A systematic methodology to identify delay marginalities in a design during first-silicon validation[1]

Prasanjeet Das and Sandeep K. Gupta

Department of Electrical Engineering – Systems

University of Southern California

Los Angeles CA 90089

prasanjd@usc.edu, sandeep@poisson.usc.edu

Abstract— In this paper, we present a systematic framework to identify delay marginalities in a design during first silicon validation. Our method guarantees the excitation of the worst-case delay of the chips in the first silicon batch without introducing any pessimism. It embodies several innovations, including a resilient gate delay model to capture process variations, new conditions that vectors must satisfy to invoke the maximum delay of a target path, and a new approach to generate multiple vectors (vector-spaces) guaranteed to invoke the worst-case delay of the target path. We also present extensive experimental results for benchmark circuits to demonstrate the effectiveness of our method.

Keywords: delay marginalities, first silicon validation, resilient delay model, multiple vectors.

I. INTRODUCTION

The development of a new digital chip starts with its *specifications*, which describe the desired functionality and key parameters, such as performance and power (see Figure 1). A design process produces a *detailed design* in the form of a gate/transistor-level netlist and a layout. In most existing flows for custom or semi-custom design, the quality of chips shipped to customers is ensured via a sequence of three processes, namely *pre-silicon* ***verification*** of a chip's design, *post-silicon* ***validation*** of the first-

[1] This research was supported by Intel Corporation.

silicon for the design and ***testing*** of each fabricated copy when the design is fabricated in high volume [1]. Any misbehavior identified during validation that is deemed likely to cause a significant fraction of fabricated chips to fail (and hence threaten the chip's economic viability) is addressed via redesign. Each such redesign is commonly referred to as a new *silicon spin*. Such redesign is expensive and time-consuming, since it requires diagnosis to identify the root cause, redesign, creation of a new set of masks, and re-fabrication. When validation is eventually successful, the corresponding set of masks is used to manufacture chips in high volume.

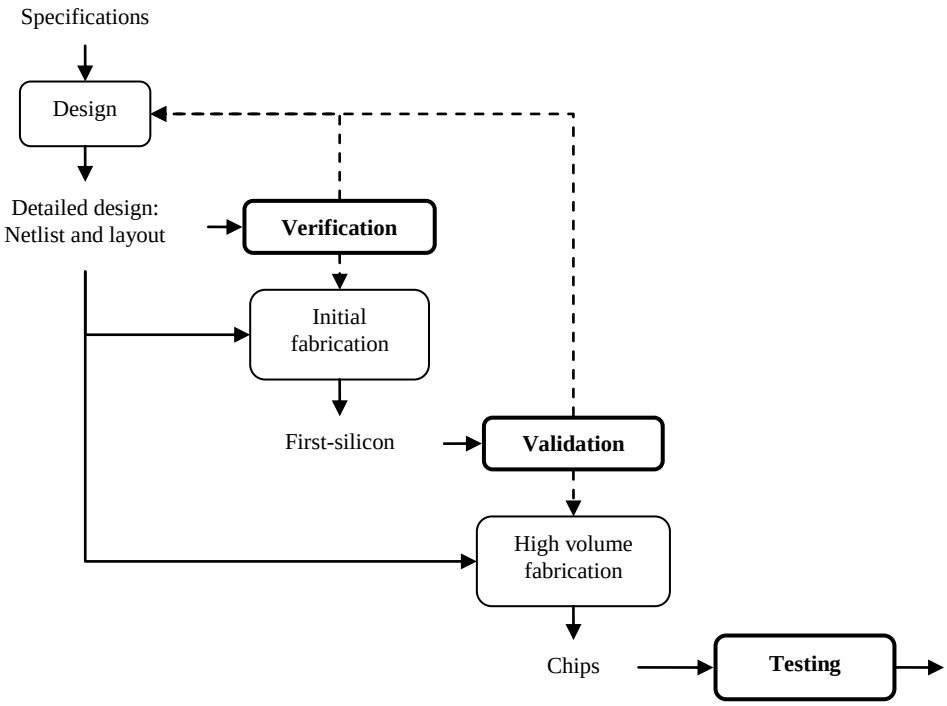


Fig. 1: A typical design flow. (Solid arrows show flow of design information, while dashed arrows indicate re-design/go-ahead signals.)

Despite advances in design and verification, it is becoming increasingly common for many chip designs to undergo multiple silicon spins. This is the case not only for high-performance custom and semi-custom chips but also for application-specific integrated circuits (ASICs), which are typically much less complex and much less aggressive in terms of area efficiency and performance. For example, as reported in [2][3], Collett International Research reports that 37% of ASICs require a second spin while 24% of ASICs require more than two spins (see Figure 2). Similar data from Numetrics Management Systems, Inc. has

been presented in [4]. The fact that multiple silicon spins are required implies that it is becoming increasingly common for many causes of serious circuit misbehavior, in particular marginalities, that can cause significant reduction in yield to be *first discovered during validation.* As the fabrication process is pushed to limits, marginalities (defined in Section II) will continue to grow in importance for the foreseeable future, rendering existing validation approaches inadequate. Hence the primary emphasis of our systematic framework is on the development of a high quality validation methodology that is guaranteed to ***detect*** all possible serious causes of circuit misbehavior (delay marginality being the one addressed here) which threaten a chip's economic viability.

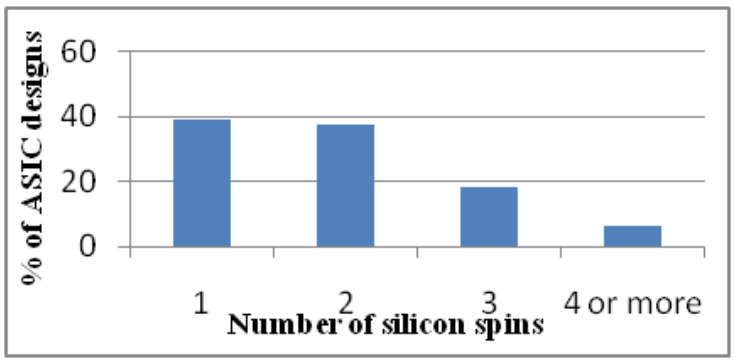


Fig. 2: Most ASICs require multiple silicon spins. (Source: Collett International Research cited in [2][3]).

For historical reasons, existing validation approaches largely target design errors missed by verification. To this end, existing validation approaches use functional, pseudo-random, and *biased* random vectors as well as verification test-benches [5][6][7] . Most approaches do not quantify the quality of vectors. Few approaches that do, do not consider marginalities but use logic and higher-level metrics adopted from software testing [8] and high-level verification [6][7], such as HDL statement/block coverage. None of the existing approaches generates vectors with the objective of invoking worst-case severities for effects that are behind marginalities.

Delay marginality is one such effect which eventually leads to slow ICs. Path delay test (PDT) has traditionally been believed to be superior for identifying slow paths and considered to be more useful for speed binning and performance characterization. However, recent silicon studies by nVidia [9], Freescale

[10], and Sun [11] have shown that existing path delay testing approaches generate vectors that fail to invoke the worst-case delays in first-silicon (see Section II). Existing approaches for generating vectors for testing for low-level effects such as capacitive crosstalk and ground bounce [12][13][14][15][16][17][18][19][20][21] and [22] , use nominal values for parameters and parasitics. This makes the generated vectors *non-resilient,* i.e., unreliable for any fabricated copy of the chip, and hence unsuitable for validation.

In this report we propose a systematic approach to generate a set of multiple vectors that will be used for high quality post silicon delay marginality validation of high performance designs. In particular our approach will guarantee invocation of the worst-case delay of the chips in first silicon batch.

The report is organized as follows. In Section II, the motivation and importance for our approach is presented. In Section III, the overall approach is presented. In Section IV, the new concepts and algorithms are presented. The experimental setup and results are described in Section V. Finally, conclusions are presented in Section VI.

II. BACKGROUND

A chip may have erroneous behavior due to (i) ***design errors***, (ii) ***marginalities***, and (iii) ***defects***. *A design error* is a logic error or a gross delay problem in a design that causes an unacceptable deviation from desired functionality or a catastrophic reduction in yield. It is caused by a designer mistake or serious inadequacies of the models or tools used for design. All information required to identify design errors in a circuit, namely its gate-level netlist and first-order models of delays of gates and wires, is available during pre-silicon verification. Since a design error is catastrophic, re-design is necessary. Hence, one should strive to detect all design errors during verification to avoid an additional silicon spin and associated mask costs and increase in time-to-market.

A ***marginality*** is any aspect of a design that makes it probable that a significant fraction of fabricated chips will have erroneous behavior, even in the *absence of defects* and even when the *variations in the fabrication process (**process variations**) are within the normally expected levels,* i.e., even when there is

no abnormal process drift. ***Low-level effects***, such as, delay variations, inadequate noise margins, excessive leakage currents, charge sharing, ground bounce, crosstalk, and inherently stochastic behavior, are root causes of marginalities. The severity of each such effect depends on the values of circuit parameters and parasitics and may be *aggravated by process variations*. Unavailability of information and computational complexity make it difficult to identify all marginalities during pre-silicon verification without significant *guard-banding*, i.e., pessimism. Also since a marginality causes erroneous behavior for a fraction of the normal range of process variation, it is possible to leave an instance of marginality un-rectified before moving the design into high volume fabrication, when the reduction in yield due to the marginality is lower than the costs (e.g., increase in the time-to-market) and penalties (e.g., performance reduction) associated with redesign.

A. Motivation

This paper is motivated by the clear trend that, despite advances in design and verification, a growing proportion of chips require two or more silicon spins. This trend will continue to become more severe for the foreseeable future – CMOS scaling in the near future and wide adoption of new nano-technologies thereafter. Hence, it is imperative to strengthen techniques for verification and validation. Marginalities constitute an increasing proportion of misbehaviors first discovered during validation. This increase in importance of marginalities – caused by low-level effects and aggravated by variations – is due to the following reasons:

- Increase in percentage variations in values of key parameters and parasitic [23], as feature sizes and separations shrink deep into nano-scale and push fabrication processes to their limits and beyond.
- Increase in the importance of low-level effects – delay variations, inadequate noise margins, charge sharing, crosstalk, and so on.
- Need for aggressive design, and hence lower guard-banding, as we experience a slowdown in scaling, notably in its performance benefits.

The fact that this increase in importance of marginalities will continue unabated for the last years of CMOS scaling is clearly evident from the amount of research effort devoted to related concerns (e.g., see the Proceedings of IEDM for any recent year). The importance of our research will increase even more dramatically when new technologies start replacing or supplementing CMOS. While the technologies that might eventually replace or supplement CMOS are still in infancy, it is clear that each will be plagued by corresponding low-level effects and noise sensitivity to even greater extents. For example, in some technologies, reliable operation can be obtained only when the functional circuit is supplemented by redundant circuitry whose total size is *many times* the size of the functional circuit (e.g., see [24][25]).

B. Previous work: Recent silicon studies

We have already reviewed shortcomings of the existing validation approaches in Section I. Since we are primarily interested in the validation of high performance circuits, here we summarize several silicon experiments from nVidia, Freescale, and Sun [9][10][11] to understand the limitations of path delay testing when used to characterize the timing behavior of circuits. Though the specific details may vary, the general methodology followed by industry for PDT is:

- Use static timing analysis (STA) to obtain a ranked list of timing critical paths.
- Categorize paths based on path cell types – wire bound and gate bound.
- Select top ranked paths.
- Generate statically-sensitized robust tests for these paths.

One common observation from [9][10][11] is that the *PDTs invoke lesser delay than the functional tests since the critical paths in silicon are different from those identified by STA.*

Reasons evident from [9][10][11] for this anomaly viz-a-viz everyone's expectations for PDT are:

- ***Inaccurate delay models* [10][11]**– Existing delay models do not take into account all relevant lower order effects.
- ***Non resilient delay models* [10][11]** – Existing delay models do not take into account the effects of variations on delay.

- ***Wrong path selection approach*** **[10][11]** – Selecting top ranked paths based on ranking will eliminate certain paths with similar delays from consideration for test generation.
- ***Wrong sensitization conditions*** **[10][11]** – Existing sensitization conditions are based on erroneous assumptions regarding gate delay and hence do not excite worst case delay and can falsely declare certain testable paths untestable.
- ***Statically sensitized robust PDT*** **[9][11]** – PDTs used in these studies apply static values at side inputs to facilitate delay diagnosis. Such tests do not guarantee invocation of the worst-case delay for a target path.

Additional reasons, such as test application differences and pre silicon – post silicon netlist mismatches, account for the anomaly under consideration. Though we deal with marginalities and not faults, a systematic PDT approach, where above shortcomings are eliminated is a suitable candidate for our framework.

C. Unique challenges and key ideas

We begin by examining the proposed framework from the perspective of its potential users to identify the practical considerations that we must use to define the objective functions and constraints for our research.

- ***Completeness:*** We must impose quality constraints on validation as these are necessary to provide the dramatic reductions in costs and time to market. In particular, our framework must detect a high percentage (preferably, 100%) of all possible instances of delay marginalities to virtually eliminate the need for a third silicon spin.
- ***No pessimism:*** It is imperative that our framework for validation does not raise many false alarms. Each false alarm requires time-consuming and expensive diagnosis.
- ***Only require realistically available models and information regarding parameters, parasitics, and variations:*** The uncertainty in the values of parameters, variations, and so on continues to grow with each scaling generation of fabrication process. Hence, any such framework must

require only a practically characterizable subset of all such information. This is a prerequisite to ensuring completeness and accuracy, since use of unreliable models and values (including those obtained via extrapolation from the past technology generation) may cause validation to miss debilitating marginalities that emerge for a new fabrication process.

In addition to above, precise models for low-level effects may be unavailable or not useable due to their complexities or reliance on unavailable parameter values.

We will deal with these challenges by using our following ***key ideas***:

- The use of actual chips allows each vector to invoke *every relevant low-level effect* while inherently considering the precise values of all circuit parameters and parasitics.
- Process variations and variations in *voltage and temperature* are explicitly captured *by sampling chips from the first-silicon batch and repeating validation for different values of voltage and temperature.* This implies that all we need to do is to generate and apply appropriate vectors and the chips take care of the rest.
- As vectors can be applied to chips at rates near 10^9 vectors/second, ***large numbers of vectors can be used for validation***, especially those applied using the circuit's functional modes. Hence we can develop approaches that use more vectors to achieve specific objectives, including
  - A methodology to deal with uncertainties in values and models.
  - to identify ***resilient sets of multiple vectors*** to detect each target, i.e., identify a set containing vectors that is guaranteed to include the vector that invokes the worst-case effect for the target despite all model, parameter, and variation uncertainties.
- We can use ***bounding approximations***, i.e., approximations that provide upper and lower bounds on the actual values (of parameters and parasitics) or behavior (of low-level electrical effects). Bounding approximations will be used to reduce run-time complexities of tools, and to tackle the unavailability of precise models and values.
- During validation, these approximations will be used to generate vectors. A loose bounding approximation might lead to generation of a larger set of vectors than necessary. However, when

the generated vectors are applied to chips, the behavior invoked is determined by the actual chips and is unaffected by any approximation used during vector generation. Hence, the ***use of approximations in our framework does not make validation pessimistic*.**

III. PROPOSED APPROACH

In this paper, combinational circuits comprised of primitive gates are considered. We start by presenting the basic terminology and definitions [26] and a top-down view of the overall approach.

A. Terminology and definitions

- **Maximum rising arrival time** (A^X_{RL}): The latest (largest) time at which a rising transition at line X may reach 50% of power supply voltage, V_{DD}. Maximum falling arrival time (A^X_{FL}), minimum rising arrival time (A^X_{RS}), and minimum (smallest) falling arrival time (A^X_{FS}) are similarly defined.
- **Logic value system**: Throughout this paper we deal with sequences of two vectors, *even though for simplicity we often refer to them as vectors*. Hence we denote logic values at a line by using a subset of {CF, CR, S0, S1, TF, TR, H0, H1}, where CF stands for clean falling (no hazard), S0 stands for static 0 (no hazard), TF stands for transition to value 0 (dynamic hazards possible), and H0 stands for hazardous 0 (static hazards possible). CR, S1, TR, and H1 are similar.
- **Controlling value** (CV): The controlling value of a multi–input gate is the logic value which when applied to any one of the gate's inputs, uniquely determines its output value. NCV represents the gate's *non controlling value*. The output value caused by the application of the controlling value at any one gate input is called *controlling response*. *Non controlling response* is the complement of the controlling response.
- **To-controlling transition:** The to-controlling transition at an input of a multi-input gate is a transition from NCV to CV. *To-non controlling transition* is similarly defined.
- **Logical path** (P): A logical path (P) is a sequence of lines along a circuit path L_1 (a primary input), L_2, …, and L_n (a primary output) and a set of signal transitions Tr_1, Tr_2, …, and Tr_n, where $Tr \in \{R, F\}$, such that Tr_i represents the signal transition at L_i. The lines L_1, L_2, …, and L_n are called on-path

lines. Gates along P are called on-path gates. If a line directly connects to one of the on-path gates but is not an on-path line, it is called a side-input of path P.

B. The overall approach

The overall approach has four main components (see Figure 3).

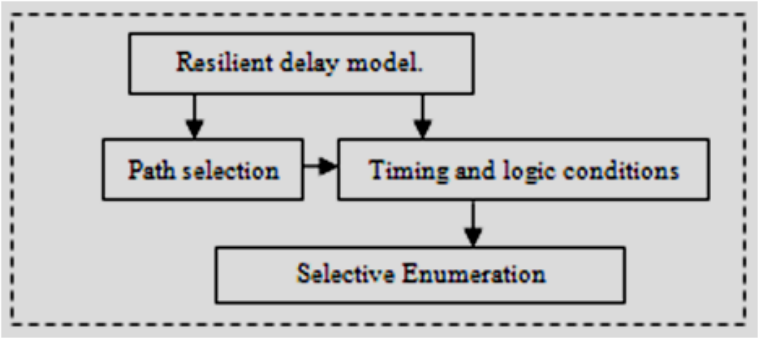


Fig. 3: The overall approach

- ***Resilient delay model:*** A delay model that will not be invalidated by inaccuracies and variations as it captures these using bounding approximations.
- ***Path selection:*** A path selection approach that uses the resilient delay model to identify a set of paths that is guaranteed to include all paths that may potentially cause a timing error if the accumulated values of additional delays along circuit paths is upper bounded by a desired limit.
- ***Timing and logic conditions for guaranteed detection:*** Identification of necessary timing and logic conditions that will guarantee invocation of worst case delay at each gate along a target path.
- ***Selective enumeration*****:** Using partial ordering (based on the worst case delay invoked) among the logic conditions, we develop an innovative search algorithm to arrive at a set of multiple vectors that will resiliently invoke maximum delay of the target path.

C. Resilient delay model

Pin-to-pin delay model is used by most STA tools. One main deficiency of this model is that it does not captures the effects of simultaneous switching on delay. Simultaneous to-controlling transitions at inputs of a primitive gate decrease gate delay due to activation of multiple charge paths [26]; simultaneous to-

non-controlling tansitions at inputs increase the gate delay due to Miller effect [27]. Figure 4 shows the *delay vs skew* for near-simultaneous transitions at the inputs of a 2-input NAND gate.

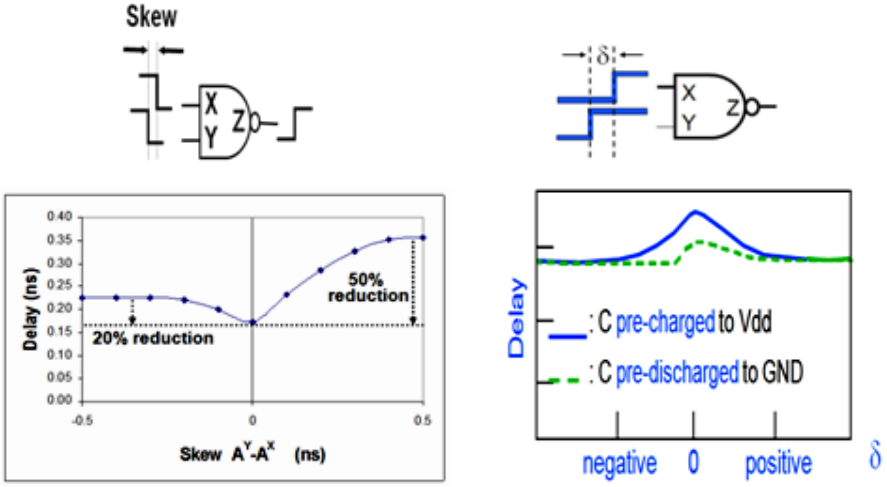


Fig. 4: Delay vs skew curve for near-simultaneous transitions

The delay model proposed in [26] only uses qualitative information such as the causality property along with provable properties of underlying physics such as the effect of near simultaneous transitions on gate delay and hence is a suitable candidate for our framework. But a single curve as proposed in [26] cannot capture all inaccuracies and variations, we need an envelope comprising of two curves – one upper bound and one lower bound – to bound all inaccuracies and effects of process variations [28].

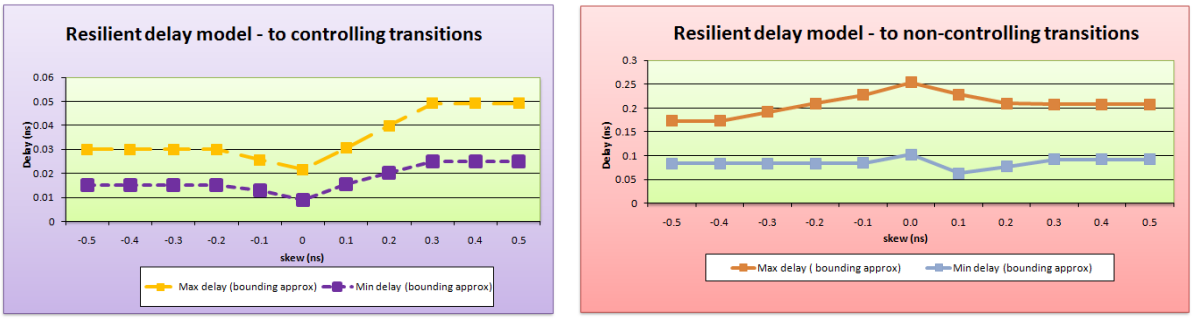


Fig. 5: Delay vs skew curve for near simultaneous transitions

The inaccuracies and variations in these delay parameters can be captured using bounding approximations where the inaccuracies at each circuit line are bounded by $\pm\rho\%$ where ρ captures inaccuracies in circuit parameters such as V_{th}, L_{eff}, t_{ox} etc. Our approach expresses variability in terms of

the parameters of the devices in the gates. Using the value of device variability from [29] we perform Monte Carlo simulations to obtain the two envelopes to bound the gate delay as shown in Figure 5(details can be found in [28]).

D. Delay defining parameters

The delay defining parameters only use the limited qualitative information to define delays.

- ***Maximum delay (α)*** is the maximum delay associated with every input to every output for a gate. For a gate with input X, output Z and rising transition at output, the maximum delay is denoted by $\alpha_{XZ}R$.
- ***Near simultaneous range (δ)*** is the minimum separation between transitions at inputs preventing the effect of one from interfering with the other. For a 2-input gate with falling transitions at inputs X and Y when X precedes Y, this is denoted by $\delta_{XY}F$.

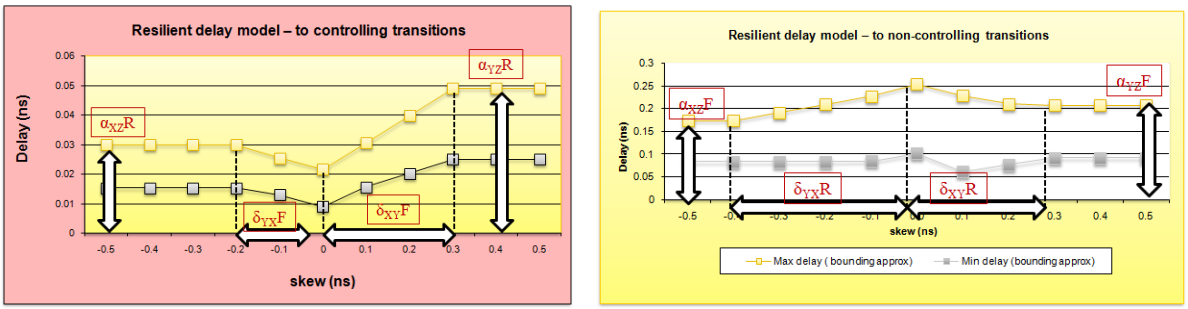


Fig. 6: Delay defining parameters for resilient delay model [28]

Figure 6 shows the delay defining parameters for the resilient delay model [28]. Note that our method can work with any other resilient delay model.

E. The value system

Delay validation and testing requires a logic value system where the state of a signal has the ability to represent any possible situation that can occur during two consecutive vectors [37]. A resilient delay validation approach requires a value system that is complete. By complete we mean to say that the value system must be able to represent transitions and corresponding delay effects. Thus a complete value system must be able to represent steady values as well as hazardous (static as well as dynamic)

transitions. Also for basic gates the impact on delay is different for static or dynamic hazard at the side input. Table I shows the analysis of existing prevalent logic value systems used for testing. The basic 8 value system of [30] can be selected as the basis for deriving a complete value system as it distinguishes successfully between static and dynamic hazards (using the concept of transition as well as edges) .

Table I: Analysis of Value System

Reference	Type of value system	Basic values	Total values	Able to represent hazards	Able to distinguish between static and dynamic hazards
[33][36][38]	Two	(0, 1)	4	N	N
[32]	Four	(S0, S1, R, F)	4	Y	N
[26][39][40]	Three	(0,1,X)	9	Y	N
[35]	Five	(0, 1, s, p, -)	6	Y	N
[34]	Five	(s0, s1, u0, u1, XX)	5	Y	N
[37]	Six	(p0, p1, s0, s1, -0, -1)	23	Y	N
[30][31]	Eight	(S0, S1, T0, T1, CR, CF, H0, H1)	53	Y	Y

The basic 8 values {S1}, {S0}, {CR}, {CF}, {T1}, {T0}, {H1}, and {H0} can then be extended to the 53 value system using the following six step procedure of [37][41] that generates a complete value system from a basic value system using sensitization conditions and subsequent forward and backward implications on basic gates:

1. Define the basic eight values - each of which contains only one of the basic logic values.
2. Define "X" as composite value consisting of all basic logic values.
3. Considering currently identified composite logic values construct (or reconstruct) tables to be used by (NOT/NAND/NOR) gates forward implication procedure.
4. By applying combinations of currently identified composite logic values to backward implication procedure, identify new composite values that are needed to be added.
5. By applying combination of currently identified composite logic values to the forward implication procedure, identify other new composite values to be added.
6. Repeat steps 3 to 5 until no new value is added.

The resultant 53 value system thus provides closure for backward and forward implication procedures. This value system ensures that no information is lost during implication and hence is complete in every

sense. Note that our delay validation approach can work with any other complete value system too.

Table II: The complete eight value system [30]

#	Value	#	Value	#	Value	#	Value	#	Value
1	{}[1]	12	{CF,T0}[5]	23	{CR,T1,H1}[5]	34	{S0,H0,CR,T1}[5]	45	{CF,CR,S1,T0,T1,H0}[5]
2	{CF}[3]	13	{CR,T1}[5]	24	{CR,T1,H0}[5]	35	{S1,H1,CF,T0}[5]	46	{CF,S0,S1,T0,H1,H0}[5]
3	{CR}[3]	14	{CR,S1}[5]	25	{S1,T1,H0}[5]	36	{CF,S0,T0,H1}[5]	47	{CR,S0,S1,T0,H1,H0}[5]
4	{S1}[3]	15	{S0,CF}[5]	26	{CF,T0,H1}[5]	37	{S0,S1,H0,H1}[5]	48	{CF,CR,S0,T0,T1,H1}[5]
5	{S0}[3]	16	{T0,H1}[5]	27	{S0,T0,H1}[5]	38	{CR,S1,T1,H1}[5]	49	{CF,CR,S1,H0,H1,T1,T0}[4]
6	{T0}[3]	17	{T0,H0}[5]	28	{S0,S1,H0}[5]	39	{CR,T0,T1,H1,H0}[5]	50	{CF,CR,S0,H0,H1,T1,T0}[4]
7	{T1}[3]	18	{T1,H0}[5]	29	{S0,S1,H1}[5]	40	{CF,T0,T1,H1,H0}[5]	51	{CF,CR,S0,S1,H1,T1,T0}[5]
8	{H0}[3]	19	{T1,H1}[5]	30	{CF,S0,S1}[5]	41	{CF,S1,S0,T0,H1}[5]	52	{CF,CR,S0,S1,H0,T1,T0}[5]
9	{H1}[3]	20	{CF,S1}[5]	31	{CR,S0,S1}[5]	42	{CR,S1,S0,T0,H1}[5]	53	{CF,CR,S0,S1,H0,H1,T1,T0}[2]
10	{S1,H1}[4]	21	{CR,S0}[5]	32	{S0,H0,CF,T0}[4]	43	{CF,CR,T0,S1,T1,H1}[4]		
11	{S0,H0}[4]	22	{CF,T0,H0}[5]	33	{S1,H1,CR,T1}[4]	44	{CF,CR,T0,S0,T1,H1}[4]		
1-Empty value, 2-Fully composite value, 3-Basic value, 4-Values based on sensitization conditions, 5-Values derived from implications (forward and backward)									

F. Path selection approach

In [30] a new approach to efficiently identify paths for delay testing is arrived at using a realistic delay model [26] and several new concepts (timing threshold, settling times [31], and timing blocking line) and algorithms, to identify a set of paths that is guaranteed to include all paths that may potentially cause a timing error if the accumulated values of additional delays along circuit paths is upper bounded by a desired limit, Δ. We use this approach as it deals with upper and lower bounds and also checks for both functional sensitization and high delay excitation at the same time.

G. Taxonomy of timing cases

Consider a 2 input NAND gate with inputs X and Y and output Z. Let X be the on-path input and Y be the side input. Let arrival times of X and Y be [A^X_{FS}, A^X_{FL}] and [A^Y_{FS}, A^Y_{FL}] respectively.

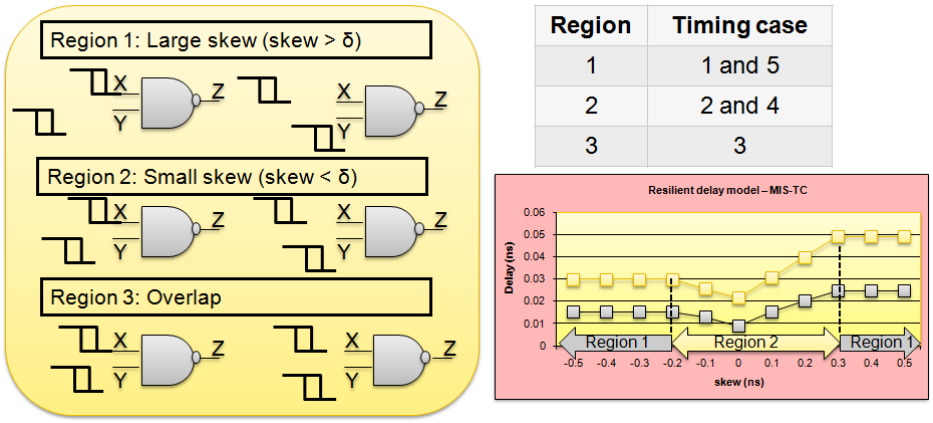


Fig. 7: To controlling transitions at the inputs of a 2-input NAND gate

Using the delay parameters α and δ, the following timing cases corresponding to a falling transition at on path input X can be derived.

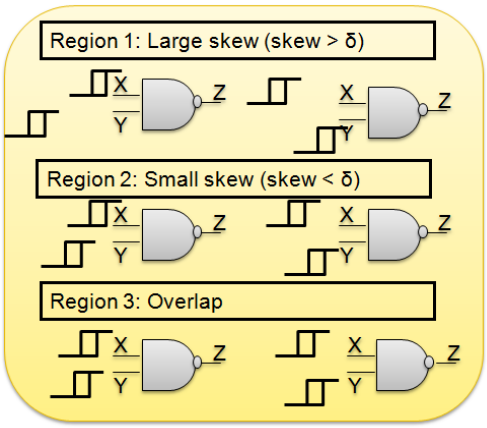
- ***Region 1: Y not near-simultaneous with respect to X:*** Transitions at X and Y are separated at least by $\delta_{YX}F$ (for Y before X) or by $\delta_{XY}F$ (for X before Y). Hence X and Y will not interact with each other. This is Region 1 of the delay curve in Figure 7.
- ***Region 2: Y near-simultaneous but does not overlap with X:*** Transitions at X and Y are separated at most by $\delta_{YX}F$ (for Y before X) or by $\delta_{XY}F$ (for X before Y). Transition at Y can affect the transition at X due to first order effects such as multiple charge paths [26]. This is Region 2 of the delay curve in Figure 7.
- ***Region 3: Y overlaps X:*** The timing ranges of X and Y are not mutually exclusive. Hence transitions at X and Y can overlap (as shown in Figure 7).

Figure 7 clearly shows that in Region 1 the delay of the NAND gate is equal to the pin to pin delay whereas in Region 2 the delay needs to be arrived by using simulations and curve fitting techniques [28].

Table III: Classification of timing cases for to-controlling transition at on path input of a 2 input NAND gate

Timing case	Explanations
1. Y before X and not near simultaneous	Y arrives at least $\boldsymbol{\delta}_{YX}F$ before transition at X. Hence Y and X cannot interfere. $\mathbf{A^Y_{FL} < =A^X_{FS} - \delta_{YX}F}$
2. Y before X and near simultaneous but no overlap	Y arrives at most $\boldsymbol{\delta}_{YX}F$ before transition at X. Effects of Y and X may interfere with each other due to Miller effect, multiple charge paths, state of internal capacitance etc. $\mathbf{A^X_{FS} > A^Y_{FL} > A^X_{FS} - \delta_{YX}F}$ $\mathbf{A^X_{FS} > A^Y_{RL} > A^X_{FS} - \delta_{YX}F}$
3. Y overlaps X	The timing ranges of X and Y overlaps. X and Y may or may not interfere with each other
4. Y after X and near simultaneous but no overlap	Y arrives at most $\delta_F XY$ after transition at X. Effects of Y and X may interfere with each other due to multiple charge paths etc. $\mathbf{A^X_{FL} < A^Y_{FS} < A^X_{FL} + \delta_{XY}F}$
5. Y after X and not near simultaneous	Y arrives at least $\delta_F XY$ after transition at X. Hence Y and X cannot interfere. $\mathbf{A^Y_{FS} >= A^X_{FL} + \delta_{XY}F}$

Table III shows the detailed classification of timing cases along with necessary equations for to-controlling transition at on path input of a 2-input NAND gate. Table IV shows the same for the to-non controlling case (also see Figure 8).



Region	Timing case
1	1 and 5
2	2 and 4
3	3

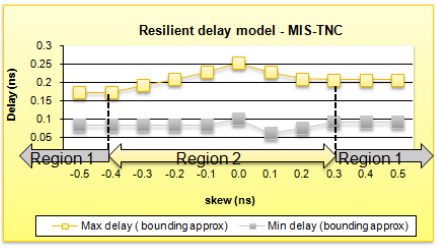


Fig. 8: To non-controlling transitions at the inputs of a 2-input NAND gate

Table IV: Classification of timing cases for to-non controlling transitions at inputs of a 2 input NAND gate

Timing case	Explanations
1. Y before X and not near simultaneous	Y arrives at least $\boldsymbol{\delta}_{YX}R$ before transition at X. Hence Y and X cannot interfere. $\mathbf{A^Y_{RL} < =A^X_{RS} - \delta_{YX}R}$
2. Y before X and near simultaneous but no overlap	Y arrives at most $\boldsymbol{\delta}_{YX}R$ before transition at X. Effects of Y and X may interfere with each other due to Miller effect, multiple charge / discharge paths etc. $\mathbf{A^X_{RS} > A^Y_{RL} > A^X_{RS} - \delta_{YX}R}$
3. Y overlaps X	The timing ranges of X and Y overlaps. X and Y may or may not interfere with each other
4. Y after X and near simultaneous but no overlap	Y arrives at most $\delta_R XY$ after transition at X. Effects of Y and X may interfere with each other due to Miller effect, multiple charge / discharge paths etc. $\mathbf{A^X_{RL} < A^Y_{RS} < A^X_{RL} + \delta_{XY}R}$ $\mathbf{A^X_{RL} < A^Y_{FS} < A^X_{RL} + \delta_{XY}R}$
5. Y after and not near simultaneous	Y arrives at least $\delta_R XY$ after transition at X. Hence Y and X cannot interfere. $\mathbf{A^Y_{RS} >= A^X_{RL} + \delta_{XY}R}$

H. Timing conditions and partially ordered graphs

Consider a to-controlling transition at on path input X (Figure 7). The *functional sensitization conditions* [31] provide the necessary conditions at the side input Y to be able to sensitize the target path from X to Z. The *robust conditions* as used in [9][11] will allow only {S1} on the off path inputs for all the timing cases and eventually will miss certain suitable candidates and hence may miss the vector that invokes the worst case delay. Hence, robust conditions are *not suitable* for generating vectors for delay marginality validation.

In contrast to robust conditions, our approach starts with the set of all possible values (see Table V) and eliminates only those cases that can be proven as being unable to invoke the worst-case delay under any circumstance.

Table V: High delay sensitization conditions for to-controlling transition at on path input of a NAND gate

Timing Case	Logic Conditions
1. Y before X and not near simultaneous	{CR, TR, S1, H1}
2. Y before X and near simultaneous but no overlap	{CR, TR, S1, H1,CF,TF}
3. Y overlaps X	{CR, TR, S1, H1, TF, CF.H0}
4. Y after X and near simultaneous but no overlap	{CF, TF, S1, H1}
5. Y after X and not near simultaneous	{CF, TF, S1, H1}

Using the limited timing information available and the timing cases we arrive at our high delay sensitization conditions (HDS) for each side input (Figure 9).

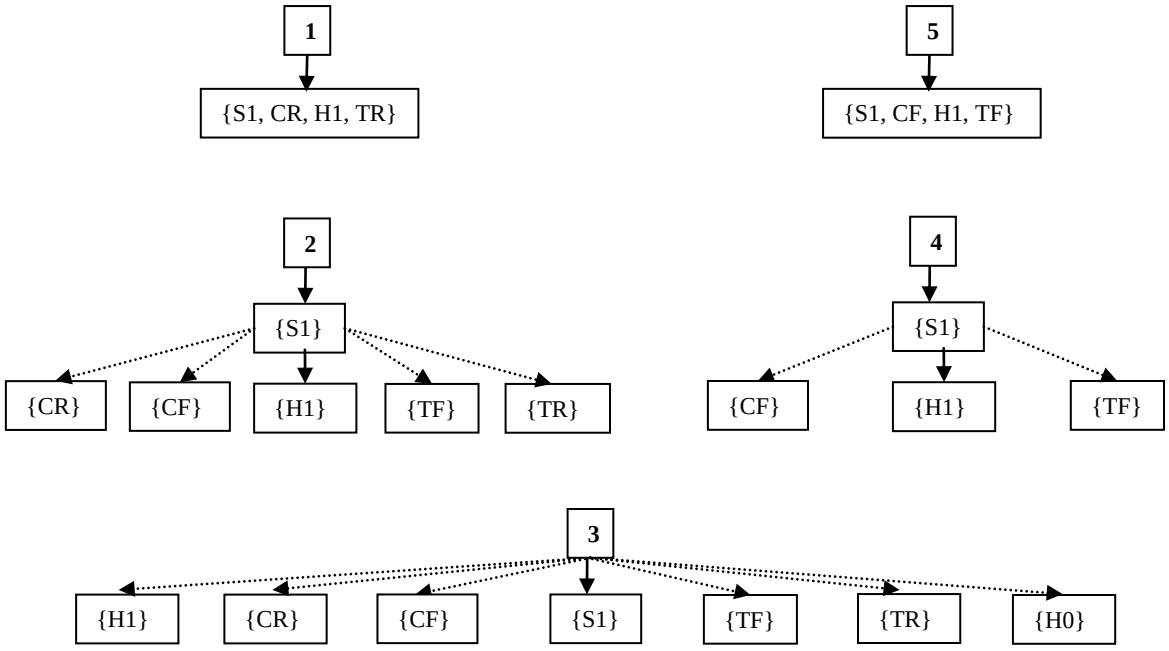


Fig. 9: Partial ordered graph for the side input values for various timing cases for to-controlling transition at on path input of a 2-input NAND gate

Consider the timing case 1 (*Y before X and not near simultaneous*), in order to propagate the falling transition at on path input X, a final value 1 at Y becomes imperative and hence we arrive at the set of

logic conditions {CR, TR, S1, H1}. Since transitions at Y and X do not interact with each other, each of the four members of this set is equally good i.e., invokes identical delay for the target path, and thus all are equivalent (Figure 9). Going along similar lines, for the timing case 5 (*Y after X and not near simultaneous*), we arrive at the equivalent set of logic conditions {CF, TF, S1, H1} (Figure 9).

Consider the timing case 4 (*Y after X and near simultaneous with no overlap*),in order to propagate the falling transition at on path input X, an initial value 1 at Y becomes imperative and hence the all inclusive logic conditions {CF, TF, S1, H1}. However, because of possibility of activating multiple charge paths [26] due to a falling transition at the side input, values {CF, TF, H1} becomes inferior to {S1} as they will invoke delay that is guaranteed to be less than or equal to the delay of a classical robust test. However, we cannot establish any provable relationship between the abilities of these three values to invoke greater delay for the target path. Hence, {H1}, {CF} and {TF} are considered non inferior with respect to each other (Fig. 8). Similarly, the partial order graph for the timing case 2 (*Y before X and near simultaneous with no overlap*) can be arrived at and shown in Figure 9.

Consider the timing case 3 *(Y overlaps with X), in order t*o attain a to-controlling response at Z, Y can have any or a subset of {S1, CR, TR, H1, CF, TF, H0}. Since transitions at Y and X can interfere with each other, nothing can be said in affirmative about the effect of one on the on-path delay over the other. Hence each of {CR}, {TR}, {S1}, {H1}, {CF}, {TF} and {H0} are non inferior with respect to each other (Figure 9).

Table VI: High delay sensitization conditions for to-non controlling transition at on path input of NAND gate

Timing Case	Logic Conditions
1. Y before X and not near simultaneous	{CR, TR, S1, H1}
2. Y before X and near simultaneous but no overlap	{CR, TR, S1, H1}
3. Y overlaps X	{CR, TR, S1, H1}
4. Y after X and near simultaneous but no overlap	{S1, H1, CR, TR}
5. Y after X and not near simultaneous	{S1, H1}

Table VI shows the high delay sensitization (HDS) conditions for the to-non controlling case. Figure 10 shows the partial ordered graphs for the same. It can be seen that the logic values for timing cases 3 and 4 (the near-simultaneous transition cases) are completely unordered. This is because of the fact that multiple to-non controlling transitions can increase as well as decrease the gate delay [27].

Consider the timing case 1 (Y before X and not near simultaneous). In order to propagate the rising transition at on path input X, a final value 1 at Y is necessary and hence we arrive at the set of logic conditions {CR, TR, S1, H1} shown in Table I. Since transitions at Y and X do not interact with each other for this timing case, each of the four members of this set is equally good, i.e., each invokes identical delay for the target path, and thus all are equivalent as shown in Figure 10. This graph depicts that ***any*** of these four values, CR, TR, S1 and H1, guarantees invocation of the same delay for the target path.

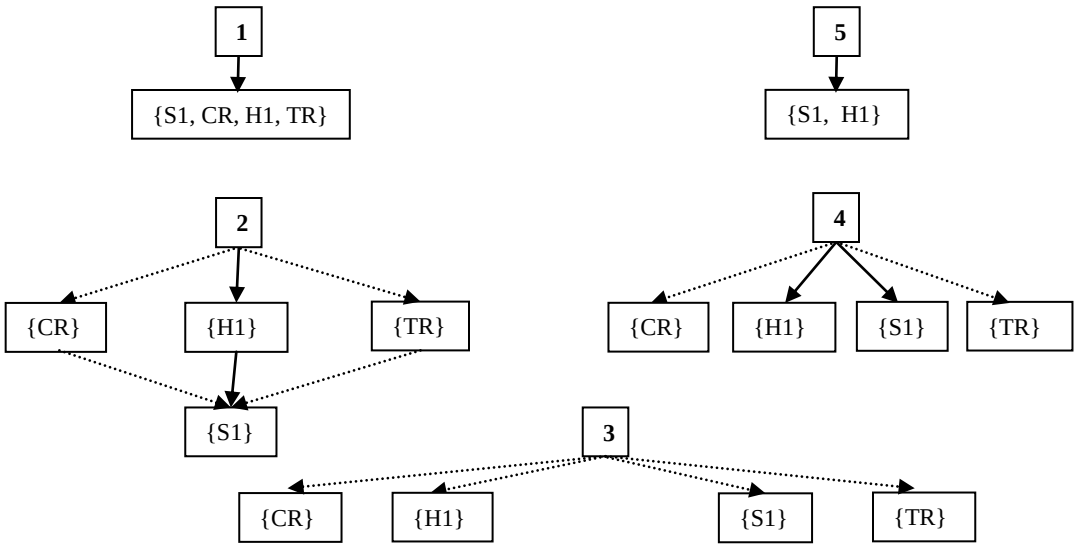


Fig. 10: Partial ordered graph for the side input values for various timing cases for to-non controlling transition at on path input of a 2-input NAND gate

Consider the timing case 2 (Y after X and near simultaneous with no overlap). In order to propagate the rising transition at on path input X, an initial value 1 at Y is necessary and hence we derive the all inclusive set of logic conditions as {CR, TR, S1, H1}. However, because of the Miller effect [27] due to a rising transition at the side input, any of the values in the set {CR, TR, H1} can be shown to be always superior to any in {S1} to invoke the worst case delay. Since we cannot derive any provable relationship

between the abilities of any of the other three values, namely {CR}, {TR}, and {H1}, to invoke greater delay for the target path compared to the remaining two, we consider {CR}, {TR}, and {H1} non inferior with respect to each other as shown in Figure 10.

Consider timing case 3. To attain a to-non-controlling response at Z, Y can have any or a subset of {S1, CR, TR, H1}. Since transitions at Y and X can interfere with each other, nothing universal can be said about the effect of one of these values on the target path's delay compared to any other value. Hence each value {CR}, {TR}, {S1} and {H1} is non inferior with respect to each other as shown in Figure 10. In such case complete enumeration is deemed necessary to guarantee that the generated vectors are collectively guaranteed to invoke the worst-case delay of the target path.

I. Selective enumeration

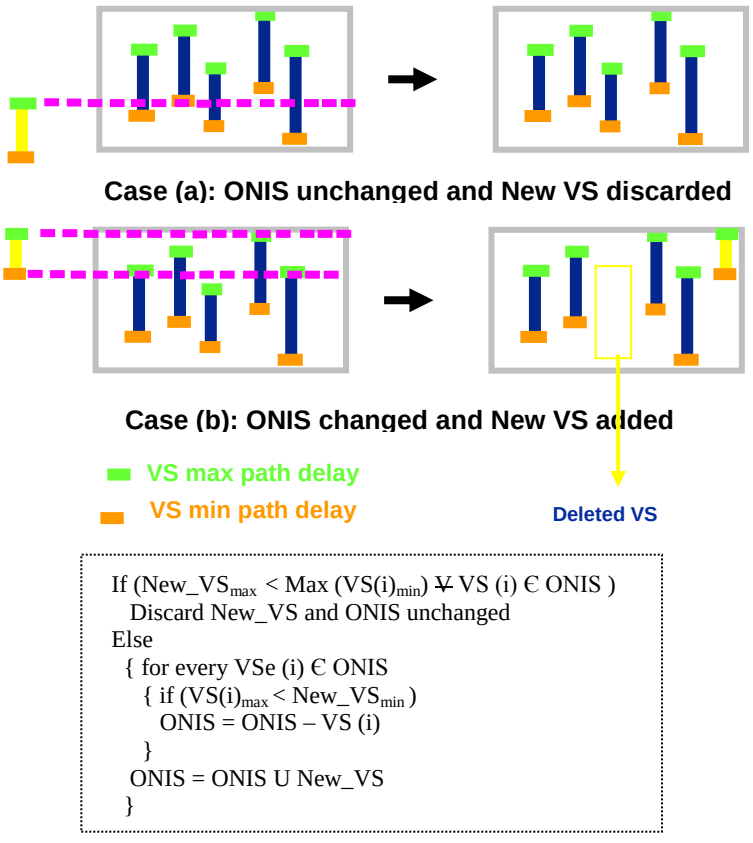
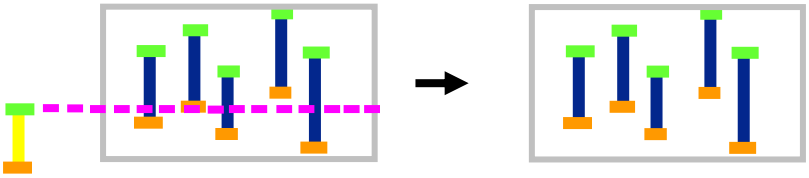


Fig. 11: Elimination of inferior vector-spaces at leaf nodes

Using the partial ordered graphs we refine the logic values at side inputs to arrive at a set of multiple vectors termed as a *test vector-space* guaranteed to resiliently detect the target path. A test vector-space is a partially-specified vector where designated partially-specified values are expanded into all possible combinations and *every one* of them will be applied during validation. (For example, if the first two partially-specified values in *xx*d10 are designated as '*x*' for expansion while the third is designated as 'd' for don't care, then our validation vector set comprises of the following four partially-specified vectors: 00d10, 01d10, 10d10, and 11d10, where d is don't care and can be replaced by either 0 or 1).

We eliminate only provably inferior (low delay invoking) vectors in our approach and hence our test vector-spaces comprise of sets of non inferior vectors that can resiliently detect a target. Our all inclusive approach selects all non inferior vector sub-spaces as suitable candidate for validation. The search starts by enumerating the values at each side input and reducing them to either a single value or a single equivalent set. At the leaf node of our search tree, it deals with the elimination of inferior vector sub-spaces and storage of non inferior vector-spaces as per the algorithm shown in Figure 11.

Let ONIS denotes the *old non inferior set* and *VS(i)* represents the i^{th} vector-spaces in the same. Then in order to speed up the search process the elimination of inferior vector- spaces can be done at intermediate nodes as per Figure 12.



Case (a): ONIS unchanged and New VS discarded

If (New_VS_{max} < Max $(VS(i)_{min})$ $\forall$ VS (i) $\in$ ONIS)
Discard New_VS and ONIS unchanged

Fig. 12: Elimination of inferior vector-spaces at intermediate nodes

IV. TIMING DEPENDENT FRAMEWORK

We now present our approach for identifying a set of multiple vectors (MV) known as a *test vector-space* guaranteed to resiliently detect the target path with no pessimism, even if loose bounding approximation models used to derive or analyze vectors. We have divided our approach to six phases (Figure 13).

- ***Phase 0: Path selection*** where using ETA [31] and a timing threshold (Δ) we arrive at a set of target paths.
- ***Phase 1: Cone exhaustive approach*** where we mark the fan in cone of each target path to arrive at the set of primary inputs where the logic values need to be enumerated.
- ***Phase 2: Apply enhanced functional sensitization (EFS)*** [31] conditions at side inputs of each gate along the target path and eliminate the target path if EFS conditions cannot be satisfied after implications.
- ***Phase 3: Apply High Delay Sensitization (HDS)*** conditions at each side input and perform implications to arrive at a mother test vector-space.
- ***Phase 4: Use side input refinement (SIR)*** algorithm using the partial order graphs for HDS conditions to refine side input values where each non-inferior refinement gives a vector sub-space.
- ***Phase 5:Use timing based pruning (TBP)*** approach for each vector sub-space to identify primary inputs where values must be enumerated.

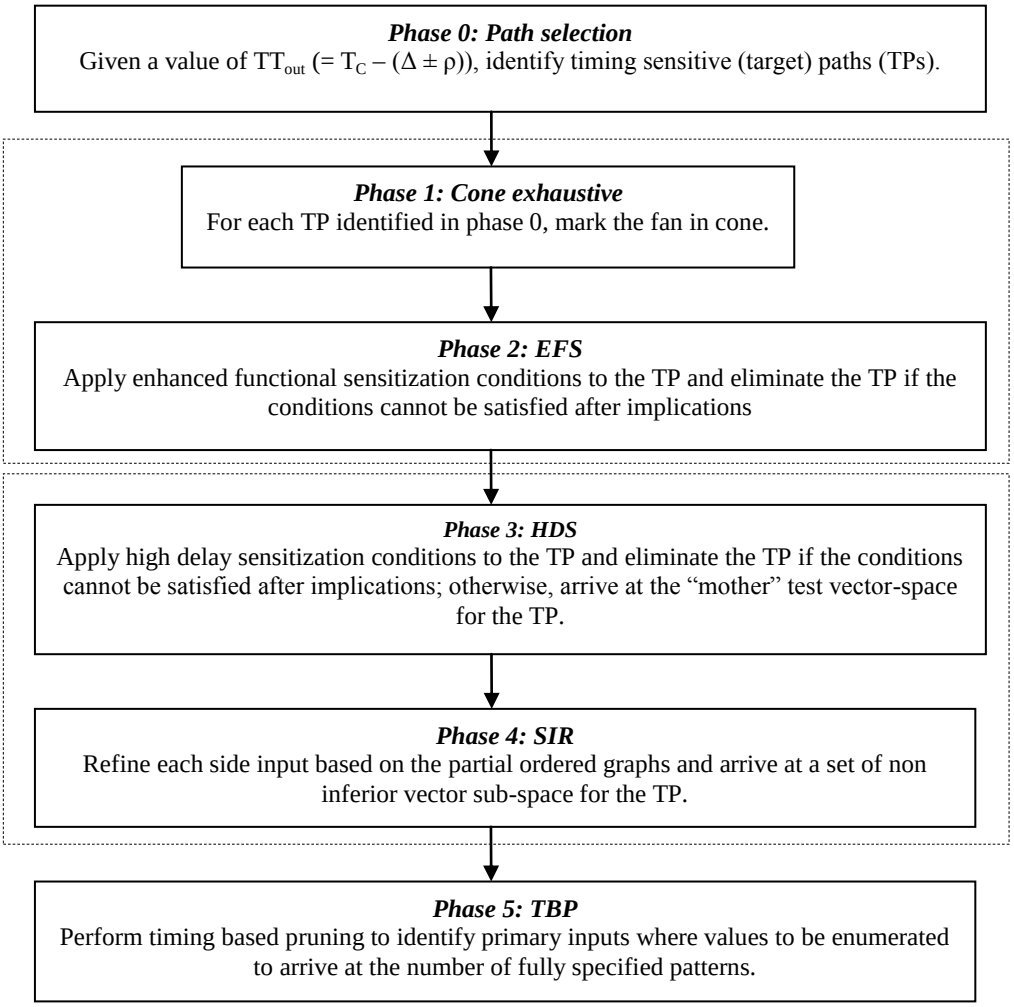


Fig. 13: An overview of our 6-phase approach

In this section we will illustrate the whole framework on the ISCAS89 benchmark circuit c17.

A. Cone exhaustive approach

Using the approach in [30] and our notion of ***significant marginalities (SM)***, i.e., those causing erroneous operations for significant proportion of fabricated chips, we perform ETA [31] and arrive at a set of target paths whose worst case delay is > T_c - Δ where T_c (0.448 ns) is the minimum clock period obtained from ETA [31]. (Recall that such a set of target paths that is guaranteed to include all paths that may potentially cause a timing error if the accumulated value of additional delays along circuit paths is upper bounded by Δ.) Then for each target path we identify the primary input cone and mark these inputs as *xx* and the rest as *dd*. Here *x* stands for {CR, CF, S1, S0} which essentially means *enumerate all*

possible combinations for these primary inputs whereas *d* stands for don't care. In Figure 14 we show the the logical path {R_3, R_7, F_9, F_{10}, R_{12}, R_{13}, F_{16}}. Since, the fan in cone has 4 primary inputs (circuit lines 1, 2, 3 and 4), at this stage the number of vectors needed for validation is $4^4 = 256$.

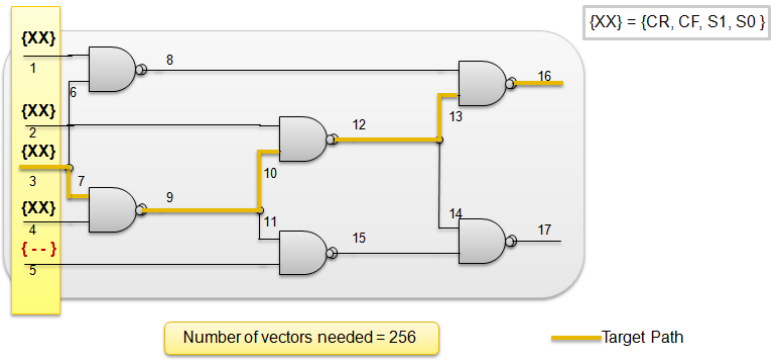


Fig. 14: Cone exhaustive approach on c17

B. Enhanced functional sensitization conditions

In [31] the authors have shown that the use of enhanced functional sensitization conditions helps to reduce the number of paths that must subsequently be considered for delay test generation. We use the EFS conditions as a baseline to demonstrate the benefits of our approach. We applied the EFS conditions and performed implications to reduce the targets to be considered further and arrive at a set of MV. The fully specified values at line 1, 2, 3 and 4 shows that the number of vectors needed for validation is 2*3*1*2 =12 as shown in Figure 15.

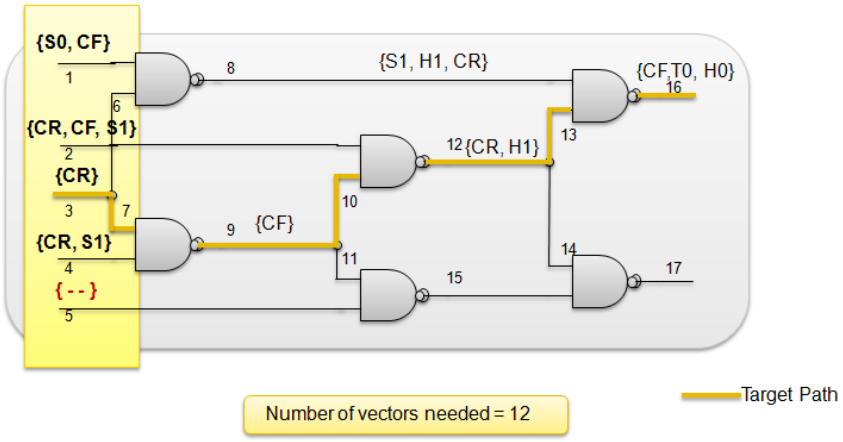


Fig 15: EFS approach on c17

C. High delay sensitization conditions

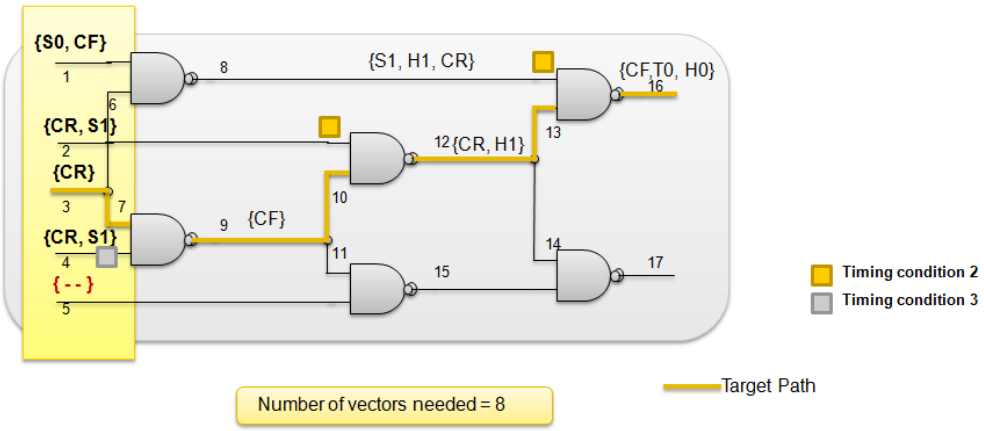


Fig. 16: HDS approach on c17

Now we apply our HDS conditions and perform implications to eliminate target paths for which necessary conditions cannot be satisfied after implications and thus we arrive at the mother test vector-space - the set of MV which is guaranteed to resiliently invoke the maximum delay for the target. The number of vectors needed for validation is 2*2*1*2 = 8 as shown in Figure 16.

D. Side input refinement and sub cube generation

Using the partial ordered graphs we refine the mother vector-space to arrive at the vector sub-space. Consider the state of circuit (lines and corresponding values assigned) after HDS as shown in Figure Fig. 16. We have three circuit lines (4, 2 and 8) which can be identified as side inputs. 2 and 8 belong to the timing case where the side input is early and near simultaneous but does not overlap with on path input. The corresponding partial ordered graph gives two ({CR}, {S1}) equivalent classes at line 2 and three ({S1}, {H1} and {CR}) equivalent classes at line 8. In this phase every side input assignment is refined to either one value or a single set of equivalent values. The corresponding search tree for c17 is shown in Figure 17.

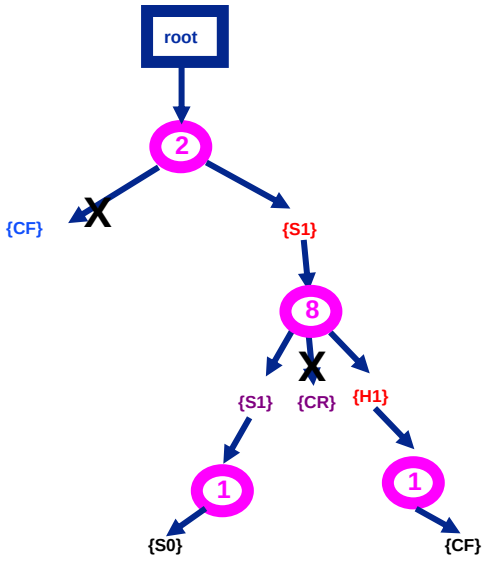


Fig. 17: Search tree for SIR on c17

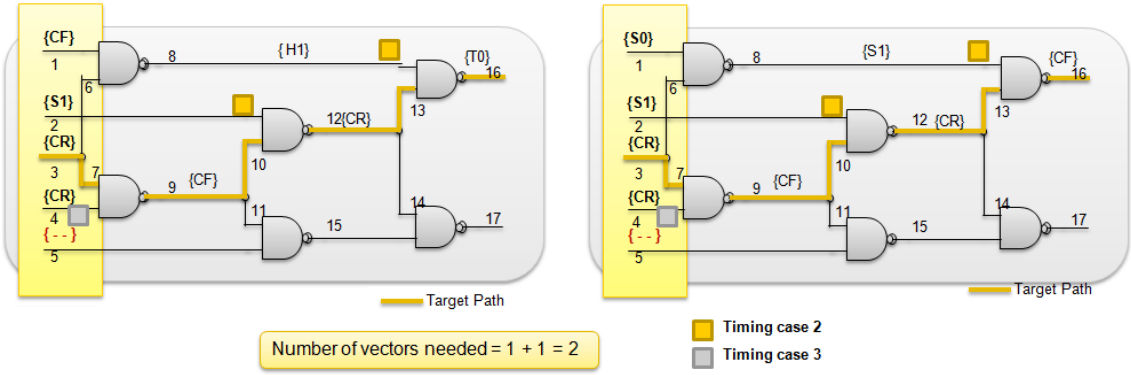


Fig. 18: SIR approach on c17

After SIR is finished we can see that we are left with only two leaf nodes thus giving rise to the two vector sub-spaces [{CF}, {S1}, {CR}, {CR}] and [{S0}, {S1}, (CR), (CR)] and each requiring just one vector to detect the target (also see Figure 18). The reason is because of implication conflict, two of the branches at intermediate node 2 are terminated whereas for intermediate node 8 each of the branches leads to a leaf node. Though the illustration shows a BFS approach we have implemented a more advanced DFS approach in our framework. Hence, at the end of this stage the number of vectors needed is 1*1*1*1 + 1*1*1*1 = 2.

E. Timing based pruning

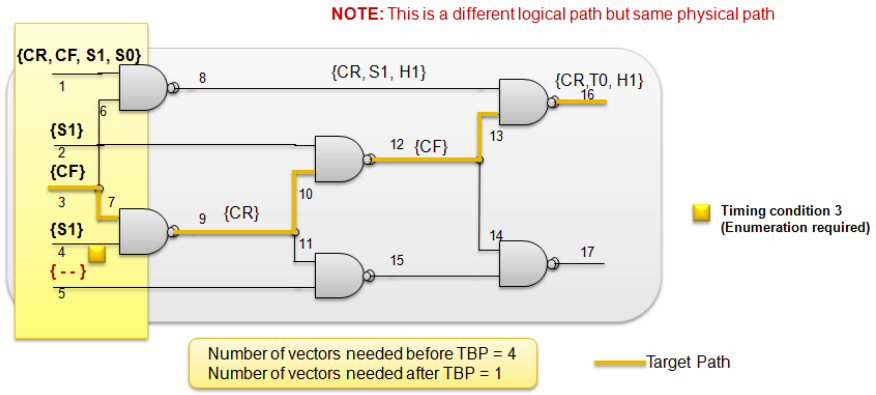


Fig. 19: TBP on c17

After SIR there is no more enumeration based on logic values as the values in singleton sets are all equivalent. Hence the objective now narrows down to identify side inputs where the values need to be enumerated because of timing conditions and by using these to back trace to identify the inputs with partially specified *(x)* values, where enumeration based on timing is required. We start with each side input where the timing case is either overlap or near-simultaneous transitions and recursively mark the fan ins and transitive fan ins with *timing matters* flag given the condition that the line does not contain any subset of static values {S0, S1}. At the end of this step only the primary inputs that are marked with the *timing matters* flag needs to be enumerated. For another logical path {F_3, F_7, R_9, R_{10}, F_{12}, F_{13}, R_{16}} as shown in Figure 19, the MV for validation have reduced from 4*1*1*1=4 to 1*1*1*1=1 after TBP (see Table VII).

Table VII: Values at Primary inputs after TBP

Line no. (PI)	Logic values	Timing Matters?
1	{S0, S1, CR, CF}	No.
2	{S1}	No.
3	{CF}	No.
4	{S1}	No

V. EXPERIMENTAL RESULTS

We have applied our approach to combinational parts of ISCAS89 benchmark circuits (see Table VIII) using a Intel Core 2 Duo 2.2 GHz machine. All gates in the benchmark circuits are assumed to use

minimum-size transistors, and a 65nm CMOS technology is assumed. Our experiments use a resilient simultaneous delay model for both to-controlling and to-non-controlling transitions [28].

Table VIII: ISCAS Benchmark circuits

ISCAS Benchmarks	No. of PI's	Logical paths	T_c (ns)	Type
S298	17	462	0.639	Traffic Light Controller
S953	45	2,312	1.142	Controller
S1196	32	6,196	2.462	Controller
S713	54	43,624	4.507	PLD

In our first set of experiments, we characterize the delays of individual gates only using the nominal delay values (zero variations). We then compute T_C as the maximum circuit delay using enhanced timing analysis [27]. Then using the variability values for the given 65nm technology, we estimate that the overall variability (primarily in threshold voltage, effective channel length, channel width and so on.) in each gate's delay is between 30% and 50%. Since in this set of experiments individual gate delay models do not capture variations, we capture variations by using Δ values between 30% and 50%. In phase 0 we use the above gate delay models (zero variability) and Δ values (30% to 50%) for target path selection.

In phase 1 for each target path identified in phase 0, we mark its fan-in cone. In phase 2 we apply EFS conditions and perform implications. The path is not functionally sensitizable if implication fails. Hence, functionally unsensitizable paths are identified without performing any search. This leads to a considerable reduction in the number of target paths. In phases 3 and 4 we respectively apply our new high delay sensitization (HDS) conditions and side input refinement (SIR) approach to obtain the sets of MV (*vector sub-spaces*) for each target path. Phase 5 uses timing based pruning (TBP), to reduce the number of fully specified patterns in each MV.

Table IX shows the results for number of target paths for three values of Δ for four benchmarks. The data for various values of Δ shows that the number of target paths rapidly increases as Δ is increased. The table demonstrates that the proposed approach identifies much fewer paths to target for every possible value of Δ. For example, for s713 and $\Delta = 0.50$ only 481 out of 43,624 logical paths can cause timing

errors at outputs if the cumulative value of additional delays along every path is upper bounded by Δ and whose worst case delay is greater than T_c - Δ.

Table IX: Analysis on ISCAS Benchmark circuits: Zero process variations in gate delay model, Δ values capture all model variations

	Δ	Phase 1			Phase 2			Phase 3			Phase 4, 5			After justification			Final vector spaces
		Paths	Vectors	CPU clocks	Paths	Vectors	CPU clocks	Paths	Vectors	CPU clocks	Paths	Vectors	CPU clocks	Paths	Vectors	CPU clocks	
S298	*0.3*	126	6.49×10^6	42	95	2.04×10^5	85	36	512,60	379	30	87	1,508	9	9	4,526	9
	0.4	272	1.04×10^7	60	220	3.57×10^5	155	106	1.21×10^5	866	92	2,816	3,724	33	104	8,537	33
	0.5	297	1.12×10^7	61	251	4.48×10^5	186	126	1.58×10^5	1,194	117	3,722	4,435	43	177	9,223	43
S953	*0.3*	486	8.52×10^{12}	206	310	3.10×10^{10}	4,054	267	1.11×10^{10}	6,255	182	11,114	1.06×10^5	3	65	1.12×10^4	3
	0.4	1,040	1.70×10^{13}	279	941	9.46×10^{10}	9,366	709	3.71×10^{10}	14,862	534	1.40×10^6	1.62×10^5	7	133	7.81×10^4	7
	0.5	1,398	2.20×10^{13}	362	1,099	1.21×10^{11}	12,352	977	4.74×10^{10}	19,128	853	1.60×10^6	2.05×10^5	47	29,914	1.67×10^5	47
S1196	*0.3*	973	2.48×10^{16}	354	587	5.87×10^{13}	8,839	419	3.83×10^{13}	13,785	258	1.41×10^7	2.83×10^5	12	44	1.49×10^5	12
	0.4	1,895	2.91×10^{16}	561	1,123	1.07×10^{14}	9,514	738	4.63×10^{13}	28,003	468	8.54×10^7	5.63×10^5	46	103,908	3.89×10^5	46
	0.5	2,739	3.48×10^{16}	751	1,809	1.91×10^{14}	26,975	1407	8.25×10^{13}	43,681	864	6.07×10^{10}	9.61×10^5	80	291,719	7.03×10^5	80
S713	*0.3*	32,638	1.07×10^{20}	6,840	254	2.96×10^{14}	46,202	138	1.01×10^{14}	42,842	65	8.87×10^{10}	2.83×10^5	7	65,535	3.91×10^5	7
	0.4	37,429	1.22×10^{20}	7,073	713	1.34×10^{15}	55,289	462	3.84×10^{14}	55,678	271	1.21×10^{12}	2.22×10^5	51	7.95×10^6	1.11×10^6	51
	0.5	38,840	1.33×10^{20}	7,469	910	2.91×10^{15}	55,347	760	7.09×10^{14}	55,986	481	1.59×10^{12}	3.19×10^5	95	1.80×10^7	8.43×10^6	95

Table X: Analysis on ISCAS Benchmark circuits: Full global process variations in gate delay model, Δ values capture other model variations

	Δ	Phase 1			Phase 2			Phase 3			Phase 4, 5			After justification			Final vector spaces
		Paths	Vectors	CPU clocks	Paths	Vectors	CPU clocks	Paths	Vectors	CPU clocks	Paths	Vectors	CPU clocks	Paths	Vectors	CPU clocks	
S298	*0.01*	142	7.39×10^6	55	100	182,392	85	44	57,656	383	40	4,967	2,987	2	64	4,179	2
	0.02	142	7.39×10^6	55	100	182,392	85	44	57,656	383	40	4,967	2,987	2	64	4,179	2
S953	*0.01*	366	7.15×10^{12}	163	366	4.90×10^{10}	669	193	1.04×10^{10}	4,417	150	604,219	60,819	2	2	79,376	2
	0.02	400	7.38×10^{12}	165	400	5.67×10^{10}	729	216	1.25×10^{10}	4,763	168	620,235	61,115	2	2	83,467	2
S1196	*0.01*	837	1.71×10^{16}	305	608	2.83×10^{13}	1,543	233	3.84×10^{12}	9,035	169	1.03×10^6	1.8×10^5	13	71	2.13×10^5	13
	0.02	947	2.34×10^{16}	355	696	7.91×10^{13}	1,726	317	1.39×10^{13}	11,233	232	5.78×10^6	1.9×10^5	18	936	3.07×10^5	18
S713	*0.01*	29,856	1.04×10^{20}	5,668	2,258	2.27×10^{14}	22,668	419	7.17×10^{13}	35,433	303	5.7×10^{10}	2.7×10^5	11	132,256	3.37×10^5	11
	0.02	29,856	1.04×10^{20}	5,668	2,258	2.27×10^{14}	22,668	419	7.17×10^{13}	35,433	303	5.7×10^{10}	2.7×10^5	11	132,256	3.37×10^5	11

Table IX also shows the total number of fully specified vectors (actually each is a two vector sequence) generated for each of the four circuits to guarantee invocation of the worst-case delay (Note that the total number of vectors reported here is obtained by simple addition for all the target paths). Also, out of 481 such timing critical paths, we could generate maximum delay tests for only 95 of them. (Note the ATPG

has a backtrack limit of 50). The results show that for a circuit like s713 and $\Delta = 0.50$ with our systematic approach the number of fully specified patterns needed for validation concentrating on delay marginalities can be reduced by a factor of 10^{13}, compared to baseline case of cone-exhaustive. We report the test generation time in CPU clocks and the final vector-spaces (to be used for efficient tester memory management) for each case as well.

In our second set of experiments, we characterize the delay for each gate using full global level of variability for the 65nm process (using a sufficiently larger number of Monte Carlo simulations). In these experiments, since the variations are incorporated in the delay models, we can use much smaller values of Δ as it must now capture only modeling errors.

The results for the four benchmarks are shown in Table X. For each circuit the trend with respect to increasing Δ values is as expected. More importantly, for all these circuits the number of vectors required to guarantee the invocation of the worst-case delay for the post-silicon validation of a design fabricated in a process with very high level of variability is very practical indeed. This is true because post-silicon validation is performed for a small sample of chips selected from first-silicon batch (unlike delay testing, which must be performed on every fabricated chip copy). Note that the test vector spaces generated by our approach can be further refined using test compaction methods.

Table XI: Robust V/s resilient

	Robust test vectors	**Our resilient test vectors**	**Improvement (%)**
S298	0.546 ns	0.626 ns	14.65%
S953	1.033 ns	1.132 ns	9.58%
S1196	2.19 ns	2.362 ns	7.85%
S713	4.197 ns	4.462 ns	6.31%

We have validated the resilient delay model along with the associated timing analysis framework against circuit level simulator (Spectre) using extensive simulations and Monte Carlo simulations. We used the same framework to calculate the delay for the most timing critical paths ($\Delta = 30\%$) for each benchmark using both robust test set and our test set (resilient vector-spaces) (Table XI). Table XI shows clearly that our resilient vector-spaces generate much higher delay than the classical robust tests. By cone-

exhaustive simulation for a small benchmark (as cone-exhaustive for larger circuits is impractical) such as s298, we validated that our resilient-vector space does contain the maximum delay sensitization vector (that invokes the delay of 0.626 ns).

Finally, let us consider how the vectors we generate can be used for our other intended post-silicon task, namely speed binning. Clearly, since speed binning is performed for every copy of the chip fabricated during high-volume manufacturing, it is impossible to apply such large number of vectors. We propose two approaches to remedy this situation. First, we can undertake more detailed quantitative version of the analysis we used to derive the partially-ordered graphs in Section III to order the delay invoked by alternative side-input values more precisely. This information can then be used to more precisely rank-order the vectors generated by our above approach and to select only the handful of vectors that excite the highest delays. Second, we can score the vectors applied during post-silicon validation in terms of their ability to invoke high delay values and select only those that invoke high delays. (We are in the process of undertaking experiments on actual silicon to illustrate this.)

VI. CONCLUSION

Experimental results show that our proposed framework along with timing dependent conditions can be used efficiently to guarantee detection of delay marginalities. The non inferior test vector-spaces generated by our algorithm further fortify our proposal for using MV to compensate for lack of exact knowledge and inaccuracies in delay parameters. We are currently working towards incorporating the effects of process variations in our approach. We will then be concentrating on approaches to exploit chip sampling using the knowledge of chip to chip vs on-chip variability to reduce the number of vectors for validation.

REFERENCES

[1] S. K. Gupta. “Validation of First-Silicon: Motivation and Paradigm.”

[2] J. Ryan Kenny, "Prototyping advanced military radar systems”, *Defense Tech Briefs*, March 2008.

[3] P. Woo, "Structured ASICs - A risk management tool", *D&R Industry Articles* (www.desgn-reuse.com).

[4] B. Vermeulen and N. Nicolici, "Post-silicon validation and debug", Embedded Tutorial (Session 10A), *European Test Symposium*, 2008.

[5] J. Yuan, K. Shultz, C. Pixley, H. Miller, and A. Aziz. "Modeling Design Constraints and Biasing in Simulation Using BDDs", In *Proceedings of International Conference of Computer Aided Design*, 1999, pp. 584-589.

[6] K. Shimizu and D. L. Dill. "Deriving a Simulation Input Generator and a Coverage Metric from a Formal Specification", In *Proceedings of Design Automation Conference*, Jun. 2002, pp. 801-806.

[7] K. Shimizu and D. L. Dill. "Using Formal Specifications for Functional Validation of Hardware Designs", In *IEEE Transactions on Design & Test of Computers*, Vol. 19, Issue 4, Aug. 2002, pp. 96-106.

[8] Y. K. Malaiya, M. N. Li, J. M. Bieman, and R. Karcich**.** "Software reliability growth with test coverage", In *IEEE Transactions on Reliability*, Vol. 51, Issue 4, Dec. 2002, , pp. 420-426.

[9] B. D. Cory, R. Kapur, and B. Underwood, "Speed Binning with Path Delay Test in 150-nm technology", In *IEEE Design & Test of Computers*, Vol. 20, Sept-Oct 2003, Issue 4, pp. 41-45.

[10] J. Zeng, M. Abadir, G. Vandling, L. Wang, A. Kolhatkar, and J. Abraham, "On correlating structural tests with functional tests for speed binning of high performance design", In *Proceedings. of International Test Conference*, 2004, pp. 31-37.

[11] L. C. Chen, P. Dickinson, P. Dahlgren, S. Davidson, O. Caty, and K. Wu, "Using Transition Test to Understand Timing Behavior of Logic Circuits on UltraSPARC^{TM}T2 Family", In *International Test Conference*, 2009, pp. 1-10.

[12] A. Rubio, N. Itazaki, X. Xu, and K. Kinoshita. "An Approach to the Analysis and Detection of Crosstalk Faults in Digital VLSI Circuits", In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Volume 13, Pages 387-394, Mar. 1994.

[13] F. Moll and A. Rubio. “Methodology of Detection of Spurious Signals in VLSI Circuits”, In *Proceedings of European Test Conference,* Pages 491-496, 1993.

[14] F. Moll and A. Rubio. “Detectability of Spurious Signals with Limited Propagation in Combinational Circuits”, In *Proceedings of Asian Test Symposium,* Pages 34-39, 1998.

[15] K. T. Lee, C. Nordquist, and J. A. Abraham. “Automatic Test Pattern Generation for Crosstalk Glitches in Digital Circuits”, In *Proceedings of VLSI Test Symposium,* Pages 34-39, 1998.

[16] R. Kundu and R. D. Blanton. “Timed Test Generation for Crosstalk Switch Failures in Domino CMOS”, In *Proceedings of VLSI Test Symposium,* Pages 379-385, May. 2002.

[17] W. Chen, S. K. Gupta, and M. A. Breuer. “Test Generation for Crosstalk-Induced Faults: Framework and Computational Results”, In *Journal of Electronic Testing: Theory and Applications*, Vol. 18, No. 1, pages 17-28, Feb. 2002.

[18] Y.-M. Jiang, K.-T. Cheng, and A. Krstic. “Estimation of Maximum Power and Instantaneous Current Using a Genetic Algorithm”, In *Proceedings of IEEE Custom Integrated Circuits Conference,* Pages 135-138, May. 1997.

[19] Y.-M. Jiang and K.-T. Cheng. “Exact and Approximate Estimations for Maximum Instantaneous Current of CMOS Circuits”, In *Proceedings of Design Automation & Test in Europe,* Pages 698-702, Feb. 1998.

[20] Y.-S. Chang, S. K. Gupta, and M. A. Breuer. “Test Generation for Ground Bounce in Internal Logic Circuitry”, In *Proceedings of VLSI Test Symposium*, pages 95-104, 1999.

[21] Y.-S. Chang, S. K. Gupta, and M. A. Breuer. “Test Generation for Maximizing Ground Bounce Considering Circuit Delay”, In *Proceedings of VLSI Test Symposium*, pages 151-157, 2003.

[22] Y.-S. Chang, S. K. Gupta, and M. A. Breuer. “Test Generation for Maximizing Ground Bounce for Internal Circuitry with Reconvergent Fan-Outs”, In *Proceedings of VLSI Test Symposium*, pages 358-366, 2001.

[23] S. Nassif. “Delay Variability: Sources, Impact and Trends”, In *Proceedings of Solid-State Circuits Conf.*, 2000, pp. 368-369.

[24] I. L. Chuang and M. A. Nielsen. "Quantum Computation and Quantum Information", Cambridge Series on Information, 2000.

[25] J. Zhang, N. Patil and S. Mitra, "Probabilistic Analysis and Design of Metallic-Carbon-Nanotube-Tolerant Digital Logic Circuits", In *IEEE Transactions on Computer Aided Design* , Vol. 28, Issue 9, 2009, pp. 1307-1320.

[26] L. C. Chen, S. K. Gupta, and M. A. Breuer, "A New Gate Delay Model for Simultaneous Switching and Its Applications", In *Proceedings of Design Automation Conference*, 2001, pp. 289-294.

[27] L. C, Chen, S. K. Gupta, and M. A. Breuer "Gate Delay Modeling for Multiple to non controlling stimuli." - unpublished

[28] P. Das and S. K. Gupta, "Capturing variability in delay models beyond pin to pin", USC technical report No. CENG-2010-3, 2010.

[29] Y. Cao, P. Gupta, A. B. Kahng, D. Sylvester, and J. Yang, "Design Sensitivities to variability: Extrapolations and assessments in nanometer VLSI", In *IEEE International ASIC/SOC Conference*, 2002, pp. 411 – 415.

[30] I.D. Huang and S. K. Gupta, "Selection of Paths for Delay Testing", *In Proceedings of Asian Test Symposium*, 2005, pp. 208 - 215.

[31] I. D. Huang, S. K. Gupta, and M. A. Breuer, "Dynamic Timing Analysis", USC technical report CENG-2004-06, 2004.

[32] L. C. Chen, S. K. Gupta, and M. A. Breuer. "High Quality Robust Tests for Path Delay Faults", In *Proc. VLSI Test Symp.*,1997, pp. 88 - 93.

[33] P. Franco and E. J. McCluskey. "Three- Pattern Tests for Delay Faults", In *Proc. VLSI Test Symp.*, 1994, pp. 452-456.

[34] J. C. Lin and S. Reddy. "On Delay Fault Testing in Logic Circuits", In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol 6, Sept. 1987, pp. 694-703.

[35] G. L. Smith, "Model for Delay Faults Based on Paths", In *Proc. International Test Conf.*, pp. 342-349, 1985.

[36] A. Pierzynska and S. Pilarski. "Non- Robust versus Robust", In *Proc. IEEE International Test Conf.*, 1995, pp. 124 – 131.

[37] S. Bose, P. Agrawal, and V.D. Agrawal. "Logic System for Path Delay Test Generation," In *Proc. European Design Automation Conf.*, 1993, pp. 200-205.

[38] S. Devadas, "Validatable Non robust delay-fault testable circuits via logic synthesis," In *Proc. IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, 1992, vol 11, issue 12, pp. 1559-1573.

[39] A. Kristic, and K. T. Cheng. "Generation of High Quality Tests for Functional Sensitizable Paths," In *Proc. VLSI Test Symp.*, 1995, pp. 374-379.

[40] L. C. Chen, S. K. Gupta, and M. A. Breuer, "A New Framework for Static Timing Analysis, Incremental Timing Refinement, and Timing Simulation," In *Asian Test Symp.*, 2000, pp. 102-107.

[41] S. Irajpour, S.K. Gupta, and M. A. Breuer, "Timing-Independent Testing of Crosstalk in the Presence of Delay Producing Defects Using Surrogate Fault Models", In *International Test Conf.*, 2004, pp. 1024-1033.