

TA-PSV - Timing Analysis for Partially Specified Vectors*

LIANG-CHI CHEN, SANDEEP K. GUPTA, and MELVIN A. BREUER

Department of Electrical Engineering - Systems, University of Southern California, Los Angeles, CA 90089-2562
{lichen, sandeep, mb}@poisson.usc.edu

Received May 22, 2001; Revised July 11, 2001

Editor: Chauchin Su

Abstract. We propose a new concept - timing analysis for partially specified vectors (TA-PSV) that enables the computation of tight timing windows. At one extreme, when the vectors are completely unspecified, TA-PSV reduces to static timing analysis (STA). At the other extreme, when the vectors are completely specified, TA-PSV performs timing simulation (TS). We present a systematic approach to construct a computationally feasible TA-PSV framework using a delay model that captures simultaneous to-controlling switching effects. We also demonstrate how TA-PSV can improve timing validation and also that TA-PSV significantly improves efficiency of timing-oriented test generation by reducing the search space.

Keywords: timing analysis for partially specified vectors (TA-PSV), delay model, static timing analysis (STA), crosstalk test generation (ATPG)

1. Introduction

Existing approaches for estimating delays in a combinational logic block generally fall into categories. At one extreme are static timing analysis (STA) techniques [1] that compute the minimum and maximum values of arrival and transition times for rising and falling transitions at each circuit line. These values are computed without considering any specific input vector and thus implicitly capture the minimum and maximum values over the universe of all possible vectors.

At the other extreme are a broad range of timing simulators that compute arrival and transition times with respect to a given pair of completely specified vectors [2][3][4]. These approaches provide different levels of accuracies at different computational complexities.

This research was motivated by the needs that arise during timing-oriented test generation [5][6]. Such test generators start with a pair of completely unspecified vectors. The min-max arrival and transition times for the universe of all possible pairs of vectors are computed via STA.

At an intermediate stage during timing-oriented test generation, specific values are assigned to some of the inputs while the values at other inputs remain unspecified. This is similar to what occurs when one executes the PODEM ATPG algorithm [7]. To reduce the amount of search required during such test generation, it is desirable that the min-max values of arrival and transition times be updated to capture tighter bounds associated with the subset of all possible vectors described by the partially-specified pair of vectors. This task, which cannot be performed using existing STA or timing simulation techniques, is called *timing analysis for partially specified vectors (TA-PSV)*.

When the input vectors are incrementally specified during test generation, only the timing information at the lines in the transitive fanouts of the refined input may need to be modified. In such cases, TA-PSV can

*. This work was supported in part by the Semiconductor Research Corp. under contract No. 98-TJ-646, and by Intel Corporation. A preliminary version of this paper was presented at the IEEE Asian Test Symposium, 2000.

be performed in an event-driven to refine the timing information. The event-driven version of TA-PSV is called incremental timing refinement (ITR) in [6][8][9][10].

We will present a new framework for TA-PSV. The development of such a framework employs a delay model as its basis. In addition, the delay model should have certain characteristics that enable identification of the combinations of transition and arrival times at a gate's inputs that lead to a particular *extreme value* of a timing range at its output. We will briefly describe a delay model [10] that has these characteristics and also captures the delay effect due to *simultaneous switching*. This is followed by the description of a STA using the delay model. Then we demonstrate a systematic approach to construct a TA-PSV framework. We show the application of TA-PSV on timing oriented ATPG and timing validation. Experimental results showing the extent to which TA-PSV improves the efficiency of timing-oriented test generation are provided.

2. Notation

- A/T/Q: arrival time, transition time, and required time, respectively (Section 5)
- R/F/tr: transition direction - rising, falling, and $tr \in \{R, F\}$ (Section 4)
- S/L: smallest and largest (Section 5)
- $Time_{tr, <extreme>}^{line}$: a timing information entity at a line, where $Time \in \{A, T\}$, $line$ is an index representing the line, $tr \in \{R, F\}$, $extreme \in \{S, L, \max\}$; \max will only be used in some special cases of this general form and explained later (Section 5). So $A_{R,S}^i$ is the smallest (S) value of the arrival time (A) at line i of a rising (R) transition.
- $\mathbf{d}^{gate\ output, <a\ gate\ input>}_{tr}$ (<list of input variables>): a delay function of a gate where tr is the transition direction of the gate output (Section 4.2). For example, $\mathbf{d}^{Z,X}_R(T^X_F)$ is the pin-to-pin delay (\mathbf{d}) of gate Z whose output is having a rising (R) transition due to a falling (F) transition at input line X with transition time T.
- $\mathbf{t}^{gate\ output, <a\ gate\ input>}_{tr}$ (<list of input variables>): an output transition time function of a gate, similar to \mathbf{d} above (Section 4.2)

- $T_{tr, \max}^{line}$: always appear as input variables of delay functions; the transition time on the line $line$ that maximizes the delay function (used in Section 5.3 and referring Figure 4 (a), (b))
- \bar{T}_{tr}^{line} : always appear as input variables of timing functions; the transition time on the line $line$ that maximizes the delay function for the given timing range (Section 5.3)
- $\delta^{line1, line2}$: the skew between transitions at $line1$ and $line2$ (Section 4)
- $\bar{\delta}^{line1, line2}$: the skew between to-controlling transitions at two inputs of a gate that minimizes the output transition time of the gate, given fixed input transition times (Section 5.3)
- $\bar{\delta}d^{line1, line2}$: the skew between two to-controlling transitions at two inputs of a gate that minimizes this gate's delay, for the given input transition time ranges (Appendix A)
- S_{tr}^Z : **state** of a transition tr on line Z (Section 6.1)

3. Motivation for TA-PSV

Using a delay model, *static timing analysis* (STA) [1] provides vector-independent min-max ranges of arrival and transition times for rising and falling transitions on each line in a circuit. Since STA considers the input vectors as being fully unspecified, the timing ranges are narrower when the input vectors are partially/fully specified. *Timing analysis for partially specified vectors* (TA-PSV) computes the ranges when input vectors are partially specified.

To perform STA given a gate delay model, we need to (a) handle both input and output timing as ranges, and (b) identify the worst case combinations, i.e., *corners*, over all possible combinations of possible logic values and arrival and transition times. This is needed so that, in forward calculations, values at the output of a gate can be calculated based on the ranges at its inputs.

An example of TA-PSV is illustrated in Figure 1 where logic values are shown as two time frames. Each line at each time frame can be 0, 1, or x. Timing ranges are shown as (minimal arrival time, maximal arrival time) for a rising or falling transition. (We also compute ranges of transition times at circuit lines. These values are not shown in Figure 1 just for avoiding

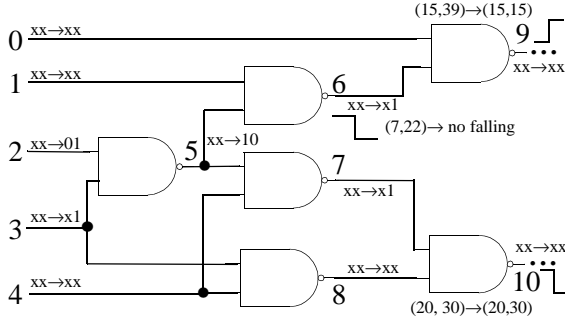


Fig. 1. An example of crosstalk test generation.

details.) We want to know if we can excite transitions in opposite directions at line 9 and 10 with a small skew (say, less than 3). Assume the target transitions are rising at line 9 and falling at line 10. Only timing ranges for the transitions of interest are shown.

All lines are initially xx with timing ranges obtained from static timing analysis. The initial arrival time ranges of the transitions at lines 9 and 10 are (15, 39) and (20, 30), respectively. The overlap of their timing ranges suggests that there *may* exist a test satisfying the desired condition.

After line 2 and line 3 are set to 01 and x1 (using a PODEM-like search [7]), using TA-PSV we find that the rising transition on line 9 has a timing range of (15, 15), i.e. at the time 15, and the falling transition on line 10 has a timing range of (20, 30). Thus the skew is not smaller than 3. Hence we can stop searching the subspace described by the above two assignments and backtrack, since no pair of vectors in this subspace can cause a significant slowdown effect due to capacitive coupling between lines 9 and 10 [11].

Without TA-PSV, this timing contradiction will not be found until the logic and timing values at lines 0, 6, 7, and 8 are fully specified, which might have required assignment of values to all five input lines, and therefore significantly increase the search time.

Without fully exploiting timing information, traditional STA and TS cannot find such contradictions early in the search process. As logic values become specified at additional primary inputs, the timing ranges at some lines *shrink*. A range may even disap-

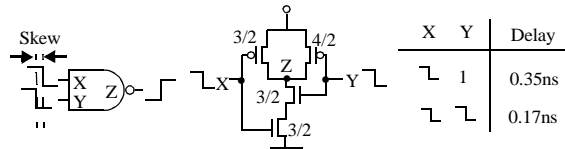


Fig. 2. Single vs. multiple to-controlling-value transitions at gate inputs.

pear if the corresponding transitions cannot occur. A gate's output timing range shrinks due to (a) the shrinkage of an input's timing range, or (b) the further specification of some input's logic values. New timing information for case (a) is computed by performing the same timing analysis on the new input timing ranges. New timing analysis methods are required to find the new worst case corners for (b) to extend STA to TA-PSV.

The proposed incremental timing analysis (TA-PSV) can be used to find tight bound for any given logic and timing information. It can therefore identify contradictions as early as possible and hence significantly reduce search complexity.

4. Proposed Delay Model

We use the delay model proposed in [10] to handle simultaneous to-controlling transitions. Using the same input variables as [12], this model gives more accurate empirical formulae than those presented in [13][14], for many cases that result in significant errors. The model has been validated using arbitrary skews over a typical range of input transition times [10].

A NAND gate, with output Z and two inputs X and Y (Figure 2), is used to illustrate the definitions. Here Z represents the gate output and also the gate. The **controlling value** of a multi-input gate Z , CV^Z , is that value, when applied to any of the gate's inputs, completely determines the value at its output. In the two-value logic system the **non-controlling value** of a gate Z , \overline{CV}^Z , is the complement of its controlling value. The **to-controlling transition** at an input of Z is denoted as a sequence of values $\langle \overline{CV}^Z, CV^Z \rangle$. If to-controlling transitions occur at one or more inputs of a gate, and the gate's non-controlling value is applied to its remaining inputs, then the transition at the gate output is called a **to-controlling response**. **To-non-controlling transition** and **response** are defined similarly.

The **transition time** (T_{tr}^X) of a transition tr , where $tr \in \{R, F\}$, on line X is the time required for a rising transition (**R**) to go from $0.1V_{dd}$ to $0.9V_{dd}$, and from $0.9V_{dd}$ to $0.1V_{dd}$ for a falling transition (**F**). The **arrival time** (A_{tr}^X) of a transition tr on line X is the time when the voltage at X reaches $0.5 V_{dd}$. The **skew**, $\delta^{X,Y}$, between transitions on lines X and Y is $A_{tr}^Y -$

A_{tr}^X . The **to-controlling gate delay function** d_{tr}^Z , defined as $A_{tr}^Z - \min(A_{tr}^X, A_{tr}^Y)$, is the gate delay of Z , where the output transition $tr \in \{R, F\}$ is a to-controlling response, and $R = \bar{F}$ and $F = \bar{R}$. The **pin-to-pin delay** from X to Z is the gate delay of Z when Y is steady at the non-controlling value and a transition is applied at X . $d_{tr}^{Z,X}$ is the pin-to-pin delay function from X to Z , where the output transition is tr . The **to-non-controlling gate delay function** is defined as $A_{tr}^Z - \max(A_{tr}^X, A_{tr}^Y)$, where $tr = F$ for NAND gates. Here A_{tr}^Z , the latest output arrival time computed through pin-to-pin delay, is $\max(A_{tr}^X + d_{tr}^{Z,X}, A_{tr}^Y + d_{tr}^{Z,Y})$. For simplicity, for the following presentation we assume that the first (last) input transition of a gate determines to-controlling (to-non-controlling) response for a gate. Although this is not always true, the required modification to the equations presented to correct for this approximation is straightforward. **To-controlling transition time function** t_{tr}^Z and **to-non-controlling transition time function** $t_{tr}^{Z,X}$ are defined similarly.

4.1 Delay Phenomena - Simultaneous Switching

Standard delay format (SDF) [15], which is commonly used for STA, uses pin-to-pin delays, i.e., delay from one input pin to the output pin, assuming all side-inputs are steady, and hence is not accurate for modeling simultaneous transitions with small skew values.

Simultaneous to-controlling transitions at inputs of a primitive gate decrease gate delay due to activation of multiple charge/discharge paths (Figure 2) [13]. *Simultaneous to-non-controlling transitions* at inputs of a primitive gate increase gate delay due to Miller effect [16]. The former is a first-order effect and the later is a second-order effect.

The delay model presented here captures the delay due to simultaneous to-controlling transitions. The pin-to-pin delay model is still used for simultaneous to-non-controlling transitions. Given the input skews and transition times of a gate, this model computes the gate delay and output transition time by formulating timing functions using empirical results.

To explain the speed-up caused by simultaneous falling transitions in Figure 2, we plot the to-controlling gate delay as a function of $\delta^{X,Y}$ for a fixed value of

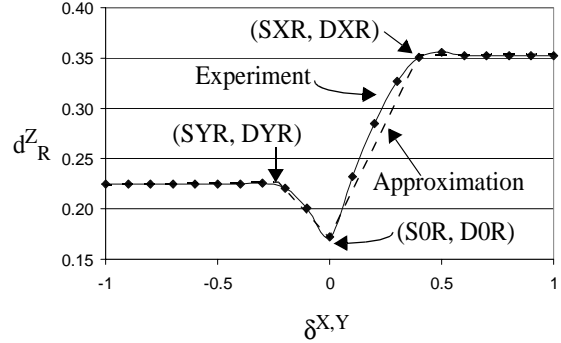


Fig. 3. Rising delay of two-input NAND gate as a function of $\delta^{X,Y}$ and its linear approximation.

T_F^X and T_F^Y , where $\delta^{X,Y} = A_F^Y - A_F^X$ (Figure 3). The speed-up caused by the simultaneous switches is significant only when $|\delta^{X,Y}|$ is small. When $|\delta^{X,Y}|$ is large, the delay is the same as the pin-to-pin delay. A linear approximation is also shown in Figure 3 along with the coordinators of the three points (SYR, DYR), (S0R, D0R), and (SXR, DXR) that define this approximation. Two transitions in the same direction on X and Y are called δ -simultaneous if $SYR \leq \delta^{X,Y} \leq SXR$. The time that output transition occurs is affected by multiple input transitions if input transitions are δ -simultaneous.

4.2 Timing Functions (for a Two-input NAND)

During test generation, all circuit parameters (e.g., device sizes and loads) remain fixed. In contrast, timing parameters (e.g., arrival times, transition times) may change from vector to vector. So the delay and transition times for a two-input NAND gate can be represented by functions of timing variables.

Given the *arrival times* and *transition times* of transitions at a gate's inputs, we compute the *gate delay* and *output transition time*. The *output arrival time* of a gate is computed using the input arrival times and gate delay.

Reconsider only the cases where all inputs of a gate have either non-controlling values or transitions to the same value. The reason is that the non-trivial output response caused by input transitions to opposite values at gate inputs are either (a) two separate transitions in opposite directions that can be processed separately using the method to be described, or (b) hazards where gate delay and output transition time are usually not the parameters of concern. The gate delay and output tran-

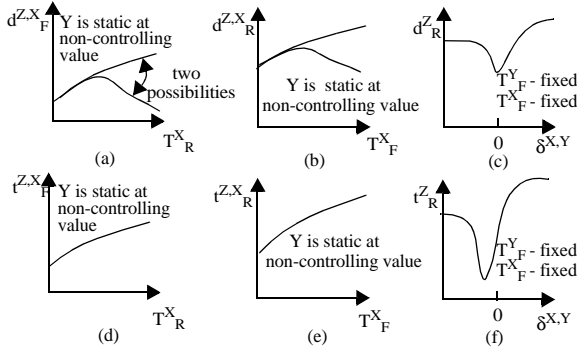


Fig. 4. Timing functions vs. input variables.

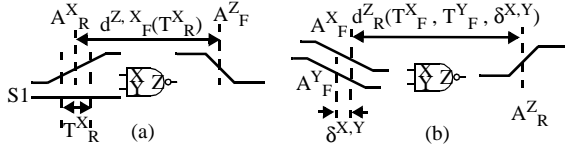


Fig. 5. (a) Fall delay function and (b) rise delay function.

sition time of a two-input NAND gate are represented by the following timing functions (Figure 5): (a) fall delay function (from input pin X), $d^{Z,X}_F(T^X_R)$; (b) fall transition time function (from input pin X), $t^{Z,X}_F(T^X_R)$; (c) rise delay function for two simultaneous input switching, $d^{Z,R}(T^X_F, T^Y_F, \delta^{X,Y})$; and (d) rise transition time function for two simultaneous input switching, $t^{Z,R}(T^X_F, T^Y_F, \delta^{X,Y})$.

4.3 Trends with Respect to Single Variables

The relations between output variables and each input variable for a two-input NAND gate (Figure 2) are further detailed in Figure 4.

Based on extensive simulations [10], we have identified that for fixed $\delta^{X,Y}$ and T^Y_{tr} , the gate delay as a function of T^X_{tr} (Figure 4 (a), (b)) may be either (a) monotonically increasing or (b) bi-tonic (monotonically increasing and then monotonically decreasing in this case). In case (b), the pin-to-pin delay may become negative for large T^X_{tr} . In such a case, this bi-tonicity is due to the fact that the input transition starts to pull the output voltage up (down) before the actual arrival time of the input transition, i.e., the time it reaches 0.5V_{dd}. The effective β_n/β_p ratio determines which shape the $T^X_{tr} - d^{Z,X}_{tr}$ curves take. Below we only treat case (b), because (a) is a special case of (b) with the

curve's peak at $T^X_{tr} = \text{infinity}$. The output transition time will always increase as T^X_{tr} increases (Figure 4 (d), (e)). Delay and output transition times have similar shapes with respect to skew (Figure 4 (c), (f)). The minimal delay always occurs at $\delta^{X,Y} = 0$, but the minimal transition time does not.

In the equations shown in the rest of the paper, we treat these four timing functions ($d^{Z,X}_F$, $t^{Z,X}_F$, $d^{Z,R}$, and $t^{Z,R}$) as known functions that capture the trends shown in Figure 4. Here we ignore the details of the delay model which can be found in [10].

5. Static Timing Analysis

We revisit STA for two reasons. First, the accuracy of STA depends heavily on the delay model used. Our delay model captures simultaneous to-controlling transitions and therefore improves STA accuracy. Second, TA-PSV is a generalization of STA. Many worst case corners and equations of TA-PSV are the same as that in STA. So when we discuss TA-PSV later, we can focus on the issues that do not exist in STA.

The three timing values considered in STA are arrival times (**A**), transition times (**T**), and required times (**Q**). Assume that primary input transitions occur at time 0. A primary output transition that occurs after a time later than T_0 , or not within a window $[T_1, T_2]$, is said to cause a **timing violation**. Given an internal node (line) X within a circuit, there is a minimum and maximum amount of delay that exists between X and a specific output. So, for a transition at X to avoid creating a timing violation at an output, the transition at X must occur within some range, referred to as the **required time**.

Static timing analysis provides min-max timing ranges for each line in a circuit for both rising and falling transitions. The ranges represent bounds on minimum and maximum delay values over all possible pairs of vectors. In timing analysis (Figure 6) arrival

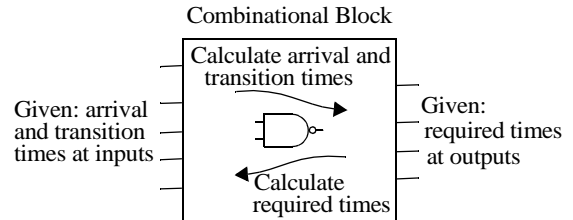


Fig. 6. Overall structure of timing analysis.

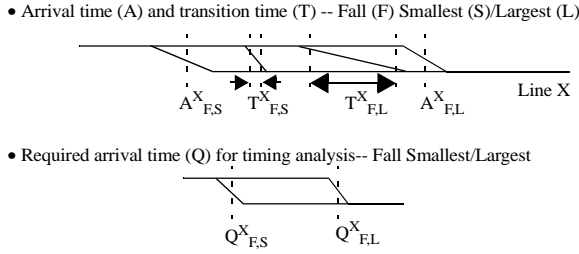


Fig. 7. Timing information used in our method.

times (**A**) and transition times (**T**) at a gate's output are calculated based on these corresponding values at the gate's inputs. These values are computed via a forward traversal starting at the primary inputs. Subsequently, the required times (**Q**) are computed via a backward traversal starting at the primary outputs. If the arrival time range does not overlap with the required time range for the rising/falling transitions at a line, then the given timing requirements cannot be satisfied. The min-max ranges in the proposed timing analysis are due to the unspecified input values, pulses, as well as approximations that ignore data dependencies caused by fanouts and reconverges. If all input values are specified, timing ranges become points.

5.1 Time Information

We use timing windows to represent min-max ranges, as shown in Figure 7. The smallest (**S**)/ largest (**L**) arrival times and the shortest (**S**)/ longest (**L**) transition times of rise/fall transitions are recorded for calculating the timing information for the next stage. The smallest (largest) arrival time of falling (rising) transition on line X is represented as $A_{F,S}^X$ ($A_{R,L}^X$). Transition and required times are represented similarly.

5.2 Worst Case Analysis - One View

A two-input NAND gate is used to demonstrate STA on our delay model. The key issue of min-max timing calculation is to identify the worst case combinations of A_{tr}^Z , T_{tr}^Z , and Q_{tr}^X , where $tr \in \{R, F\}$, based on the characteristics of the delay model, given the min-max timing ranges of X, Y, and Z. Note that A_{tr}^Z represents a single value in Section 4, but now it is extended to a timing range $[A_{tr,S}^Z, A_{tr,L}^Z]$.

In Figure 8 we illustrate how $A_{R,L}^Z$ is computed. In terms of logic values, there are three ways to justify a

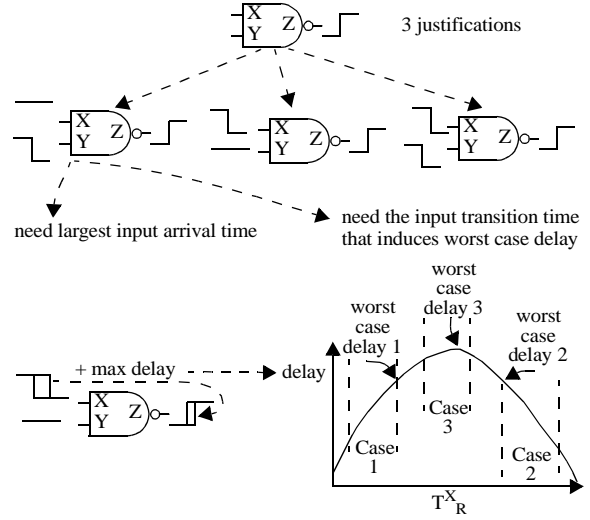


Fig. 8. Worst case analysis for latest output arrival time. rising transition at the output. These three justifications are analyzed separately and the results are combined to find the worst case.

In the first justification, $A_{R,L}^Z$ occurs when both the input arrival time and the gate delay are maximal. The maximal gate delay may occur when the input transition time is either (a) maximal, (b) minimal, or (c) at some values in between. These three scenarios correspond, respectively, to the three cases shown in T_{R}^X -delay curve in Figure 8, where the min-max range is to the left of the peak (Case 1), right of the peak (Case 2), and contains the peak (Case 3), respectively. The output response for the third justification needs to be computed by applying the proper T_{F}^X , T_{F}^Y , and skew values to delay function for simultaneous switching d_{R}^Z .

5.3 Calculating Arrival and Transition Times

Given the arrival and transition times at a gate's inputs, we calculate the corresponding quantities for the gate's outputs. To reduce computation effort, instead of trying all possible input combinations and taking the union to obtain the ranges of the output response, we perform the same operations only on all possible worst case corners. For example, for comput-

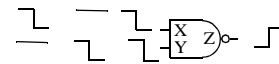


Fig. 9. Possible input combinations for output rising transition.

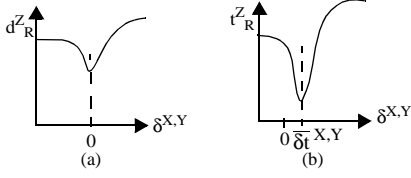


Fig. 10. (a) Delay as a function of skew, and (b) transition time as a function of skew.

ing $A^Z_{R,L}$ in Section 5.2, the third justification will never result in an A^Z_R larger than the first two, since simultaneous switching can only reduce the delay. So we only need to consider the first two justifications for computing $A^Z_{R,L}$.

The relations between output variables and input variables in Section 4.3 help identify the input A and T combinations that possibly induce worst case values on the computed output quantities. Calculation of A and T for output rising transition (Figure 9) using the proposed delay model is explained next.

$$A^Z_{R,S} = \min [A^X_{F,S}, A^Y_{F,S}] + \min_{\beta, \gamma \in \{S, L\}} [d^Z_R(T^X_{F,\beta}, T^Y_{F,\gamma}, A^Y_{F,S} - A^X_{F,S})] \quad (1)$$

$$A^Z_{R,L} = \max [A^X_{F,L} + [d^Z_R(T^X_{F,L}), A^Y_{F,L} + [d^Z_R(T^Y_{F,L})]]] \quad (2)$$

where $\bar{T}^X_F = \begin{cases} T^X_{F,\max}, & \text{if } T^X_{F,\max} \in (T^X_{F,S}, T^X_{F,L}); \\ T^X_{F,S}, & \text{else if } d^Z_R(T^X_{F,S}) > d^Z_R(T^X_{F,L}); \\ T^X_{F,L}, & \text{otherwise.} \end{cases}$

$T^X_{F,\max}$ is the value of T^X_F that maximizes $d^Z_R(T^X_F)$. \bar{T}^Y_F is defined similarly to \bar{T}^X_F .

$$T^Z_{R,S} = \begin{cases} t^Z_R(T^X_{F,S}, T^Y_{F,S}, \bar{\delta}^{X,Y}) & \text{if } (A^X_{F,S} + \bar{\delta}^{X,Y}, A^X_{F,L} + \bar{\delta}^{X,Y}) \\ & \cap (A^Y_{F,S}, A^Y_{F,L}) \neq \emptyset; \\ \min [t^Z_R(T^X_{F,S}, T^Y_{F,S}, A^Y_{F,S} - A^X_{F,L}), & \text{otherwise.} \\ t^Z_R(T^X_{F,S}, T^Y_{F,S}, A^Y_{F,L} - A^X_{F,S}), & \end{cases} \quad (3)$$

Here, $\bar{\delta}^{X,Y}$ is the value of skew $\delta^{X,Y}$ that minimizes $t^Z_R(T^X_F, T^Y_F, \delta^{X,Y})$ for given T^X_F and T^Y_F . $T^Z_{R,L} = \max [t^Z_R(T^X_{F,L}), t^Z_R(T^Y_{F,L})]$. (4)

In Equation (1), for an output rising transition to arrive as early as possible, we expect all input transitions to arrive as early as possible because an earlier arrival transition helps pull up the output earlier, compared with a later one, assuming both have the same transition times. The transition times at both X and Y should be either minimal or maximal, depending on which one causes shorter pin-to-pin delay on X and Y, since the shortest delay may be caused by the shortest

or longest transition time (Figure 8), but not at any other time in between. Equation (2) has been explained in Section 5.2 and earlier in this section.

In Equation (3), although minimal gate delay always occurs when $\delta^{X,Y} = 0$ (Figure 10 (a)), the minimal output rising transition time may occur when $\delta^{X,Y} = \bar{\delta}^{X,Y} \neq 0$ (Figure 10 (b)). $\delta^{X,Y} (= A^Y_F - A^X_F)$ may be equal to $\bar{\delta}^{X,Y}$ if $(A^X_{F,S} + \bar{\delta}^{X,Y}, A^X_{F,L} + \bar{\delta}^{X,Y}) \cap (A^Y_{F,S}, A^Y_{F,L}) \neq \emptyset$. Otherwise, either the minimal or maximal $\delta^{X,Y}$ closest to $\bar{\delta}^{X,Y}$ will cause a minimal output transition time. A minimal output transition time occurs when both input transition times are minimal since it monotonically increases with T^X_F and T^Y_F .

Calculation of A and T for an output falling transition employs a pin-to-pin delay as shown below.

$$A^Z_{F,S} = \min [A^X_{R,S} + \min \{d^Z_{X,F}(T^X_{R,S}), d^Z_{X,F}(T^X_{R,L})\}, \quad (5)$$

$$A^Y_{R,S} + \min \{d^Z_{Y,F}(T^Y_{R,S}), d^Z_{Y,F}(T^Y_{R,L})\}].$$

$$A^Z_{F,L} = \max [A^X_{R,L} + d^Z_{X,F}(\bar{T}^X_R), A^Y_{R,L} + d^Z_{Y,F}(\bar{T}^Y_R)] \quad (6)$$

where $\bar{T}^X_R = \begin{cases} T^X_{R,\max}, & \text{if } T^X_{R,\max} \in (T^X_{R,S}, T^X_{R,L}); \\ T^X_{R,S}, & \text{else if } d^Z_{X,F}(T^X_{R,S}) > d^Z_{X,F}(T^X_{R,L}); \\ T^X_{R,L}, & \text{otherwise.} \end{cases}$

Here, $T^X_{R,\max}$ is the value of T^X_R that maximizes $d^Z_{X,F}(T^X_R)$. $T^Y_{R,\max}$ is defined similarly. The transition time can be computed as

$$T^Z_{F,S} = \min \{t^Z_{X,F}(T^X_{R,S}), t^Z_{Y,F}(T^Y_{R,S})\}. \quad (7)$$

$$T^Z_{F,L} = \max \{t^Z_{X,F}(T^X_{R,L}), t^Z_{Y,F}(T^Y_{R,L})\}. \quad (8)$$

Calculation of the required times for STA can be found in Appendix A.

6. Timing Analysis for Partially Specified Vectors

After partially specific values are assigned to some circuit lines, worst case corners identified by STA may no longer occur. **Timing analysis for partially specified vectors (TA-PSV)** identifies worst case corners and computes timing information for any given partially specified input values. We will demonstrate how to perform TA-PSV for the proposed delay model.

6.1 Logic Value System

For timing simulation and test generation, two-vector tests are needed to create transitions carrying timing information. In addition to the timing information, a sequence of two values, (v_1, v_2) , is used to record the

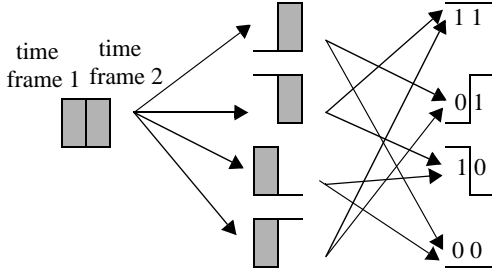


Fig. 11. Refinement of logic values (■ : X, □ : 0, and □ with horizontal line : 1).

logic information for each line. The line values in each timeframe can be 0, 1, or x, where x represents the unspecified value for a primary input, and the unknown value for any other line. The possible refinements of logic values are shown in Figure 11. As the value at a line is further specified, forward and backward logic implications may refine the values at other lines. The required implication procedure can be obtained by extending a basic implication method ([17]) to two time frames.

Among the nine logic values, {00, 01, 0x, 10, 11, 1x, x0, x1, xx}, for two-frame logic, 01 specifies a rising transition, and 0x, x1, and xx specify a potential rising transition. The remaining five values indicate that Z does not have a rising transition. According to the analysis for transitions on the nine logic values, we define the **state** of a transition tr on line Z, S_{tr}^Z , as follows:

$$S_{tr}^Z = \begin{cases} 1, & \text{if line Z has a transition } tr; \\ 0, & \text{if line Z potentially has a transition } tr; \\ -1, & \text{if line Z definitely does not have a transition } tr. \end{cases}$$

S_{tr}^Z captures the characteristics we need. In our analysis, for simplification, all possible partial specified values of input logic combinations will be classified according to the three possible values of S_{tr}^Z , instead of the original nine possible logic values.

S_{tr}^Z can be computed given the logic value on Z. When S_{tr}^Z is -1, none of the timing values pertaining to the transition tr at line Z are meaningful; each such timing value must hence be considered as undefined. (Verifying the state of a line before using this line's timing values will avoid these undefined values from being used incorrectly.) In the other two cases ($S_{tr}^Z = 1$ or 0), the timing parameters are identical to those in STA. STA is a special case of TA-PSV where $S_{tr} = 0$ for

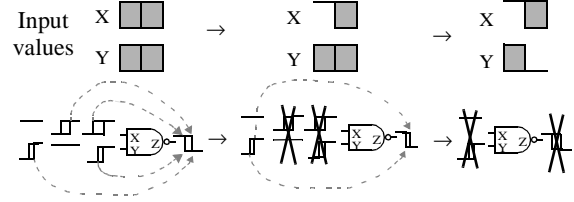


Fig. 12. An example of timing refinement. every line. A method to calculate the timing values at each line is illustrated next.

6.2 Timing Refinement

An example of TA-PSV for an output falling transition is shown in Figure 12. Initially both inputs X and Y are unspecified, and the ranges are the same as those in STA, where the minimal output arrival time is caused by the earlier of the rising transitions at X or Y, and the maximal output arrival time is caused by the later of the transitions at X or Y. When X is specified as 1x, no rising transition can occur at X, so the minimal output arrival time depends only on the rising transition at Y. If Y is later specified as x0, then no output falling transition is possible. When rising (falling) transitions on some lines definitely occur or definitely do not occur, the output timing ranges may shrink from the ranges computed in static timing analysis to smaller ranges or even disappear.

6.3 Proposed Approach

We have proposed a delay model where, given the input logic values and timing ranges at a gate's inputs, we can compute the gate's output response. But since the model itself is not capable of handling potential transition ($S_{tr}^Z = 0$), similarly to what we did in STA, we need to (a) find all possible input state combinations that may produce a worst case value for the output entity, (b) apply all these corners, and (c) select the appropriate one to obtain the output's worst case value.

The main difficulty of implementing such an TA-PSV is the enumeration of all possible cases to obtain the general solution. In STA, each line has the same logic value xx. But in TA-PSV, each of the nine logic values listed in Section 6.1 may be the current line value. For an n -input gate, there are 9^n input logic combinations in TA-PSV, compared with only one in STA.

In our approach (Figure 13), we first reduce the nine logic values to three possible states (Section 6.1).

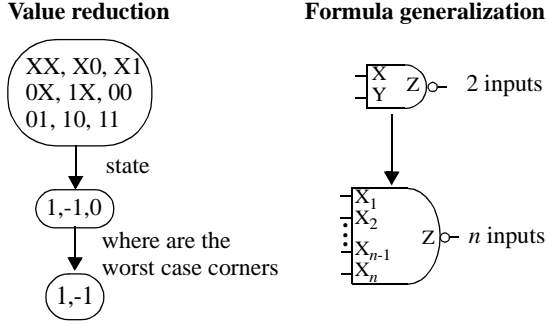


Fig. 13. Problem reduction in TA-PSV.

Focusing on worst case corner identification, we again convert the three states to two collapsed states (Section 6.4.2).

Formulation for TA-PSV is performed on a two-input NAND gate, in a way that can be easily extended to n inputs. Since each input may have one of these two collapsed states, only 2^2 possible combinations need to be analyzed, and then extended to the general case.

After the classification for TA-PSV, we represent all possible cases by the three original states (Section 6.4.3) since they can be computed directly from the logic values. State collapsing (Section 6.4.2) is performed only once in the pre-characterization stage and not in any TA-PSV computation.

6.4 Calculating Arrival and Transition Times

6.4.1 Concepts for State Collapsing

When an input of a gate has a potential transition ($S_{tr}=0$), the worst case value of computed output response entity (e.g., $A_{R,S}^Z$) may occur either when this input has a transition (set S_{tr} to 1) or not (set S_{tr} to -1). We will define the rules to set the zero-value states at inputs of a gate to either 1 or -1, to obtain the input corners that are candidates for exciting the worst case value at the gate output. So the three **original input states** (1, 0, -1) will be converted to two **collapsed input states** (1, -1). The conversion is shown in Table 1 that will be explained in Section 6.4.2.

Setting S_{tr} to 1 means specifying a line to have a desired transition. Setting S_{tr} to -1 means that this line is specified to have no transition, i.e., at both time frames it has a non-controlling value to allow the desired transition to be propagated.

If two distinct original input states can be converted to the same collapsed states, then they will have the same worst case value and can share the same equations for TA-PSV. For example, in Table 1, for optimization target $A_{F,L}^Z$, original input states (S_{R}^X, S_{R}^Y) = (0,0) and (0, 1) have the same collapsed input states (1, 1). We will classify all possible cases of input combinations based on these two collapsed states.

6.4.2 Identifying Worst Case Input Corners

Again a NAND gate with output Z and two inputs X and Y is used for illustrating TA-PSV. An **optimization target** ($OPT_{tr, extreme}^Z$) is an assignment of OPT to line Z whose *extreme* value is desired for transition tr , where $OPT \in \{T, A\}$, $tr \in \{R, F\}$ and $extreme \in \{S, L\}$. S_{tr}^Z is only related to S_{tr}^X and S_{tr}^Y , where \bar{tr} is R(F) when tr is F(R). In the following, this mapping is always assumed if we do not specify the transition direction.

If the output Z has a rising transition, then the first frame of X and Y should each be 1. During TA-PSV, we temporarily perform backward implication to reduce the number of combinations considered at the gate inputs and hence to capture tighter timing ranges.

To find the extreme value for an optimization target on Z , we need to determine (a) if this extreme value prefers single or multiple input transitions, (b) given the current logic values on X and Y (we lose some choices at X if $S_{tr}^X = 1$ or -1, similarly for Y), do we prefer to have transitions on X and Y if $S_{tr}^X = 0$ ($S_{tr}^Y = 0$), and (c) for the inputs with transitions, which corner to select (minimal, maximal, or peak) for A and T to excite the extreme value on the optimization target.

Rules for changing the zero value of S_{tr}^X for minimizing the arrival time at Z (for both rising and falling transitions) are shown below:

1. $S_{tr}^Y = -1$, where tr is either a to-controlling transition or a to-non-controlling transition: set S_{tr}^X to 1 for creating a transition at Z .
2. $S_{tr}^Y = 1$, where tr is a to-controlling transition: set S_{tr}^X to 1 because an additional input transition may speed up the output transition.

Table 1. The implied values of S_{tr} for obtaining the extreme cases for optimization target.

original input state		Optimization target															
		$A_{F,S}^Z$		$A_{F,L}^Z$		$A_{R,S}^Z$		$A_{R,L}^Z$		$T_{F,S}^Z$		$T_{F,L}^Z$		$T_{R,S}^Z$		$T_{R,L}^Z$	
S_{tr}^X	S_{tr}^Y	$S_{R,S}^X$	$S_{R,S}^Y$	$S_{F,L}^X$	$S_{F,L}^Y$	$S_{F,S}^X$	$S_{F,S}^Y$	$S_{R,L}^X$	$S_{R,L}^Y$	$S_{F,S}^X$	$S_{F,S}^Y$	$S_{F,L}^X$	$S_{F,L}^Y$	$S_{R,S}^X$	$S_{R,S}^Y$	$S_{R,L}^X$	$S_{R,L}^Y$
0	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1
0	0	1	-1	1	1	1	1	1	-1	1	-1	1	-1	1	1	1	-1
		-1	1							-1	1	-1	1	1	-1	-1	1
0	1	-1	1	1	1	1	1	-1	1	1	1	1	1	1	1	1	1
										-1	1	-1	1	-1	1	-1	1

- $S_{tr}^Y = 1$, where tr is a to-non-controlling transition: set S_{tr}^X to -1 because additional input transition may slow down the output transition.
- $S_{tr}^Y = 0$, where tr is a to-controlling transition: set (S_{tr}^X, S_{tr}^Y) to (1, 1), because simultaneous switching in this direction reduces the gate delay.
- $S_{tr}^Y = 0$, where tr is a to-non-controlling transition: try two possibilities, namely $(S_{tr}^X, S_{tr}^Y) = (1, -1)$ and $(-1, 1)$, because simultaneous switching is not desired, but at least one input transition is required to create a transition at the output.

According to these five rules, we obtain the zero-value setting of S_{tr}^X and S_{tr}^Y for the eight optimization targets - minimal/maximal output arrival and transition times for both rising and falling transitions. The setting for extreme values on optimization targets is shown in Table 1. Only the cases where S_{tr}^X is 0 are shown. The cases where S_{tr}^Y is 0 are symmetric to the shown cases. Non-zero S_{tr}^X and S_{tr}^Y will not be changed, so the cases with both S_{tr}^X and S_{tr}^Y are non-zero are not shown. There are cases where worst case corners can not be covered by one set of zero-value setting. In these cases, all input setting listed in Table 1 need to be tried, and the worst case values among them selected.

For worst case output arrival and transition times on a gate Z with inputs X_1, X_2, \dots, X_n , when $S_{tr}^{X_j} = 0$, there are three possible cases for setting $S_{tr}^{X_j}$ to -1 or 1, $\forall j \in \{1, 2, \dots, n\}$. The three cases are (1) set all such $S_{tr}^{X_j}$ to 1, (2) set all of them to -1, and (3) set one of them to 1 and all others to -1. After the worst case cor-

ners are found, A , T , and Q can be calculated by directly extending the above techniques.

When the effect of input position is considered or transistors of different sizes are used in the parallel transistor, to-controlling delay should be redefined with respect to the dominating input. Here **dominating input**, DI_{tr}^Z , of gate Z with inputs X_1, X_2, \dots, X_n , is the gate input X_j that the transition on X_j minimizes (maximizes) $(A_{trj}^{X_j} + d^{Z,X_j})$ for $j \in \{1, 2, \dots, n\}$, when tr is a to-controlling (to-non-controlling) response. Here trl equals tr for non-inverting gates, and \bar{tr} for inverting gates. The output arrival time becomes the sum of the dominating input arrival time plus gate delay.

6.4.3 TA-PSV Computation Examples - $A_{R,S}^Z$ and $A_{R,L}^Z$

Based on the TA-PSV approach proposed, we develop the formula for each optimization target. Two of them, $A_{R,S}^Z$ and $A_{R,L}^Z$, are shown below to illustrate the form of these formulae. According to original input states, the formulation of $A_{R,S}^Z$ and $A_{R,L}^Z$ is classified into three and six cases respectively. For most cases, the formulae are the same as that in STA or that according to pin-to-pin delay model. Some cases share the same formulae or use formulae symmetric to other cases. The increase in the complexity and the number of cases is controlled quite well. This makes TA-PSV computationally feasible.

For $A_{R,S}^Z$

Case 1. $S_F^X \neq -1$ and $S_F^Y \neq -1$: Use the equations for STA.

Case 2. $S_F^Y = -1$: only a falling transition at X may cause a rising transition at Z. For this case,

$$A_{R,S}^Z = A_{F,S}^X + \min [d_{R,F}^{Z,X}(T_{F,S}^X), d_{R,F}^{Z,X}(T_{F,L}^X)]. \quad (9)$$

Case 3. $S_F^X = -1$: similar to Case 2.

For $A_{R,L}^Z$

Case 1. $S_F^X = 0$ and $S_F^Y = 0$: Use the equations for STA.

Case 2. $S_F^X = 1$ and $S_F^Y = 1$: Both inputs have falling transitions. For this case,

$$A_{R,L}^Z = \min (A_{F,L}^X, A_{F,L}^Y) + d_{R,F}^{Z,X}(\bar{T}_F^X, \bar{T}_F^Y, A_{F,L}^Y - A_{F,L}^X). \quad (10)$$

Here \bar{T}_F^X is as defined in equation (2).

Case 3. $S_F^Y = -1$ and $S_F^X = 0$ or 1: Only the falling transition at X may cause a rising transition at Z. For this case,

$$A_{R,L}^Z = A_{F,L}^X + d_{R,F}^{Z,X}(\bar{T}_F^X). \quad (11)$$

Here \bar{T}_F^X is as defined in equation (2).

Case 4. $S_F^X = 1$ and $S_F^Y = 0$: Same equations as that in Case 3.

Case 5. $S_F^X = -1$ and $S_F^Y = 0$ or 1: similar to Case 3.

Case 6. $S_F^Y = 1$ and $S_F^X = 0$: same as Case 5.

When $S_F^Y = S_F^X = 1$, the extreme value occurs when both input transitions arrive as late as possible. The proper values of T^X and T^Y for maximizing the delay need to be explored. Here \bar{T}_F^X and \bar{T}_F^Y have three possible values as that for $A_{R,L}^Z$ in STA. But in STA, $T_{F,\max}^X$ maximizes $d_{R,F}^{Z,X}(T_F^X, T_F^Y, A_{F,L}^Y - A_{F,L}^X)$ only for fixed T_F^Y and $\delta^{X,Y}$. We use numerical method to iteratively calculate $T_{F,\max}^X$ and $T_{F,\max}^Y$ until the pair $(T_{F,\max}^X, T_{F,\max}^Y)$ that maximizes the delay function is found for a fixed $\delta^{X,Y}$. If the gate delay function in Figure 4 (b) is monotonic, then $\bar{T}_F^X = T_{F,L}^X$ and $\bar{T}_F^Y = T_{F,L}^Y$, so $A_{R,L}^Z$ can be identified without numerical computation.

6.4.4 TA-PSV on Pin-to-pin Delay Model

Using a pin-to-pin delay model, the equations for TA-PSV is simpler since in each case the output response will be determined by only one input. The output falling response is listed below.

$$A_{F,S}^Z = \begin{cases} \min [A_{R,S}^X + \min \{d_{R,S}^{Z,X}(T_{R,S}^X), d_{R,S}^{Z,X}(T_{R,L}^X)\}, \\ A_{R,S}^Y + \min \{d_{R,S}^{Z,Y}(T_{R,S}^Y), d_{R,S}^{Z,Y}(T_{R,L}^Y)\}] \\ \text{if rising transitions can occur at both X and Y, but} \\ \text{no definite rising transition is specified.} \\ \max [A_{R,S}^X + \min \{d_{R,S}^{Z,X}(T_{R,S}^X), d_{R,S}^{Z,X}(T_{R,L}^X)\}, \\ A_{R,S}^Y + \min \{d_{R,S}^{Z,Y}(T_{R,S}^Y), d_{R,S}^{Z,Y}(T_{R,L}^Y)\}] \\ \text{if definite rising transitions occur at both X and Y.} \\ A_{R,S}^Y + \min \{d_{R,S}^{Z,Y}(T_{R,S}^Y), d_{R,S}^{Z,Y}(T_{R,L}^Y)\} \\ \text{if a definite rising transition at Y, or no possible} \\ \text{rising transition at X.} \\ A_{R,S}^X + \min \{d_{R,S}^{Z,X}(T_{R,S}^X), d_{R,S}^{Z,X}(T_{R,L}^X)\} \\ \text{symmetric to the case above.} \end{cases} \quad (12)$$

$$A_{F,L}^Z = \begin{cases} \max [A_{R,L}^X + d_{R,F}^{Z,X}(\bar{T}_R^X), A_{R,L}^Y + d_{R,F}^{Z,Y}(\bar{T}_R^Y)] \\ \text{if rising transitions can occur at both X and Y} \\ A_{R,L}^Y + d_{R,F}^{Z,Y}(\bar{T}_R^Y) \\ \text{if a definite rising transition at Y, or no possible} \\ \text{rising transition at X.} \\ A_{R,L}^X + d_{R,F}^{Z,X}(\bar{T}_R^X) \\ \text{symmetric to the case above.} \end{cases} \quad (13)$$

For $T_{F,S}^Z$ and $T_{F,L}^Z$, the three cases are analyzed similarly as that for $A_{F,L}^Z$.

6.5 Comparison with STA

A comparison between STA and TA-PSV is shown in Table 2.

For a delay model to be compatible with TA-PSV, worst case corners for partial specified input logic values with timing ranges need to be identifiable. One sufficient condition is that all timing functions be monotonic or bi-tonic with respect to each input variable.

Table 2. STA vs. TA-PSV.

Static Timing Analysis (STA)	Timing Analysis for Partially Specified Vectors (TA-PSV)
Vector independent	Takes advantage of logic/timing information - timing ranges shrink as more values are specified
Rising and falling transitions on each line is always possible	Logic values on lines may exclude or force other transitions
A special case of TA-PSV	A generalization of STA

7. Applications of TA-PSV

7.1 Timing Oriented ATPG

A “crosstalk delay fault” is used to illustrate how our delay model, along with STA and TA-PSV, can be used in a timing oriented ATPG system.

During test generation, a crosstalk target consisting of a victim line (V) and an affecting line (A) are first selected (see Figure 14). If (a) there exists an input vector pair that causes opposite transitions on each of these two lines, (b) the skew between these two transitions is small, e.g., within one or two gate delays, and (c) there is a significant coupling capacitance between these two lines, then both signals will experience a slowdown effect. That is, their transition times and effective arrival times will increase.

To excite a crosstalk effect between a pair of lines, a test needs to satisfy both logic ((a) above) and timing ((b) above) constraints. Then traditional *two-vector ATPG methods* and *TA-PSV* can be used to determine if there exists a test that can excite the fault effects and propagate its effect to a primary output or a flip-flop with a setup time violation.

During test generation, logic constraints are processed using techniques such as forward/backward implications [17]. In time-oriented ATPG, timing constraints can also be processed to further reduce the search space.

Our timing-oriented ATPG [9] contains (a) a delay model including timing ranges, (b) fault excitation conditions at the fault sites, and propagation conditions at fault-free sites, (c) a search engine to implicitly enumerate the logic search space, and (d) ITR (incremental timing refinement, an event-driven version of TA-PSV) that computes accurate timing ranges when logic values or timing ranges at lines are further specified. Timing violation should be checked after new timing ranges are calculated.

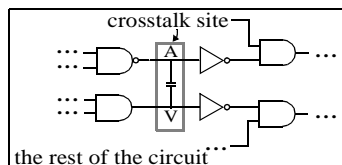


Fig. 14. A view of the circuit during crosstalk test generation.

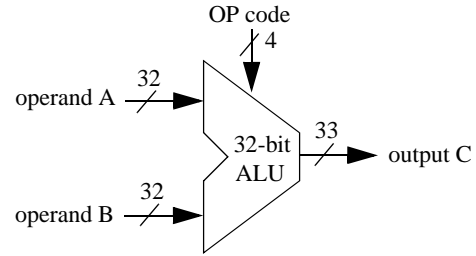


Fig. 15. An ALU example.

7.2 Timing Validation

There are several other applications for using the proposed delay model and TA-PSV. For example, during timing validation of a circuit one may pre-specify some input values and desire to know the corresponding worst case timing windows.

Figure 15 shows the diagram of an ALU with two 32-bit operands and ten possible operations (e.g. ADD, SUB, reverse SUB, AND, OR, XOR, shift left, shift right, pass A, pass B). The ten operations, represented by decimal 0 to 9, are encoded as four-bit operation codes (OP codes) to control the ALU.

STA can be used to identify the maximal delay of this ALU. If the longest true path is excitable only by the OP code that is not used, then STA would report an overly pessimistic value for delay. This unused long path need not to be optimized during the design cycle, since it will not affect the actual performance of this ALU. But STA cannot calculate the output timing with all unused OP codes excluded.

TA-PSV can employ a minimal Boolean cover of the input space of the actual OP codes. The union of timing ranges computed by this cover represents a more realistic estimation of the output timing ranges.

8. Experimental Results

As a simple example, TA-PSV was performed on benchmark C17 (Figure 16) to observe how timing ranges shrink, where, in each step an input value becomes more specified. The result is shown in Table 3. Here **V** means logic value; **FS** is the arrival time for the earliest falling transition; **FL** is the arrival time for latest falling transition; **RS** is the arrival time for earliest rising transition; and **RL** is the arrival time for latest rising transition. The time unit is 0.01ns. All inputs are initially unspecified. At this point, the timing information is the same as that in static timing analysis. At each

Table 3. An TA-PSV example on C17.

0	1	2	3	4	5	6				7				8				9				10								
V	V	V	V	V	V	V	FS	FL	RS	RL	V	FS	FL	RS	RL	V	FS	FL	RS	RL	V	FS	FL	RS	RL	V	FS	FL	RS	RL
xx	xx	xx	xx	xx	xx	xx	7	7	13	15	xx	7	22	13	26	xx	7	22	13	26	xx	20	33	23	39	xx	20	33	23	41
00	xx	xx	xx	xx	11	xx	7	7	13	15	xx	7	22	13	26	xx	7	22	13	26	xx	20	33	23	39	xx	20	33	23	41
00	11	xx	xx	xx	11	xx	7	7	13	13	xx	7	20	13	26	xx	7	20	13	26	xx	20	33	23	36	xx	20	33	23	39
00	11	10	xx	xx	11	01	-	-	13	13	1x	20	20	-	-	1x	20	20	-	-	0x	-	-	36	36	0x	-	-	37	39
00	11	10	01	xx	11	01	-	-	13	13	10	20	20	-	-	1x	20	20	-	-	01	-	-	36	36	01	-	-	37	39
00	11	10	01	x0	11	01	-	-	13	13	10	20	20	-	-	11	-	-	-	-	01	-	-	36	36	01	-	-	39	39

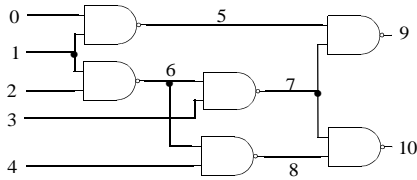


Fig. 16. Benchmark circuit C17.

step, going from row to row, the value at one primary input becomes more specific and the timing ranges are updated. Timing windows naturally shrink. If the logic value on a line implies no rising (falling) transition, then the timing information for this transition becomes meaningless, and is denoted by ‘-’ in the table. After the values of the first three inputs are fully specified, the arrival time of a rising transition on output 9 has shrunk from (23, 39) to (36, 36), a fixed value.

TA-PSV was also performed on ISCAS85 circuits by randomly specifying values at inputs. The fractional amount of total timing range shrinkage for all lines in a circuit as a function of the percent of the inputs specified is shown in Figure 17. These results demonstrate that timing ranges shrink appreciably even when only a small fraction of the values are specified. This is why TA-PSV is very helpful in reducing the search required in timing oriented test generation [18].

Performance of a crosstalk test generator [18] with (w/) and without (w/o) TA-PSV is shown in Table 4. A target is aborted if the number of backtracks exceeds 1000. With TA-PSV, the average ATPG efficiency increased from 40% to 83%, because many objectives that do not meet the timing criteria are identified early in the test generation process and lead to a backtrack. Hence the search space is reduced. Using TA-PSV, CPU time increases by a factor of from 10 to 20. In Table 5 test generation results with and without TA-PSV are shown for comparable run times. As can be seen, by increasing test generation time of an ATPG

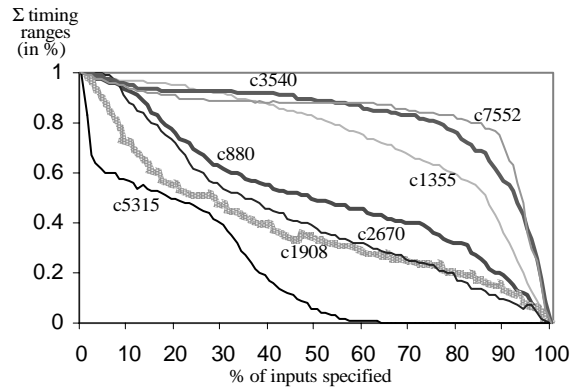


Fig. 17. Total timing range shrink as more inputs are specified.

without TA-PSV by a factor of 10 only improves the efficiency by a very small amount. This clearly shows that TA-PSV is essential for efficiency improvement of timing oriented ATPG.

Compared with traditional ATPGs that handle only logic conditions, timing oriented ATPGs require larger runtime due to the computation of timing information. However, it is still computationally feasible to carry out timing oriented ATPG on extremely large circuits.

The reason is that for single stuck-at fault, test generation may deal with more than 10^6 faults. Normally there are less than 10^3 crosstalk sites to be addressed. Another application where a timing oriented ATPG system is of value is in the generation of vectors for dynamic timing analysis. Here one may desire to generate only one test that produces the maximum delay through a block of combinational logic, given a timing model for the gates and interconnect.

Hence even if the run time for each fault is three order larger than that for single stuck-at fault, our ATPG is still practical due to much fewer fault targets in these applications. We are developing hybrid logic and timing implication techniques that are not reported

Table 4. Comparison of ATPG efficiency for crosstalk delay with and without TA-PSV for the same backtrack number.

Circuit Name	ATPG Efficiency (%)		TG Time (s)	
	w/o TA-PSV	w/ TA-PSV	w/o TA-PSV	w/ TA-PSV
C432	37	83	138	1167
C880	49	85	112	1664
C1355	28	77	423	3403
C1908	43	85	354	2555
C2670	43	85	277	4870
C3540	33	76	539	4661
C5315	49	85	441	7745
C7552	35	86	627	8651
AVE.	39.63	82.75	406	4406

Table 5. Comparison of ATPG efficiency for crosstalk delay with and without TA-PSV for similar run times.

Circuit Name	ATPG Efficiency (%)		TG Time (s)	
	w/o TA-PSV	w/ TA-PSV	w/o TA-PSV	w/ TA-PSV
C432	38	83	2199	1167
C880	50	85	1771	1664
C1355	29	77	3409	3403
C1908	45	85	8476	2555
C2670	43	85	5620	4870
C3540	33	76	6013	4661
C5315	49	85	7515	7745
C7552	35	86	10032	8651
AVE.	40.25	82.75	5628	4406

here. That will further speed up the above test generation process.

9. Conclusions

We propose a new concept - timing analysis for partially specified vectors (TA-PSV) - that enables the computation of tight timing windows when input vectors are partially specified. Using a more accurate delay model, we propose a method to identify the worst case corners and compute more accurate timing ranges.

We describe a systematic approach to determine the main characteristic used in TA-PSV, namely states of transitions. We reduce the numbers of states by collapsing together states that have the same worst case corners. We extend our TA-PSV method to handle gates with more than two inputs.

We show the differences between STA and TA-PSV. Worst case corners need to be identifiable for this new delay model to enable the use of TA-PSV in conjunction with this model. A sufficient condition for adopting our method for worst case corner identification is reported.

TA-PSV can be used to bound timing ranges and hence reduce the search space for timing-oriented test generation. TA-PSV can also provide more accurate timing ranges during timing validation. We show how TA-PSV significantly reduces timing windows on the ISCAS85 benchmark circuits. We also demonstrate how ATPG efficiency is significantly improved by TA-PSV.

Appendix A

Calculation of required times in static timing analysis.

$$\begin{aligned}
 Q_{R,S}^X &= Q_{F,S}^Z - \mathbf{d}^Z_{X_F}(\bar{T}_R^X), \\
 \text{where } \bar{T}_R^X &= \begin{cases} T_{R,\max}^X, & \text{if } T_{R,\max}^X \in (T_{R,S}^X, T_{R,L}^X); \\ T_{R,S}^X, & \text{else if } \mathbf{d}^Z_{X_F}(T_{R,S}^X) > \mathbf{d}^Z_{X_F}(T_{R,L}^X); \\ T_{R,L}^X, & \text{otherwise.} \end{cases} \\
 T_{R,\max}^X &: T_R^X \text{ that maximizes } \mathbf{d}^Z_{X_F}(T_R^X). \\
 Q_{R,L}^X &= Q_{F,L}^Z - \min[\mathbf{d}^Z_{X_F}(T_{R,S}^X), \mathbf{d}^Z_{X_F}(T_{R,L}^X)]. \\
 Q_{F,S}^X &= Q_{R,S}^Z - \mathbf{d}^Z_{X_R}(\bar{T}_F^X), \\
 \text{where } \bar{T}_F^X &= \begin{cases} T_{F,\max}^X, & \text{if } T_{F,\max}^X \in (T_{F,S}^X, T_{F,L}^X); \\ T_{F,S}^X, & \text{else if } \mathbf{d}^Z_{X_R}(T_{F,S}^X) > \mathbf{d}^Z_{X_R}(T_{F,L}^X); \\ T_{F,L}^X, & \text{otherwise.} \end{cases} \\
 T_{F,\max}^X &: T_F^X \text{ that maximizes } \mathbf{d}^Z_{X_R}(T_F^X). \\
 Q_{F,L}^X &= \begin{cases} Q_{R,L}^Z - \min_{\beta, \gamma \in \{S, L\}} [\mathbf{d}^Z_{R}(T_{F,\beta}^X, T_{F,\gamma}^Y, \bar{\delta}^{X,Y})] \\ \text{if } (A_{F,S}^X + \bar{\delta}^{X,Y}, A_{F,L}^X + \bar{\delta}^{X,Y}) \\ \quad \cap (A_{F,S}^Y, A_{F,L}^Y) \neq \emptyset; \\ Q_{R,L}^Z - \min_{\beta, \gamma \in \{S, L\}} [\mathbf{d}^Z_{R}(T_{F,\beta}^X, T_{F,\gamma}^Y, A_{F,S}^Y - A_{F,L}^X), \\ \quad \mathbf{d}^Z_{R}(T_{F,\beta}^X, T_{F,\gamma}^Y, A_{F,L}^Y - A_{F,S}^X)] \\ \text{otherwise.} \\ \bar{\delta}^{X,Y}: \delta^{X,Y} \text{ that minimizes } \mathbf{d}^Z_{R}(T_{F,\beta}^X, T_{F,\gamma}^Y, \delta^{X,Y}); \\ \quad \forall \beta, \gamma \in \{S, L\}. \end{cases}
 \end{aligned}$$

References

1. R.B. Hitchcock, "Timing Verification and Timing Analysis Program", Proc. of ACM Design Automation Conf. 1982, pp. 594-604.
2. C. Visweswariah and R.A. Rohrer, "Piecewise Approximate Circuit Simulation", IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 10, pp. 861-870, July 1991.
3. Y.H. Shih, Y. Leblebici, and S.M. Kang, "ILLIADS: A Fast Timing and Reliability Simulator for Digital MOS Circuits", IEEE

Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 12, pp. 1387-1402, Sept. 1993.

4. L.W. Nagel, "SPICE2, A Computer Program to Simulate Semiconductor Circuits", Memo UCB / ERL M520, Univ. Cal., Berkeley, May 1975.

5. W.Y. Chen, S.K. Gupta, and M.A. Breuer, "Test Generation in VLSI Circuits for Crosstalk Noise", Proc. of IEEE Int'l Test Conf., pp. 641-650, 1998.

6. W.Y. Chen, S.K. Gupta, and M.A. Breuer, "Test Generation for Crosstalk-induced Delay in Integrated Circuits", Proc. of IEEE Int'l Test Conf., pp. 191-200, 1999.

7. P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits", IEEE Trans. Computers, vol. 30, pp. 215-222, March, 1981.

8. L.C. Chen, S.K. Gupta, and M.A. Breuer, "Incremental Timing Analysis - Intuition and Implementation", SRC TECHCON 2000, September 2000.

9. L.C. Chen, S.K. Gupta, and M.A. Breuer, "A New Framework for Static Timing Analysis, Incremental Timing Analysis, and Timing Simulation", Proc. of IEEE Asian Test Symp., pp.102-107, 2000.

10. L.C. Chen, S.K. Gupta, and M.A. Breuer, "A New Gate Delay Model for Simultaneous Switching and Its Applications", Proc. of ACM Design Automation Conf., pp. 289-294, 2001.

11. W.Y. Chen, S.K. Gupta, and M.A. Breuer, "Analytic Models for Crosstalk Delay and Pulse Analysis under Non-ideal Inputs", Proc. of IEEE Int'l Test Conf., pp. 809-818, 1997.

12. V. Chandramouli and K.A. Sakallah, "Modeling the Effects of Temporal Proximity of Input Transitions on Gate Propagation Delay and Transition Time", Proc. of ACM Design Automation Conf., pp. 617-622, 1996.

13. Y.H. Jun, K. Jun, and S.B. Park, "An Accurate and Efficient Delay Time Modeling for MOS Logic Circuits Using Polynomial Approximation", IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 8, pp. 1027-1032, Sept. 1989.

14. A. Nabavi-Lishi and N.C. Rumin, "Inverter Models of CMOS Gates for Supply Current and Delay Evaluation", IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 13, pp. 1271-1279, Oct. 1994.

15. IEEE DASC Standard Delay Format (SDF) - web page <http://vhdl.org/vi/sdf/>.

16. A. Chatzigeorgiou, S. Nikolaidis, and I. Tsoukalas, "A Modeling Technique for CMOS Gates", IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 18, pp. 557-575, May 1999.

17. M. Abramovici, M.A. Breuer, and A.D. Friedman, *Digital Systems Testing and Testable Design*, IEEE Press, 1995.

18. W.Y. Chen, "Test Generation for Crosstalk Noise in VLSI Circuits", Ph.D. Dissertation, University of Southern California, EE-System Dept., 2000.

Liang-Chi Chen received his Bachelor degree in Electrical Engineering from National Cheng-Kung University, Tainan, Taiwan, in 1990 and obtained MS degree in Electrical and Computer Engineering from the State University of New York at Buffalo in 1993. He is current pursuing the Ph.D. degree in the Department of Electrical Engineering - Systems at the University of Southern California, Los Angeles. His research interests include timing modeling and validation, delay testing and diagnosis, at-speed

testing, and defect based testing.

Sandeep Gupta received his Bachelors degree in Electrical Engineering from the Indian Institute of Technology, Kharagpur, India, in 1985 and obtained MS and PhD degrees in Electrical and Computer Engineering from the University of Massachusetts at Amherst in 1989 and 1991. He is an associate professor in the Department of Electrical Engineering-Systems at the University of Southern California, Los Angeles. He is also an associate editor of the *IEEE Transactions on Computers*. His research interests are in the area of VLSI testing and design. He is currently involved in projects on test and validation of deep submicron circuits, testing multicore systems-on-silicon, and delay testing and diagnosis of digital circuits. He is also involved in a project on testing and verification of network protocols. He is a recipient of the US National Science Foundation's Research Initiation Award (1992) and CAREER Award (1995). He is also a recipient of the Northrop Grumman Assistant Professorship (1995) and Zumberge Fellowship (1996) at the University of Southern California. He also received the Honorable Mention Award from the International Test Conference in 1997, and the co-author of the best paper at the 2000 Asian Test Symposium. He is a member of the IEEE Computer Society.

Melvin A. Breuer received his Ph.D. in Electrical Engineering from the University of California, Berkeley, is currently a Professor of both Electrical Engineering and Computer Science at the University of Southern California, Los Angeles, California, and is the Charles Lee Powell Professor of Electrical Engineering and Computer Science. He was Chairman of the Department of Electrical Engineering-Systems from 1991-1994, and is currently Chair once again. He was Chair of the Faculty of the School of Engineering, USC, for the 1997-98 academic year. His main interests are in the area of computer-aided design of digital systems, design-for-test and built-in self-test, and VLSI circuits.

Dr. Breuer is the editor and co-author of *Design Automation of Digital Systems: Theory and Techniques* (Prentice-Hall); editor of *Digital Systems Design Automation: Languages, Simulation and Data Base* (Computer Science Press); co-author of *Diagnosis and Reliable Design of Digital Systems* (Computer Science Press); co-editor of *Computer Hardware Description Languages and their Applications* (North-Holland); co-editor and contributor to *Knowledge Based Systems for Test and Diagnosis* (North-Holland); and co-author of *Digital System Testing and Testable Design* (Computer Science Press 1990 and reprinted in 1995 by the IEEE Press). He has published over 200 technical papers and was formerly the editor-in-chief of the *Journal of Design Automation and Fault Tolerant Computing*, on the editorial board of the *Journal of Electronic Testing*, the co-editor of the *Journal of Digital Systems*, and the Program Chairman of the Fifth International IFIP Conference on Computer Hardware Description Languages and Their Applications. He was co-author of a paper that received an honorable mention award at the 1997 International Test Conference, and the co-author of the best paper at the 2000 Asian Test Symposium. He is a Fellow of the IEEE; was a Fullbright-Hays scholar (1972); received the 1991 Associates Award for Creativity in Research and Scholarship from the University of Southern California, the 1991 USC School of Engineering Award for Exceptional Service, the IEEE Computer Society's 1993 Taylor L. Booth Education Award, and the first (2000) Engineering Faculty

Council Award for Outstanding Meritorious Service to the USC School of Engineering. He was the keynote speaker at the Fourth Multimedia Technology and Applications Symposium, 1999, and the Ninth Asian Test Symposium, 2000.