not, some signals may also have implied values. For a signal a_i on a sensitizable path P , we check the implied values of a_i 's coupling neighbors. If a neighbor has the same value as the signal a_i , this neighbor is a path-dependent obedient neighbor. If it has an opposite value to the signal a_i , it is a path dependent disobedient neighbor. In this way we find some path P -dependent obedient and disobedient neighbors.

Recursive learning [18] is an efficient technique to identify logical correlations. Some completely obedient and disobedient neighbors can be found while performing recursive learning before the sensitization condition of any path is checked. Applying recursive learning again after the path sensitization is checked, more path-dependent obedient and disobedient neighbors can be found.

Path Delay Estimation

Once we have functional correlation information about the coupling neighbors for a path, we can predict the path delay more accurately by making use of the following observations.

Observation 1 If two signals are completely obedient neighbors, there is no need to consider a slowdown effect between them. If two signals are completely disobedient neighbors, there is no need to consider a speedup effect between them.

Observation 2 For a given path P propagating a transition, its path-dependent obedient neighbors do not cause a slowdown. Similarly, path P -dependent disobedient neighbors do not cause a speedup on P .

Before performing functional analysis, we have already performed the iterative topological analysis and computed a pessimistic bound of arrival timing windows on each signal in the circuit. When computing path delay after identifying path-dependent obedient and disobedient neighbors, we apply Observation 2 and recompute signal arrival timing windows for all on inputs of the path. The pessimistic arrival timing windows of active neighbors are used to estimate the coupling effects. The detailed procedure is a simple modification of *estimate_MALAT* described in section "Worst Arrival Time Estimation". When computing the latest arrival time for a victim's output, we do not consider slowdown effects from path-dependent obedient neighbors; and when computing the earliest arrival time at a victim's output, we do not consider speedup effects from the path dependent disobedient neighbors. Here we do not repeat the detailed description of the procedure.

Critical Path Identification

Critical paths are those paths whose delays are longer than the given specification. We call the gates on the critical paths as the *critical gates*. There are different methods to identify critical paths in the presence of crosstalk delay effects. They are described below beginning with the simplest and ending with the most complicated. Some of the simple methods do not guarantee that the selected critical paths are sensitizable.

1. Selecting paths based on negative slacks: a slack of each signal can be computed from its required time and arrival time. If the arrival time and the required times are computed accurately, the paths which contain only nodes with negative slacks are identified

as critical. This is the fastest way to identify the critical paths. Because of computational complexity, arrival time and required time are usually computed based on topological analysis. This method provides only a lower bound to the actual slack and may report pessimistically a large number of critical paths.

2. Selecting paths based on coupling delay of each path: coupling delay of each stage is computed as explained in Section “Worst Arrival Time Estimation”. Those paths whose coupling delays are longer than the timing specification are reported as critical by this method. The method removes the inaccuracy introduced by using estimated required time, so this approach reports less pessimistic results as compared to the first method. No gate functional information is considered.
3. Selecting paths based on coupling delay and sensitizability of each path: in this method, only functional irredundant paths whose coupling delays are greater than the specification are selected. The path selection method proposed in Ref. [5] can be used.
4. Selecting paths based on coupling delay, sensitizability and functional correlations between coupling neighbors: in this method, for each functional irredundant path, we identify the path-dependent obedient and disobedient neighbors, and then recompute the coupling delay of this path as explained before. Utilizing this kind of functional correlation information can reduce the pessimism in estimating path delay. The number of critical paths selected is much smaller than the number reported by previous methods. We will explain the details of this method later in this section.
5. Identifying critical paths by simulation: simulating a given path under various input vectors provides the most accurate delay of the path, and allows us to determine which paths are actually critical under particular input vectors. Simulating each path in the circuit is too expensive and impractical; thus usually only a small subset of paths selected by one of the above methods is simulated. Another difficulty is in generating the test vectors that cause the worst delay for a given path. This task is even more complicated in the presence of crosstalk delay effects. Our target is to identify the critical paths for static timing optimization; thus in this paper we do not discuss the issues of simulation and test vector generation.

Figure 19 presents a procedure for identifying critical paths when timing specification τ is given. Each path is treated as two logical paths: one for the rising transition at the primary input, and one for the falling transition. Starting from a primary input and a logical value, a partial path is extended by one gate at a time. The side inputs of the gate will all be set to noncontrolling value if its on input is noncontrolling value. The values at the side inputs must be justified. Complete value justification is very timing consuming for large circuits. Thus, only implication is applied to speed up the process. If implication

Procedure find_critical_paths

```
LSP = empty; PPS = empty;
For each primary input:
  Initialize partial path with value = 1 into PPS;
  Initialize partial path with value = 0 into PPS;
While (partial path P <- top of PPS)
  If P is not sensitizable:
    Remove from PPS, continue;
  If P is a complete path (reaches PO):
    Identify path dependent obedient and disobedient neighbors;
    Compute path coupling delay  $C_d$  using functional correlation;
    If  $C_d \geq \tau$ , P → LSP;
    Remove P from PPS;
  Else:
    Expand P into a new partial path  $P_{new}$ 
    by picking a remaining fanout;
    If  $esperance$  of  $P_{new} \geq \tau$ , insert  $P_{new}$  → PPS.
```

FIGURE 19 The algorithm of critical path selection.

reports conflict, it means that the partial path is functional redundant so no further traversal down this partial path is necessary.

Not only the functional irredundant condition checked for each partial path, but delay of each partial path is also computed. *Esperance* of a partial path is the delay of the longest path which contains the partial path. The longest distance from the gate to any primary output, d_{from_max} , is pre-computed in reverse topological order. *Esperance* of a partial path is computed by adding delay of this partial path and d_{from_max} of the ending gate of the partial path. If the *esperance* of a partial path is shorter than the timing specification τ , it means this partial path is not included in any critical path, so no further traversal down this partial path is attempted.

We denote by LSP the stored long sensitizable paths. We denote by PPS the stored, still investigated, partial paths. A partial path is extended one gate at a time, *esperance* is computed, and sensitization checking is incrementally applied after each extension. The short paths and unsensitizable partial paths are removed immediately from the PPS. For each partial path, we also store the signals required to sensitize it, which already have implied logic values. To save the memory usage, PPS will store only those paths that start from one primary input. To reduce run time complexity, we compute path delays using functional correlation information only for a complete path, i.e. when a path reaches primary output, and we do not order partial paths in PPS by *esperance*. The procedure can also be used to find the longest paths if τ is reasonably close to the longest path delay. It can also be modified to find the K longest paths, which will require searching all paths from all primary inputs at the same time, and ordering paths in PPS. This task is more memory-intensive.

Note that we can not only apply the above procedure to all paths in this circuit, but we can also choose to apply it only to some paths we are interested in. For example, we could choose paths which consists only of nodes with negative slacks. If the number of critical paths is large, it is

TABLE III Bound of coupling delay

Circuits	No. of cells	No. of pins	Max level	Nominal delay	Simple worst	Bound
C1355	583	1611	27	14.30	23.19 (62%)	19.06 (33%)
C1908	407	1574	26	12.15	21.78 (80%)	15.46 (27%)
C3540	812	3224	36	16.95	27.32 (61%)	19.70 (16%)
C499	360	1257	20	10.05	16.93 (69%)	13.66 (36%)
C6288	2435	7187	124	61.7	106.40 (72%)	77.88 (26%)
C880	311	1008	20	9.25	15.04 (63%)	11.59 (25%)
C17	13	25	3	2.4	3.95 (65%)	2.90 (21%)
C2670	971	2788	24	9.95	17.05 (71%)	11.51 (16%)
C432	190	599	22	11.35	19.77 (74%)	13.75 (21%)
C5315	1398	5408	30	14.9	24.81 (67%)	16.51 (11%)
C7552	2056	7474	27	12.45	23.18 (86%)	14.03 (13%)
Average	867	2924	33	15.95	27.22 (70%)	22.30 (20%)

better to perform the incremental path delay calculation and the sensitization check at the same time. If the number of critical paths is small, it is better to first find all topological critical paths and then perform the sensitization check only on those paths.

Functional correlation analysis gives less pessimistic estimation of path delays, identifying many fewer paths as critical. It is different from the false path analysis. The false paths analysis can be done before static timing analysis, and false paths can be removed from the timing graph when timing analysis is performed. But the correlation analysis may have to be run more than once, as it may be too expensive to store all correlation information if memory is limited.

EXPERIMENTAL RESULTS

Experiments with coupling delay analysis require layouts in deep submicron technology. We have extracted resistance and capacitance of some coupling structures in 0.35 μm technology, but we did not have complete layouts for the benchmark circuits. So we added delay information to the synthesized and mapped logic netlists, based on the data derived from these laid-out structures. In our experiments, each logic stage may have delay from 50 ps to 800 ps. Slew rate can be between 40 and 400 ps. Each wire has up to three coupling neighbors. Depending on the delay of the coupling neighbor, its contribution to

the coupling delay of each neighbor may cause up to 45% increase of the nominal delay.

We list results of the 11 ISCAS benchmark circuits in Table III. In our experiments, we get the same results for the longest path delay from both iteration processes whether starting from simple worst case or starting from the nominal case, which is the *bound* listed in the table. It takes at most five iterations to converge. The unit of delay is nano second (ns). Max. level is the maximum number of logic stages in the circuit. After the coupling effect of each neighbor is characterized in terms of the effective skew window and the worst slowdown, iterations proceed very fast. For all circuits, run times on the Sun Ultra 10 for five iterations are within 20 s. The percentage following each delay value is the increase of delay due to crosstalk, compared with the nominal case delay. The simple worst delay is very pessimistic (on the average, it is 70% more than the nominal case delay) but the actual longest path delay is only about 20% more than the nominal case delay.

In Table IV, we show how coupling delay of the longest path converges as the iterations progress starting from simple worst case. *Coupling delay1* shows results after the first iteration of simple worst delay, and *coupling delay2* shows results after the second iteration. The bound column shows the final converging value. Percentages shown represent extra delays caused by the coupling over the nominal delay. For most circuits, two iterations are enough. But it takes more iterations for C6288.

We have also tested a greedy approach to reduce the total number of violations in the nodes processing order.

TABLE IV Convergence of iterations

Circuits	Nominal delay	Coupling delay1	Coupling delay2	Bound
C1355	14.30	20.07 (40%)	19.06 (33%)	19.06 (33%)
C1908	12.15	16.32 (34%)	15.51 (28%)	15.46 (27%)
C3540	16.95	20.74 (22%)	19.70 (16%)	19.70 (16%)
C499	10.05	13.86 (38%)	13.66 (36%)	13.66 (36%)
C6288	61.7	80.16 (30%)	78.49 (27%)	77.88 (26%)
C880	9.25	12.02 (30%)	11.59 (25%)	11.59 (25%)
C17	2.4	2.90 (21%)	2.90 (21%)	2.90 (21%)
C2670	9.95	11.76 (18%)	11.51 (16%)	11.51 (16%)
C432	11.35	15.20 (34%)	14.01 (23%)	13.75 (21%)
C5315	14.9	16.83 (13%)	16.51 (11%)	16.51 (11%)
C7552	12.45	14.70 (18%)	14.03 (13%)	14.03 (13%)

TABLE V Range of coupling ratios

Circuits	Iterative topo. analysis (%)		Simple worst estimation (%)	
C1355	33	43	63	74
C1908	27	31	86	86
C3540	16	17	75	75
C499	34	39	68	70
C6288	26	29	72	78
C880	25	30	63	77
C17	21	22	65	90
C2670	16	24	71	95
C432	20	21	73	77
C5315	11	15	83	87
C7552	13	18	86	93

However, we have not observed any improvements in the run times. One reason is that the violations in the original node orders are dominated by the intrinsic violations, i.e. the nodes are coupled within the same fanout cones, and such violations cannot be resolved by reordering.

In Table V, we show the coupling ratios computed for each circuit. We randomly generated 1K instances for each benchmark circuit, with different wire lengths and coupling neighbors. *Coupling ratio* is the ratio of coupling-caused delay over the nominal delay of a circuit. We identify the minimum and maximum coupling ratios in these 1K instances, computed by our iterative topological analysis and the simple worst estimation. Our analysis always reports much less pessimistic results than the simple worst estimation.

To illustrate how functional correlation reduces the pessimism of path delay estimation, in Table VI, we show how coupling delay of 16 long paths in C499 changes if we use functional correlation analysis as described in the last section. We choose 16 paths whose topological delays are longer than the longest nominal delay. *Bound* in the table is the coupling delay obtained from iterative topological analysis starting from simple worst case. *Rising delay* and *falling delay* are coupling delays computed after functional correlation analysis, corresponding to the case when primary input is rising or falling, respectively. U means that the corresponding transition of the path cannot be sensitized; thus it does not impact the performance of the circuit. The percentage is the increase of a delay due to crosstalk compared to the nominal case delay. Using the functional correlation can reduce the pessimism of path delay estimation. Coupling delay computed with functional correlations for some paths may be 10% less than that computed without functional correlations.

In Table VII, we include results of the functional correlation analysis for some ISCAS benchmark circuits. Delays are in ns. We show delays of the longest path in the nominal case (“nominal delay”), delays after iterative topological analysis starting from the simple worst case (“topological long”), and delays after the functional correlation analysis (“functional long”). The percentages show the extra coupling delays in comparison to the

TABLE VI Functional coupling delay of C499

Path	Nominal delay	Bound	Rising delay	Falling delay
1	8.55	10.40 (22%)	10.01 (17%)	U
2	8.75	10.57 (21%)	10.17 (16%)	9.38 (7%)
3	8.75	10.62 (21%)	10.06 (15%)	U
4	8.75	10.57 (21%)	9.58 (9%)	10.17 (16%)
5	8.4	10.92 (30%)	U	10.00 (19%)
6	9.15	11.06 (21%)	10.05 (9%)	10.64 (16%)
7	8.55	10.67 (25%)	U	9.95 (16%)
8	8.5	10.28 (21%)	9.60 (13%)	U
9	8.4	10.92 (30%)	10.04 (20%)	U
10	9.5	12.09 (27%)	10.81 (14%)	11.08 (17%)
11	9.05	11.44 (26%)	10.42 (15%)	10.49 (16%)
12	8.95	11.85 (32%)	U	10.53 (18%)
13	8.30	11.23 (35%)	9.92 (20%)	9.40 (13%)
14	10	12.84 (28%)	11.84 (18%)	U
15	10	12.84 (28%)	U	11.69 (17%)
16	9.65	13.66 (42%)	12.64 (31%)	U

nominal delay. For circuit C6288 we could not finish the functional analysis to find the longest path, as there are too many paths in the circuit for the memory capacity of our machine. The functional correlation analysis can reduce substantially the pessimism of the static timing analyzer.

In Table VIII, we show the number of critical paths and the number of critical gates (gates on critical paths) reported by different analysis methods. Paths which have delays longer than the longest nominal delay in the circuit are considered critical paths. The column “total path” shows the total number of paths in each circuit. The “topo. long” column gives the number of paths whose coupling delays are longer, according to topological analysis, than the longest nominal delay in the circuit. The number of functional irredundant paths is listed in the column “irredundant long”. The column “violate” shows the number of long paths after functional correlation analysis. These are the paths whose coupling delay is still longer than the longest nominal delay.

In many circuits, the long sensitizable paths are just a small portion of all the paths in the circuit. And the difference between the number of topological long paths (“topo. long”) and actual long paths (“violate”) may be quite large. The difference between the “irredundant long.” and “violate” shows the effectiveness of detecting the path dependent signal correlations in path delay estimation.

TABLE VII Bound of coupling delay with functional analysis

Circuit	Nominal delay	Topological long	Functional long
C1355	14.30	19.06 (33%)	17.72 (24%)
C1908	12.15	15.46 (27%)	15.37 (27%)
C3540	16.95	19.70 (16%)	19.46 (15%)
C499	10.05	13.66 (36%)	13.11 (30%)
C6288	61.7	77.88 (26%)	?
C880	9.25	11.59 (25%)	11.43 (24%)
C2670	9.95	11.51 (16%)	11.51 (16%)
C432	11.35	13.75 (21%)	13.40 (18%)
C5315	14.9	16.51 (11%)	16.36 (10%)
C7552	12.45	14.06 (13%)	13.78 (11%)

TABLE VII Critical paths and gates

Circuit	Total path	Topo. long	Irredundant long	Violate	Total gates	Neg. slack	Topo. gates	Funct. gates	Run time
C1355	9,276,432	1,051,730	126,506	55,051	583	508	449	401	69 min
C1908	1,458,112	122,554	68,118	38,139	407	256	227	218	60 min
C3540	53,206,636	194,784	46,253	20,039	812	482	384	355	201 min
C499	695,776	64,112	40,528	17,752	360	284	243	240	20 min
C880	16,284	2248	2272	1796	311	102	66	66	42 s
C2670	488,476	5984	3276	2489	971	262	128	123	7 min
C432	483,652	16,738	9422	2661	190	127	119	113	7 min
C5315	2,682,418	6520	1548	1425	1398	272	113	107	5 min
C7552	1,452,636	5460	663	590	2056	704	153	146	10 min

The column “neg. slack” gives the number of gates with negative slack when we use only the simple worst delay to compute the required time for each signal. The column “topo. gates” gives the number of critical gates on the topologically long path. The column “funct. gates” gives the number of critical gates on the long paths after the functional correlation analysis. We perform the functional correlation analysis on all paths in the circuit, reporting the run times on Sun Ultra 10.

DISCUSSION AND FUTURE WORK

In this paper we have studied the problem of incorporating temporal and functional information of coupled signals to improve the accuracy of traditional static timing analysis. Our target is to perform efficient coupling delay estimation with acceptable error, which can be used for fast timing verification and for timing driven layout synthesis and optimization.

First we have discussed the problem of computing the earliest and the latest arrival times in crosstalk aware static timing analysis. We have shown that coupling effects should be considered in an effective skew window, and we have explained how to find an approximate effective skew window. We have proposed an efficient method to estimate the earliest and the latest arrival times when multiple aggressors are present. We have shown that our method can accurately predict the latest arrival times, whereas previous methods may yield overly optimistic or unduly pessimistic results.

We have investigated convergence issues in topological analysis. We have shown that the iterative update of timing windows is necessary when signals on the same paths are mutually coupled. The iteration converges quickly in our experiments.

For the first time in the literature, we have proposed to use an efficient functional correlation analysis based on path sensitization criteria to further reduce the pessimism of coupling delay estimation. Our experimental results show that the functional correlation analysis can yield a less pessimistic estimation of path delays, and identify many fewer critical paths than the methods which do not apply the functional correlation analysis. Functional correlation analysis, when applied to all the paths in the circuit, may be computation intensive for some circuits.

However, it can be applied after topological long paths are identified, and can significantly reduce number of critical paths falsely reported.

For technologies with features below 0.35 μm , capacitive coupling effects are even more severe. Efficient techniques to analyze and optimize crosstalk induced timing are and will remain essential. Our future work will include applying the described crosstalk aware static timing analyzer to guide layout synthesis in deep-submicron technologies.

Acknowledgements

The authors would like to thank Dr A Kristic for helpful discussions, and Dr Chung-yang (Ric) Huang for making his framework [15] including implication engines available for our experiments.

This work was supported in part by the National Science Foundation grant CCR 9811528, and by the State of California MICRO grant through Conexant and Synopsys.

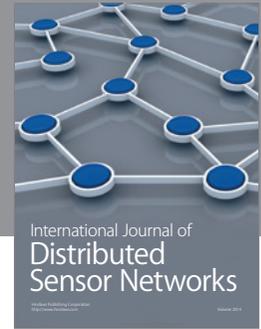
References

- [1] Arunachalam, R., Rajagopal, K. and Pileggi, L.T. (2000) “TACO: timing analysis with coupling”, *Proc. Des. Automation Conf.* **June**, 266–269.
- [2] Blaauw, D., Zolotov, V., Sundareswaren, S., Oh, C. and Panda, R. (2000) “Slope propagation in static timing analysis”, *Proc. Int. Conf. Comput.-Aided Des.* **Nov.**, 338–343.
- [3] Chen, H.C., Du, D.H.C. and Cheng, S.W. (1992) “Circuit enhancement by eliminating long false paths”, *Proc. Des. Automation Conf.* **June**, 249–252.
- [4] Chen, H.C. and Du, D.H.C. (1993) “Path sensitization in critical path problem”, *IEEE Trans. Comput.-Aided Des.* **12**(2), 196–207.
- [5] Chen, H.C., Du, D.H.C. and Liu, L.R. (1993) “Critical path selection for performance optimization”, *IEEE Trans. Comput.-Aided Des.* **12**(2), 185–195.
- [6] Dartu, F., “Gate and Transistor Level Waveform Calculation for Timing Analysis” Ph.D. Disst. Carnegie Mellon University.
- [7] Dartu, F. and Pileggi, L.T. (1997) “Calculating worst-case gate delays due to dominant capacitance coupling”, *Proc. 34th Des. Automation Conf.* **June**, 46–51.
- [8] Devgan, A. and Stok, L. (1997) “Timing analysis and optimization”, *Proc. Int. Conf. Comput.-Aided Des. Tutorial 1* **Nov.**
- [9] Chen, P. and Keutzer, K. (1999) “Towards true crosstalk noise analysis”, *Int. Conf. Comput.-Aided Des.* **Nov.**, 132–137.
- [10] Chen, P., Kirkpatrick, D.A. and Keutzer, K. (2000) “Switching window computation for static timing analysis in presence of crosstalk noise”, *Int. Conf. Comput.-Aided Des.* **Nov.**, 331–337.
- [11] Cheng, K.T. and Chen, H.C. (1996) “Classification and identification of nonrobust untestable path delay faults”, *IEEE Trans. Comput.-Aided Des.* **15**(8), 845–853.

- [12] Franzini, B., Forzan, C., Pandini, D., Scandolara, P. and Dal Fabbro, A. (2000) "Crosstalk aware static timing analysis: a two step approach", *Proc. Int. Sym. Quality Electronic Des.* **March**, 499–503.
- [13] Garey, M.R. and Johnson, D.S. (1991) *Computers and Intractability—A Guide to the Theory of NP-Completeness* (W.H. Freeman Company, New York).
- [14] Gross, P.D., Arunachalam, R., Rajagopal, K. and Pileggi, L.T. (1998) "Determination of worst-case aggressor alignment for delay calculation", *Proc. Int. Conf. Comput.-Aided Des.* **Nov.**, 212–219.
- [15] Huang, R. (C.-Y.), Wang, Y. and Cheng, K.-T. (1998) "LIBRA—A Library-Independent Framework for Post-Layout Performance Optimization", *Proc. Int. Sym. Physical Des.* **April**, 135–140.
- [16] Kirkpatrick, D.A. and Sangiovanni-Vincentelli, A.L. (1996) "Digital sensitivity: predicting signal interaction using functional analysis", *Proc. Int. Conf. Comput.-Aided Des.* **Nov.**, 536–541.
- [17] Kriplani, H., Najm, F.N. and Hajj, I.N. (1995) "Pattern independent maximum current estimation in power and ground buses of CMOS VLSI circuits: algorithms, signal correlations, and their resolution", *IEEE Trans. Comput.-Aided Des.* **14**(8), 998–1012.
- [18] Kunz, W. and Pradhan, D.K. (1994) "Recursive learning: a new implication technique for efficient solutions to CAD problems—test, verification, and optimization", *IEEE Trans. Comput.-Aided Des.* **13**(9), 1143–1158.
- [19] McGeer, P.C. and Brayton, R.K. (1991) *Integrating Functional and Temporal Domains in Logic Design—The False Path Problems and Its Implications* (Kluwer Academic Publisher).
- [20] Millman J. and Grabel A. *Microelectronics*, 2nd Ed., pp. 483–487.
- [21] Pileggi, L.T. (1990) "Asymptotic waveform evaluation for timing analysis", *IEEE Trans. Comput.-Aided Des.* **9**(4), 352–366.
- [22] Pillage, L. (1955) "Coping with RC(L) interconnect design headaches", *Proc. ICCCD*, 246–253.
- [23] Qian, J., Pulella, S. and Pillage, L. (1994) "Modeling the "Effective Capacitance" for the RC interconnect of CMOS gates", *IEEE Trans. Comput.-Aided Des.* **13**(12), 1526–1535.
- [24] Ringe, M., Lindenkreuz, T. and Barke, E. (2000) "Static timing analysis taking crosstalk into account", *Des., Automation Test Eur. Conf.* **March**, 451–455.
- [25] Sapatnekar, S.S. and Timing, A. (2000) "Model incorporating the effect of crosstalk on delay and its application to optimal channel routing", *IEEE Trans. Comput.-Aided Des.* **19**(5), 550–559.
- [26] Sasaki, Y. and DeMicheli, G. (1999) "Crosstalk delay analysis using relative window method", *Proc. IEEE Int. ASIC/SOC Conf.* **Sept.**, 9–13.
- [27] Sasaki, Y. and Yano, K. (2000) "Multiple aggressor relative window method for timing analysis including crosstalk delay degradation", *Proc. IEEE Custom Integrated Circuits Conf.* **May**, 495–498.
- [28] Sivaraman, M. and Strojwas, A.J. (1998) *A Unified Approach for Timing Verification and Delay Fault Testing* (Kluwer Academic Publishers).
- [29] Sun, S.Z., Du, D.H.C. and Hsi-Chuan, Chen (1998) "Efficient timing analysis for CMOS circuits considering data dependent delays", *IEEE Trans. Comput.-Aided Des.* **17**(6), 546–552.
- [30] Vittal, A. and Marek-Sadowska, M. (1997) "Crosstalk reduction for VLSI", *IEEE Trans. Comput.-Aided Des.* **16**(3), 290–298.
- [31] Vittal, A., Chen, L.H., Marek-Sadowska, M., Wang, K.P. and Yang, X. (1999) "Crosstalk in resistive VLSI interconnections", *IEEE Trans. Comput.-Aided Des.* **18**(12), 1817–1824.
- [32] Xiao, T. and Marek-Sadowska, M. (2000) "Efficient delay calculation in presence of crosstalk", *IEEE Int. Sym. Quality Electronic Des.* **March**, 491–497.
- [33] Yee, G., Chandra, R., Ganesan, V. and Sechen, C. (1997) "Wire delay in the presence of crosstalk", *Proc. TAU Workshop Timing Digital Syst.* **Dec.**, 170–175.

Tong Xiao received the B.S. degree in electrical engineering from Zhejiang University, P.R. China in 1990, the M.S. degree in electronics engineering from Southeast University, P.R. China in 1993, and the Ph.D. degree in Computer Engineering from University of California, Santa Barbara in 2001. She is currently working in Sun Microsystems, Inc. in Sunnyvale, California, Her research interests include timing analysis and optimization considering deep submicron signal integrity issues. She is a member of IEEE.

Malgorzata Marek-Sadowska received the M.S. degree in Applied Mathematics and the Ph.D. degree in Electrical Engineering from Politechnika Warszawska, Poland. From 1976 to 1982 she was an assistant professor at the Institute of Electron Technology at the Politechnika Warszawska. She became a research engineer at the Electronics Research Laboratory in 1982 and continued there until 1990, when she joined the Department of Electrical and Computer Engineering at the University of California, Santa Barbara, as a Professor. From 1993 to 1995 she was Editor-In-Chief of IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. She is IEEE Fellow.



Hindawi

Submit your manuscripts at
<http://www.hindawi.com>

