

A New Framework for Static Timing Analysis, Incremental Timing Refinement, and Timing Simulation *

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

Abstract

In this paper we present a framework that enables the computation of tight ranges of signal arrival, transition, and required times for rising and falling transitions at each circuit line, given an input sequence consisting of two partially specified vectors. At one extreme, when the vectors are completely unspecified, this framework becomes identical to static timing analysis (STA). At the other extreme, when the vectors are completely specified, this framework performs timing simulation (TS). Our key motivation for developing this framework was to reduce the amount of search required by a test generator that uses timing information. During test generation for a target fault, values are specified incrementally and this framework enables refinement of timing windows. We demonstrate that this approach significantly improves test generation efficiency. In this mode, the ATPG is said to be performing incremental timing refinement (ITR).

1. Introduction

Static timing analysis (STA) [1] as well as timing simulation (TS) are both integral steps in validating the timing of a circuit. In the former case, for the universe comprised of all possible sequences of two vectors, the minimum and maximum values, called ranges, of signal transition, arrival, and required times are computed for each circuit line. For timing simulation, computation is performed for a given sequence of two completely specified vectors and, if hazards are ignored, each range becomes a point.

An intermediate situation arises for timing-oriented test generation [2][3][4]. In this case, the timing of events must satisfy certain conditions for a sequence of two (in some cases, three) vectors to be a test for a target. Due to this reason, the timing of transitions is considered as an integral part of test generation.

Test generation for a target fault begins with a sequence of completely unspecified vectors. At this stage STA can be performed to determine the timing ranges of signal arrival and transition times. As test generation proceeds, specific logic values are successively assigned to circuit inputs. Since the assignment of specific logic values reduces the set of possible sequences that is being considered, a timing analysis method taking advantage of the constraints implied by these assignments can help narrow

the timing ranges. We call such a generalized timing analysis approach *incremental timing refinement (ITR)*.

In the following, we present a new framework that encompasses STA, ITR, and TS. The development of such a framework needs a delay model as its basis. Here a pin-to-pin delay model is used (and expressed in the standard delay format (SDF) [5]). 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 for a timing range at its output. We will present a delay model that has these characteristics. This is followed by the description of a STA using the delay model. Then we demonstrate how ITR can be used to reduce timing ranges when some input values are specified. Implementation of ITR shows how much ITR improves efficiency of timing-based test generation.

In Section 2, we illustrate the usefulness of ITR by an example. In Section 3, a pin-to-pin delay model is proposed. In Section 4, static timing analysis is developed for the proposed model. In Section 5, fundamental concepts of ITR are presented. Experimental results are shown in Section 6.

2. Motivation for Incremental Timing Refinement

Based on a delay model, *static timing analysis (STA)* [1] provides vector-independent min-max range of arrival and required time for rising and falling transitions on each line in a circuit. Since STA considers the input vectors as being fully unspecified, the timing ranges should be narrower when the input vectors are partially/fully specified. *Incremental timing refinement (ITR)* [4] starts with the timing ranges computed by STA and refines the ranges as input vectors become more specified.

To perform STA on a delay mode, we need to (1) handle both input and output timing as ranges, and (2) identify the worst case corners over all possible input combinations, so, in forward calculations, the output ranges of a gate can be calculated based on the gate's input ranges. Required time is calculated backwards from a gate's output to each of its inputs using required times at the output and pin-to-pin gate delays.

During test generation, logic constraints are processed using techniques such as forward and backward implications [6] and recursive learning [7] in order to reduce the search space. In time-based ATPG, timing constraints can also be processed to further reduce the search space.

*. This work was supported in part by the Semiconductor Research Corp. under contract No. 98-TJ-646, and by Intel Corporation.

Here ATPG for crosstalk faults is used as an example. During the test generation, a crosstalk target consisting of a victim line and an affecting line are first selected. If (1) there exists an input vector pair that causes opposite rising and falling transitions on each of these two lines, (2) the skew between these two transitions is small, e.g., within one or two gate delays, and (3) there is a significant coupling capacitance between these two lines, then both signals will experience a slowdown effect. That is, their transition times will increase and their effective arrival times will also increase.

To excite a crosstalk effect between a pair of lines, a test needs to satisfy both logic ((1) above) and timing ((2) above) constraints. Timing based test generation with ITR is performed on the example circuit in Figure 1 to illustrate the weakness of an ATPG that employs STA and timing simulation for fully specified values without ITR. (The details of the delay model can be found in [9]).

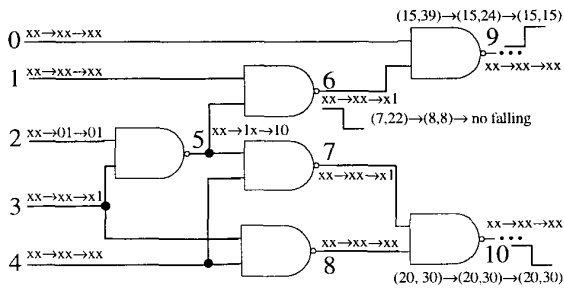


FIGURE 1. An example for crosstalk test generation.

Consider the crosstalk target consisting of signal lines 9 and 10. Assume the target transitions are rising at line 9 and falling at line 10. Static timing analysis shows that the arrival timing ranges of the transitions at lines 9 and 10 are (15, 39) and (20, 30) respectively. The overlap of their timing ranges suggests that test generation for this crosstalk target may be fruitful.

We used PODEM [8] to perform test generation. All lines are initially xx at Step 0. At this time, all timing information is obtained from static timing analysis. At each test generation (TG) step, one input will be assigned a more specified two-frame value. The logic value for each TG step is shown at each line. Timing information is shown only for the two target transitions at line 9 and 10, and the falling transition at line 6, which affects the rising transition at line 9. To obtain a falling transition at line 10, in Step 1, we backtrack to line 2 and assign value 01 at the line. As a consequence of the above assignment, the logic value at line 5 changes to 1x. Hence, no rising transition can occur at line 5. Now only a rising transition at line 1 may cause the falling transition at line 6. So the timing range for a falling transition at line 6 shrinks from (7, 22) to (8, 8). This change at line 6 reduce the timing range for a rising transition at line 9 from (15, 39) to (15, 24). In Step 2, line 3 is set to x1 to justify the value on line 10. This assignment causes the timing range at line 9 to shrink to (15, 15) since no falling transition at line 6 exists (with these assignment, only a transition at line 0 can cause a rising transition at line 9). Now the minimal skew between these two lines (five) is too large to excite a crosstalk slowdown effect. Hence we quit searching the subspace described by the above two assignments and backtrack, since no pair of vectors in this subspace can cause significant slowdown.

Without ITR, this timing contradiction will not be found until the logic and timing values at lines 0, 6, 7, and 8 are fully specified, which would have required assignment of values to all five input lines. Then the depth of the decision tree will be increased from two to five. The ATPG run time, which is dominated by the size of the search tree can grow exponentially with the depth of the decision tree. Hence, the ability to identify contradictions between logic and timing constraints as early as possible may increase ATPG efficiency.

A traditional ATPG cannot find such contradictions early since it does not fully exploit timing information. As logic values become specified at additional primary inputs, the timing ranges at some lines may shrink. The proposed incremental timing analysis (ITR) can be used to find the tightest bound for logic/timing information. It can therefore identify contradictions as early as possible.

As specific logic values are assigned to circuit inputs during test generation, the timing ranges for each line may shrink. The ranges may even disappear, if no rising/falling transitions can occur at this line. A gate's output timing ranges shrink due to (1) the shrinkage of an input's timing range (which causes the timing range shrinkage at line 9 in Step 1), or (2) the further specification of some input's logic values (which causes the timing range shrinkage at line 9 in Step 2). New timing information for case (1) 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 (2) to extend STA to ITR.

3. Delay Model

A NAND gate with output Z and two inputs X and Y is used as an example to illustrate the following definitions. Here Z represents a gate as well as its output. The **controlling value** of gate Z is 0 and its **non-controlling value** is 1. (V_1, V_2) represents two values that appears at a line in sequence. The **to-controlling transition** at an input of Z is a sequence of values (1, 0). If to-controlling transitions occur at one or more inputs, and the gate's non-controlling value is applied to its remaining inputs, then the transition at the gate output is a **to-controlling response** (0, 1). **To-non-controlling transitions and responses** 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 to go from 0.1V_{dd} to 0.9V_{dd} for a rising transition (R), 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 reaches 0.5 V_{dd}.

3.1 Timing Functions (for a two-input NAND gate)

During test generation, all circuit parameters (e.g., device sizes and loads) remain fixed. In contrast, timing parameters (e.g., arrival times, transition times) change from vector to vector. 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 then computed using the input arrival times and the gate's delay.

We only consider the cases where each input of a gate has either a non-controlling value or a transition to the same value as other inputs. The reason is that the non-trivial output responses

caused by input transitions to opposite values are either (1) two separate transitions in opposite directions that can be processed separately using the method, or (2) glitches where gate delay and output transition time are not the parameters of concern [3].

We use a pin-to-pin delay model, where the **pin-to-pin delay** from X to Z is the gate delay of Z when Y is steady at 1 (the gate's non-controlling value) and a transition is applied on X. The fall delay function and the fall transition time function (from X to Z) are represented by $d^{Z,X}_F(T^{X_R})$ and $t^{Z,X}_F(T^{X_R})$ as shown in Figure 2. Rising delay and transition time functions are defined similarly. When both inputs have rising (to-non-controlling) transitions, the output arrival time A^Z_F is defined as $\max\{A^{X_R} + d^{Z,X}_F(T^{X_R}), A^{Y_R} + d^{Z,Y}_F(T^{Y_R})\}$, because the output voltage starts to change after the later input transition changes voltage. When both inputs have falling (to-controlling) transitions, the output arrival time A^Z_R is defined as $\min\{A^{X_F} + d^{Z,X}_R(T^{X_F}), A^{Y_F} + d^{Z,Y}_R(T^{Y_F})\}$.

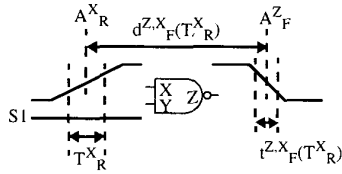


FIGURE 2. Fall delay function.

3.2 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 3.

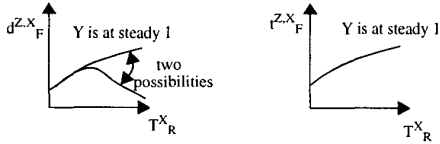


FIGURE 3. Timing functions vs. input variables.

The gate delay as a function of $T^{X_{tr}}$ may be either (1) monotonically increasing or (2) bi-tonic (monotonically increasing and then monotonically decreasing in this case). In Case (2), the pin-to-pin delay may become negative for large $T^{X_{tr}}$. This bi-tonicity is due to the fact that the input transition starts to drive the output voltage up or 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 $T^{X_{tr}} - d^{Z,X}_F$ curves take. Below we consider only Case (2) since Case (1) is a special case of (2) with the curve's peak at infinity. Output transition time will always increase as $T^{X_{tr}}$ increases.

4. Static Timing Analysis

The three timing values considered in STA are arrival times (A), transition times (T), and required times (Q). Assume that

input transitions occur at time 0. An 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 allowed 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 the minimum and maximum delay values over all possible pairs of vectors. In timing analysis (Figure 4) arrival times (A) and transition times (T) at a gate's output are calculated based on these values at its inputs. These values are computed via a forward traversal starting at the primary inputs. Required times (Q) are computed via a backward traversal starting at 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 and no delay error can be produced using the current input values. Delay transfer functions for forward and backward calculations in timing analysis are defined in the proposed model. 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. In our current delay model which ignores pulses, if all input values are specified, timing ranges become points.

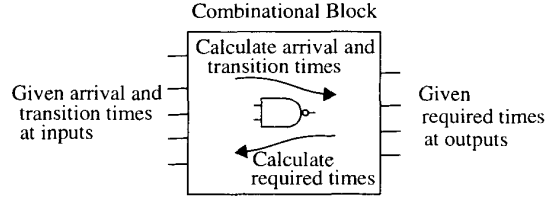
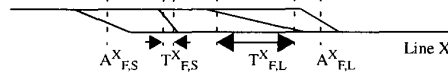


FIGURE 4. Overall structure of timing analysis.

4.1 Time Information

In our min-max range representation, the timing windows in [4] are used and shown in Figure 5. 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 are represented as $A^{X_{FS}}$ ($A^{X_{RL}}$). Transition and required times are represented similarly.

- Arrival time (A) and transition time (T) -- Fall Smallest/Largest



- Required arrival time (Q) for timing analysis-- Fall Smallest/Largest

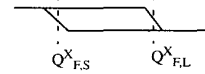


FIGURE 5. Timing information used in our method.

4.2 Worst Case Analysis

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 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.

In Figure 6 we illustrate how A_F^Z is computed. Based only on logic values, there are three ways to justify a falling transition at the output. These three cases are analyzed separately and the results are combined to find the worst case.

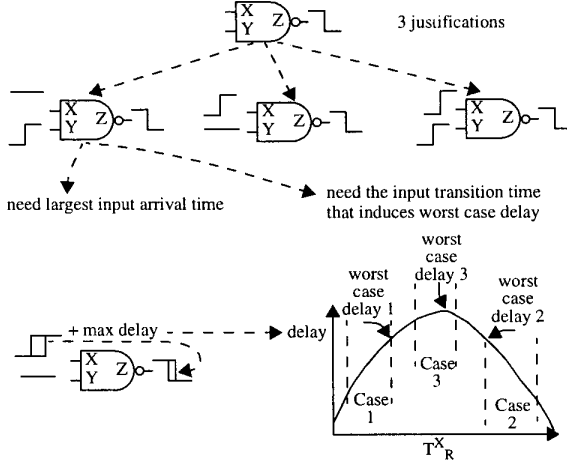


FIGURE 6. Worst case analysis for latest output arrival time.

A_F^Z is maximum when both the input arrival times and the gate delay are maximal. The maximal gate delay may occur when the input transition times are either (1) maximal, (2) minimal, or (3) at some values in between. These three scenarios correspond, respectively, to the three cases shown in Figure 6, where the min-max range is on the left of the peak (Case 1), right of the peak (Case 2), and containing the peak (Case 3), respectively.

The arrival times and the transition times for an output falling transition are:

$$A_{FS}^Z = \min [A_{RS}^X + \min \{d_{F,R,S}^{Z,X}(T_{R,S}^X), d_{F,R,L}^{Z,X}(T_{R,L}^X)\}, \\ A_{RL}^Y + \min \{d_{F,R,S}^{Z,Y}(T_{R,S}^Y), d_{F,R,L}^{Z,Y}(T_{R,L}^Y)\}].$$

$$A_{FL}^Z = \max [A_{RL}^X + d_{F,R}^{Z,X}(T_{R'}^X), A_{RL}^Y + d_{F,R}^{Z,Y}(T_{R'}^Y)]$$

$$\text{where } T_{R'}^X = T_{R,\max}^X, \text{ if } T_{R,\max}^X \in (T_{R,S}^X, T_{R,L}^X); \\ = T_{R,S}^X, \text{ else if } d_{F,R,S}^{Z,X}(T_{R,S}^X) > d_{F,R,L}^{Z,X}(T_{R,L}^X); \\ = T_{R,L}^X, \text{ otherwise.}$$

Here, $T_{R,\max}^X$ is the value of T_{R}^X that maximizes $d_{F,R}^{Z,X}(T_{R}^X)$. $T_{R,\max}^Y$ is defined similarly. The transition time can be computed as

$$T_{FS}^Z = \min \{t_{F,R,S}^{Z,X}(T_{R,S}^X), t_{F,R,S}^{Z,Y}(T_{R,S}^Y)\}.$$

$$T_{FL}^Z = \max \{t_{F,R,L}^{Z,X}(T_{R,L}^X), t_{F,R,L}^{Z,Y}(T_{R,L}^Y)\}.$$

Given the required arrival time (Q) at a gate's output, and the minimal/maximal arrival and transition times at its inputs, we calculate Q for each input. There is no required *transition time* in our analysis. The maximal required (arrival) time at an input equals the associated maximal required time at the output minus the minimal delay from this input to the output. The calculations of A, T, and Q for output falling transition are similar.

5. Incremental Timing Refinement

STA provides vector-independent min-max timing ranges for rising and falling transitions at each line. It can be used to provide initial timing information for test generation since a test generator starts with all unknown values. But as more specific values are assigned during test generation process, the min-max ranges become narrower due to (1) the increased specificity of the input vectors, and (2) the logic and timing dependencies between lines (ignored in STA) become apparent as more input values are specified. Thus, worst case corners considered during STA may become obviously impossible to attain after some input values are specified. We have developed a timing mechanism called **incremental timing refinement (ITR)** for identifying new worst case corners and computing new A, T, and Q windows at these corners.

5.1 Logic Value System

For timing simulation and test generation, two-pattern tests are needed to create transitions carrying timing information. In addition to the timing information, a two-frame value (v_1, v_2) is used to record the logic information for each line. A value in each frame can be 0, 1, or x, where x represents an unspecified value for a primary input, and an unknown value for any other line. The possible refinements of logic values are shown in Figure 7. 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 ([6]) to two time frames.

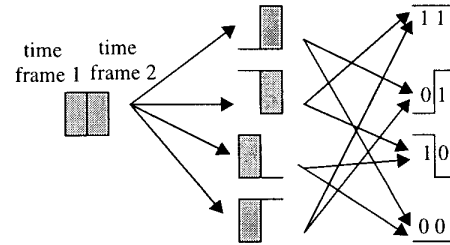


FIGURE 7. Refinement of logic values (■ : X, □ : 0, and ▭ : 1).

Among the nine logic values, {00, 01, 0x, 10, 11, 1x, x0, x1, xx}, for two-frame logic, 01 specifies a definite rising transition. 0x, x1, and xx may be refined to 01, so they potentially support rising transitions. Other logic values may not be refined to 01, and hence are incompatible with a rising transition. The **state** of a line with respect to a rising (falling) transition is defined to be one of the three cases below: (1) potentially supports a rising transition, (2) definitely supports a rising transition, and (3) definitely does not support a rising transition. As a logic value is refined, its state may change from (1) to (2) or (3). No other state transitions exist during further specification of logic values.

5.2 Timing Refinement

An example of incremental timing refinement for an output falling transition is shown in Figure 8. Initially both inputs X and Y are unspecified, and the ranges are the same as those in STA, where the minimal output arrival time may be caused by the earlier of the rising transitions at X or Y, and the maximal output arrival time may be 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.

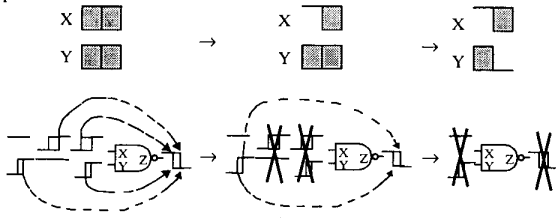


FIGURE 8. An example of timing refinement.

In incremental timing refinement, the arrival times and transition times of an output falling transition are calculated as below. Logic implication will be performed first. Incremental timing refinement will be performed for a transition at a line only when this transition is definitely/potentially supported at this line.

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

$$A_{FL}^Z = \begin{cases} \max \{ A_{R,L}^X + d_{F,R,L}^{Z,X}(T_{R,L}^X), A_{R,L}^Y + d_{F,R,L}^{Z,Y}(T_{R,L}^Y) \} \\ \text{if rising transitions may occur at both X and Y} \\ A_{R,L}^Y + d_{F,R,L}^{Z,Y}(T_{R,L}^Y) \\ \text{if a definite rising transition at Y, or no possible} \\ \text{rising transition at X.} \\ A_{R,L}^X + d_{F,R,L}^{Z,X}(T_{R,L}^X) \\ \text{symmetric to the case above.} \end{cases}$$

Here $T_{R,L}^X$, $T_{R,L}^Y$ are defined in Section 4.2.

For A_{FS}^Z and A_{FL}^Z , the three cases are analyzed similarly as that for A_{FS}^Z .

A complete description of ITR and its extension to gates with more than two inputs are described in [9].

A comparison between STA and ITR is shown in Table 1.

TABLE 1. STA vs. ITR

| Static Timing Analysis (STA) | Incremental Timing Refinement (ITR) |
|--|---|
| 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 |
| Completed in one iteration | Refinement is needed as more values are specified |
| A special case of ITR | A generalization of STA |

For a model to be compatible with our STA/ITR formulation in [9], it is a sufficient condition that all timing functions be monotonic or bi-tonic with respect to each input variable.

6. Experimental Results

ITR was performed on benchmark C17 (Figure 9) to observe how timing ranges shrink, where, in each step an input value becomes more specified. The result is shown in Table 2. Here V

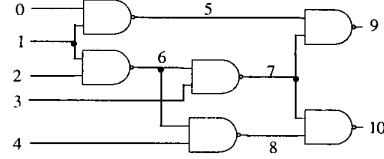


FIGURE 9. Benchmark circuit C17.

TABLE 2. An ITR example on C17.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|----|----|----|----|----|----|----|----|----|----|----|
| v | v | v | v | v | v | FS | FL | RS | RL | v |
| xx | xx | xx | xx | xx | xx | 7 | 7 | 13 | 15 | xx |
| 00 | xx | xx | xx | xx | 11 | xx | 7 | 7 | 13 | 15 |
| 00 | 11 | xx | xx | xx | 11 | xx | 7 | 7 | 13 | 15 |
| 00 | 11 | 10 | xx | xx | 11 | 01 | - | - | 13 | 13 |
| 00 | 11 | 10 | 01 | xx | 11 | 01 | - | - | 13 | 13 |
| 00 | 11 | 10 | 01 | x0 | 11 | 01 | - | - | 13 | 13 |

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. In each step, going from row to row, the value at one primary input becomes more specific and the timing parameters are updated. Timing windows shrink as inputs become specified. If the logic value on a line implies no rising (falling) transition, then the timing information for this transition becomes meaningless and denoted by '-' in this table. After the values of the first three inputs are specified in the table, the arrival time of a rising transition on output 9 has shrunk from (23, 39) to (36, 36), a fixed value. This shows that partially specified vectors may specify the exact timing at certain lines.

ITR was also performed on ISCAS85 circuits by randomly specifying values at inputs. In our current approach where pulses are ignored, timing ranges become points when all inputs are specified. The degree of timing ranges shrink, as inputs are specified, is shown in Figure 10. Here we show that the timing ranges

of large circuits may shrink as fast as those for smaller circuits, in term of percentage of inputs specified. These results demonstrate that timing ranges shrink appreciably even when only a small fraction of the values are specified. This is the reason why ITR is very helpful in reducing the search required by timing oriented test generation [10].

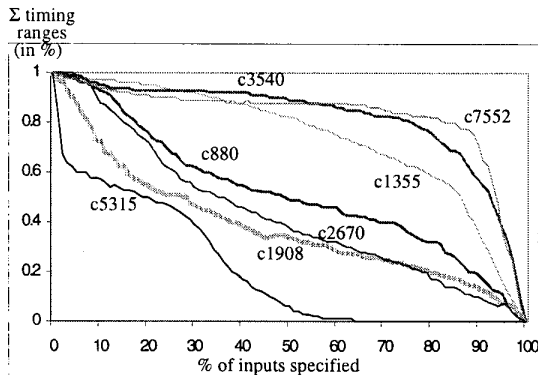


FIGURE 10. Total timing range shrink as more inputs are specified.

Performance of a crosstalk test generator [10] with (w/) and without (w/o) ITR are shown in Table 3. Timing criterion is set at the primary outputs. Test targets are aborted if the number of backtracks exceeds 1000. With ITR, the ATPG efficiency increases significantly, 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. ITR helps timing oriented test generation algorithm by reducing the search space and so (1) finding more detectable targets (the average detection rate is increased from 7.5% to 10.25%), and (2) identifying more undetectable targets (the average undetectable rate is increased from 32.13% to 72.5%). Using ITR, CPU time is increased about 10 to 20 times, and ATPG efficiency is improved from 39.63% to 82.75%. In Table 4 test generation results with and without ITR are shown for comparable run time. As can be seen in this table, by increasing test generation time of an ATPG without ITR by a factor of 10 only improves the efficiency of an ATPG without ITR by a very small amount. This clearly shows that ITR is essential for efficiency improvement of timing based ATPG.

TABLE 3. Comparison of ATPG efficiency for crosstalk delay with and without incremental timing refinement for same backtrack number.

| Circuit Name | Successful TG (%) | | | | TG Aborted (%) | | ATPG Efficiency (%) | | TG Time (s) | |
|--------------|-------------------|--------|--------------|--------|----------------|--------|---------------------|--------|-------------|--------|
| | Detected | | Undetectable | | w/o ITR | | w/ ITR | | w/o ITR | w/ ITR |
| | w/o ITR | w/ ITR | w/o ITR | w/ ITR | w/o ITR | w/ ITR | w/o ITR | w/ ITR | w/o ITR | w/ ITR |
| C432 | 5 | 15 | 32 | 68 | 63 | 17 | 37 | 85 | 138 | 1167 |
| C880 | 9 | 13 | 40 | 72 | 51 | 15 | 49 | 85 | 112 | 1664 |
| C1355 | 6 | 6 | 22 | 71 | 72 | 23 | 28 | 77 | 423 | 3403 |
| C1908 | 9 | 15 | 34 | 70 | 57 | 15 | 43 | 85 | 354 | 2555 |
| C2670 | 9 | 9 | 34 | 76 | 57 | 15 | 43 | 85 | 277 | 4870 |
| C3540 | 3 | 4 | 30 | 72 | 67 | 24 | 33 | 76 | 539 | 4661 |
| C5315 | 10 | 11 | 39 | 74 | 51 | 15 | 49 | 85 | 441 | 7745 |
| C7552 | 9 | 9 | 26 | 77 | 65 | 14 | 35 | 86 | 627 | 8651 |
| AVE. | 7.5 | 10.25 | 32.13 | 72.5 | 60.38 | 17.25 | 39.63 | 82.75 | 406 | 4406 |

TABLE 4. Comparison of ATPG efficiency for crosstalk delay with and without incremental timing refinement for similar run time.

| Circuit Name | Successful TG (%) | | | | TG Aborted (%) | | ATPG Efficiency (%) | | TG Time (s) | |
|--------------|-------------------|--------|--------------|--------|----------------|--------|---------------------|--------|-------------|--------|
| | Detected | | Undetectable | | w/o ITR | | w/ ITR | | w/o ITR | w/ ITR |
| | w/o ITR | w/ ITR | w/o ITR | w/ ITR | w/o ITR | w/ ITR | w/o ITR | w/ ITR | w/o ITR | w/ ITR |
| C432 | 6 | 15 | 32 | 68 | 62 | 17 | 38 | 83 | 2199 | 1167 |
| C880 | 10 | 13 | 40 | 72 | 50 | 15 | 50 | 85 | 1771 | 1664 |
| C1355 | 6 | 6 | 23 | 71 | 71 | 23 | 29 | 77 | 3409 | 3403 |
| C1908 | 9 | 15 | 36 | 70 | 55 | 15 | 45 | 85 | 8476 | 2555 |
| C2670 | 9 | 9 | 34 | 76 | 57 | 15 | 43 | 85 | 5620 | 4870 |
| C3540 | 3 | 4 | 30 | 72 | 67 | 24 | 33 | 76 | 6013 | 4661 |
| C5315 | 10 | 11 | 39 | 74 | 51 | 15 | 49 | 85 | 7515 | 7745 |
| C7552 | 9 | 9 | 26 | 77 | 65 | 14 | 35 | 86 | 10032 | 8651 |
| AVE. | 7.75 | 10.25 | 32.5 | 72.5 | 59.75 | 17.25 | 40.25 | 82.75 | 5628 | 4406 |

7. Summary

We proposed a framework, incremental timing analysis, that fills in the gap between static timing analysis and timing simulation. How incremental timing refinement reduces the search space was illustrated. Timing analysis is performed on a pin-to-pin delay model to find min-max timing ranges. Approaches for computing tighter min-max ranges after more line values are specified have been proposed. The differences between static timing analysis and incremental timing analysis are listed. The degree to which timing ranges shrink as more inputs specified is shown. ATPG efficiency improvement by employing incremental timing refinement is presented.

References

- [1] R. B. Hitchcock, "Timing verification and timing analysis program", Proc. of the 19th ACM / IEEE DAC 1982, pp. 594-604.
- [2] L. C. Chen, S. K. Gupta, and M. A. Breuer, "High quality robust tests for path delay faults", Proceedings VLSI Test Symposium, pp. 88-93, April 1997.
- [3] W. Y. Chen, S. K. Gupta, and M. A. Breuer, "Test generation in VLSI circuits for crosstalk noise", Proc. Int'l Test Conference, pp. 641-650, 1998.
- [4] W. Y. Chen, S. K. Gupta, and M. A. Breuer, "Test generation for crosstalk-induced delay in integrated circuits", Proc. Int'l Test Conference, pp. 191-200, 1999.
- [5] IEEE DASC standard delay format (SDF) - web page <http://vhdl.org/vi/sdf/>.
- [6] M. Abramovici, M. Breuer, and A. Friedman, "Digital Systems Testing and Testable Design", IEEE Press, 1990.
- [7] W. Kunz and D. K. Pradhan, "Recursive learning: a new implication technique for efficient solutions to CAD problems - test, verification, and optimization", IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 11, pp. 1143-1158, Sept. 1994.
- [8] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits", IEEE Trans. on Computers, vol. 30, pp. 215-222, March, 1981.
- [9] L. C. Chen, S. K. Gupta, and M. A. Breuer, "Incremental timing refinement on a min-max delay model", Computer Engineer technical report No. 00-01, Electrical Engineer - System Dept., University of Southern California, April 2000.
- [10] W. Y. Chen, "Test generation for crosstalk noise in VLSI circuits", Ph.D. dissertation, University of Southern California, EE-System Dept., 2000.