The Dissertation Committee for Rajeshwary G. Tayade
certifies that this is the approved version of the following dissertation:

Incorporating the Effect of Delay Variability in Path Based Delay Testing

Committee:

Jacob Abraham, Supervisor

Michael Orshansky

David Pan

Sani Nassif

Linda Reichl

Incorporating the Effect of Delay Variability in Path Based Delay Testing

by

Rajeshwary G. Tayade, B.E., M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2009

Dedicated to my family.

Acknowledgments

First and foremost I would like to thank my advisor Dr. Jacob Abraham. He has been a big inspiration to me and without his guidance, this dissertation could not have completed. I would also like to thank all my committee members especially Dr. Sani Nassif and Dr. Michael Orshanky who have given me valuable feedback and technical guidance throughout my work. My colleagues at the CERC laboratory namely Ramtilak Vemu, Sriram Sambamuthry, Hong Joon Shin, Joon Sung Park, Ji Seon Park, Xio Pu, Minsik Cho, Sankarnarayan Gurumurthy, Ramyanshu Datta and Shobha Vasudevan have always been very helpful. They always provided a healthy competitive spirit and engaged in intellectually stimulating discussions which made the four years of my graduate school very interesting. In addition to my collegues, I would like to sincerely thank our staff at CERC, including Andrew Keishnick, Melanie Gluick, Debi Prather and Melissa Compos who have been of constant help and very efficient in resolving all administrative issues.

I would not have been able to enjoy my years in graduate school without the constant moral support and encouragement provided by husband Anis Abdul. He has always provided my with valuable advice and helped me maintain the focus towards my research.

And above all I would like to thank my parents and my brother, who have always encouraged me in all my endeavors. Their love and support has always been a big moral support and a source of motivation for me.

Incorporating the Effect of Delay Variability in Path Based Delay Testing

Publication No. _____

Rajeshwary G. Tayade, Ph.D.
The University of Texas at Austin, 2009

Supervisor: Jacob Abraham

Delay variability poses a formidable challenge in both design and test of nanometer circuits. While process parameter variability is increasing with technology scaling, as circuits are becoming more complex, the dynamic or vector dependent variability is also increasing steadily. In this research, we develop solutions to incorporate the effect of delay variability in delay testing. We focus on two different applications of delay testing.

In the first case, delay testing is used for testing the timing performance of a circuit using path based fault models. We show that if dynamic delay variability is not accounted for during the path selection phase, then it can result in targeting a wrong set of paths for test. We have developed efficient techniques to model the effect of two different dynamic effects namely multiple-input switching noise and coupling noise. The basic strategy to incorporate the effect of dynamic delay variability is to estimate the maximum vector delay of a path without being too pessimistic.

In the second case, the objective was to increase the defect coverage of reliability defects in the presence of process variations. Such defects cause very small delay changes

and hence can easily escape regular tests. We develop a circuit that facilitates accurate control over the capture edge and thus enable faster than at-speed testing. We further develop an efficient path selection algorithm that can select a path that detects the smallest detectable defect at any node in the presence of process variations.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Delay Test Background

The performance and reliability of circuits in nanometer technology is determined by various factors such as parameter variability, defect distribution and operating conditions. One of the most important performance parameters is the circuit timing, or the maximum frequency at which a circuit can operate. Delay tests are parametric tests that are targeted for detecting any possible timing failure in a circuit. Since, many different defect types can affect circuit delays, delay testing can also be used for testing circuit reliability. In fact, delay test implementation and design is highly dependent on the test objectives being targeted. The three primary test objectives are the following.

- ***Defect Based Tests***: Defect based testing involves characterizing the failure modes of different type of manufacturing defects and identifying the best testing strategy for detecting these defects. Several defect analysis and fault diagnosis experiments have shown that stuck-at tests are not enough to get acceptable defect coverage [11], [26]. Different parametric tests have been developed to detect the defects that are not modeled by stuck-at faults. These tests include I_{ddq} testing [67], Very-Low-Voltage (VLV) testing [32], Min-Vdd testing [73] and delay testing. In [9], it was shown that no single test method is enough to cover all defect types. For instance, I_{ddq} tests can be ineffective in detecting stuck-open and resistive open defects [49] [61], but delay

testing can detect these defects. Thus delay testing can be targeted towards detecting defects that are not covered by stuck-at tests and other parametric tests.

- ***Test for Timing Performance***: Rather than designing tests to detect defects of a certain class, delay tests can be used to test the timing performance of a circuit. Thus, here the test objective is to check if the circuit under test meets timing specification in all possible operating conditions. The timing failures could be manifestations of actual defects or process parameter variations. Traditionally, functional tests have been used to check the timing performance of complex integrated circuits. For microprocessors, this consists of either running specific applications or carefully derived instruction sequences at functional speed during test mode. With the current trend toward developing systems on a chip, designs are becoming increasingly complex, making it extremely challenging to develop good and dependable functional tests. Another problem of using functional tests is that there is no formal measure to quantify the coverage and effectiveness of the test. Delay tests are structural tests and hence test generation can be automated based on different fault models and test quality can be quantified using various coverage metrics. By definition, this type of delay test is not targeted towards any particular type of defect. Any defects that are large enough to cause timing failure should be detected, and defects smaller than that will be ignored.

- ***Speed Binning***: The maximum frequency at which a defect-free circuit can operate with correct functionality varies due to variations in process parameters during fabrication. Instead of discarding the slower circuits they can be sold at a lower price with a different frequency specification. The process of sorting devices into various frequency bins based on the maximum frequency at which it can operate correctly is

called speed-binning. Typically, speed-binning is done by running carefully designed functional tests on the target circuit and then sweeping the operating frequency to find its frequency bin. Recently, structural tests based on path based delay fault models

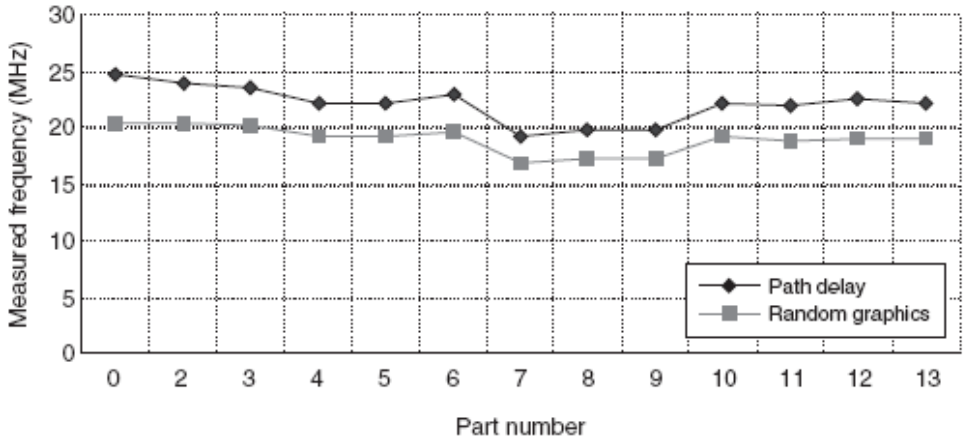


Figure 1.1: Speed-binning using structural Vs functional tests [23]

have been used for speed-binning [23], [81]. It was observed in [23] that results obtained using path delay tests followed the same trend as those obtained using functional tests, which shows that structural path delay tests have the potential for removing the dependency on functional tests for speed-binning.

1.1.1 Delay Fault Models

Various delay faults models have evolved over time, targeting different test objectives. One of the earliest delay fault models is the *Transition Delay Fault model* [21], which can detect point defects that can cause a node slow-to-rise or slow-to-fall. Such defects can be detected only by applying a transition at the target node and hence cannot be covered by stuck-at-tests. This model assumes that the defect is large enough to be detected independent of the path used for fault propagation. The *Transition Delay Fault Model* thus is targeted

towards gross point defects and does not model distributed delay defects which can affect a circuit path. Therefore, this fault model cannot guarantee detection of timing failure that result from the sum of several small delay defects. The size of the delay defect is not modeled and the fault coverage is based on the percentage of circuit nodes covered.

The *Gate Delay Fault Model* [41] is similar to the *Transition Delay Fault Model* except that the circuit delays are taken into consideration and the delay fault size is specified. The number of faults is linear in the number of gates in the circuit but this fault model also does not cover distributed defects.

The *Segment Delay Fault Model* [38] provides more flexibility on the target fault size. The fault can range from a spot defect to a distributed defect depending on the length of the segment selected.

In the *Path Delay Fault Model*, [70] a circuit is considered faulty if any path in the circuit exceeds specified delay. Thus it accounts for all defects types and defect sizes that can cause timing failures. Tests based on path delay fault model can detect small distributed defects and failures due to excessive process variations. The main challenge here is that the number of paths in most modern circuits is very large, so for efficient testing it is important to select a small subset of paths which when tested can maximize the probability of defect detection. One method is to select all paths with delays that exceed a certain threshold. The basic premise here is that smaller size defects are more likely to occur than large defects and paths with longer delays are more likely to fail timing due to the small margins available. The threshold for path selection can be selected based on the defect size distributions observed during manufacturing. The limitation of this technique is that the number of faults can increase drastically as the threshold is reduced and, at the same time the selected paths could have a low node coverage. Thus even a large delay defect on an uncovered node will

go undetected if none of the paths through the node exceed the threshold.

The *Line Delay Fault Model* [55] is a path based delay fault model that selects the longest testable path through each node for test. Thus the number of faults is linear with the number of circuit nodes. Also, the probability of defect detection is maximized by selecting the longest path, since the longest paths are most likely to fail for any defect size at a node.

Both the *Path Delay Fault Model* and the *Line Delay Fault Model* only target defects that cause a circuit to fail timing specification. A delay defect on a node will not be detected by these fault models if it does not cause the selected test paths to exceed the timing specification. However, such defects can affect circuit reliability which can result in field failures. Such defects are called *Small Delay Defects* and are becoming increasingly important in nanometer technologies.

1.2 Challenges in Delay Testing

Test automation for structural tests such as stuck-at test is a fairly mature process. A typical methodology is shown in Figure 1.2. The challenges in developing an efficient and effective delay test methodology are much different that those for stuck-at tests. Unlike stuck-at tests, where fault listing can be obtained directly from the nodes in the circuit netlist, path based delay testing requires complex path selection algorithms, with accurate timing models for fault listing. Path selection is based on the fault model being used. For instance, the threshold based model requires selecting all paths above a certain threshold value, while the line delay fault model requires selecting the longest path through each node.

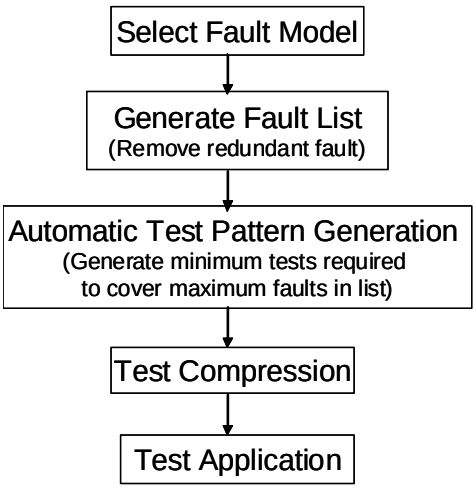


Figure 1.2: Structural test methodology

1.2.1 Path Selection

In traditional structural test methodology, test generation is a separate step that follows fault listing, and fault coverage is measured as the percentage of listed faults for which a test was found. For path based delay testing however, path selection and test generation cannot be completely independent. This is because most circuits of reasonable size have a lot of false paths, i.e. paths which cannot be sensitized by any vector. If path selection and test generation are completely independent, then a lot of false paths will be selected during the fault listing step, making the process very inefficient. Several path selection algorithms [13], [34] have shown that including false path elimination techniques during path selection improves the run times significantly.

Other than efficient handling of false paths, one of the biggest challenge in path selection for delay testing is the variability in circuit delay introduced due to the inherent process parameter variation. In an ideal case, if all manufactured circuits are exactly identical, then the timing performance test of the circuit can be performed by simply testing the critical

(longest delay) path in the circuit. However, due to delay variability, each manufactured part could have a different critical path. Thus for timing performance test, path selection algorithms need to select all possible paths that can be critical. In the threshold based path selection technique, this would require selecting all paths that have a certain probability of exceeding the target delay. However, deciding on a good threshold is still a non-trivial problem. A similar issue is present in the *Line Delay Fault Model*, where it becomes difficult to identify the longest path through each node, since different paths could be maximal in different parts. If a conservative approach is taken in path selection, then the path set can increase drastically, since in a timing optimized synthesized circuit there could be many near-critical paths [78]. Thus, for the purpose of timing performance test, the problem of path selection is to identify the smallest set of paths such that if these paths meet timing during test, then it is guaranteed that the circuit meets timing.

1.2.2 Test Generation

Once the path set for test is selected, the test generation process needs to find vectors for sensitizing each path. Each vector that can sensitize a path can introduce a different amount of delay on the path. This dynamic or vector dependent delay variation is referred to as dynamic noise. If the test objective is to find if a circuit meets timing performance, then the selected test paths should be tested at their worst possible delays. The test generation process should therefore try to find a test vector that maximizes the delay of any target path. This requires Automatic Test Pattern Generation (ATPG) tools to model the different dynamic delay effects and have in-built search algorithms that find the optimal test vector. Due to the large search space involved, finding the worst case vector is a non-trivial problem. In [42] and [44], genetic algorithm based search techniques are used for generating tests that

maximize the vector dependent delay of paths. However, in general obtaining high quality test vectors will require detailed timing models or dynamic simulations which can be very costly in terms of run time.

1.2.3 Practical Challenges

One of the earliest challenges in successfully applying delay test was to generate the required transitions at the fault sites. Generating a transition requires two vectors to be applied successively, without changing the system state in between. Various solutions have been developed in the past to make this possible. The simplest scheme is to add an additional latch to each flop, which holds the value from the first vector while the second vector is being scanned in. This scheme is called the *Enhanced Scan* scheme. Another approach is to make use of sequential logic, and use the results captured from one stage as the transition vector for the following stage. This is the *Launch on Capture* scheme. Yet another technique is to obtain the second vector by shifting the first vector by a single bit position. This is called the *Launch on Shift* scheme. Thus in delay testing, the test generation needs to take into account the scan scheme being used. The fault coverage and the efficiency of the test generation algorithm is dependent on the type of scan scheme being used. This makes delay test generation a much more challenging task compared to other structural tests.

For delay testing to be effective in any of the above target applications, the basic requirement is that the test should be run at the rated clock frequency [40]. This was feasible in older technologies where the Device Under Test (DUT) operating frequencies were lower than the Automatic Test Equipment (ATE) frequencies. However, due to the aggressive scaling of CMOS devices, the DUT frequencies are increasing steadily, while ATE frequencies are not increasing at the same rate. Even if faster ATE clocks are made available,

factors such as tester skew, pad delays and internal clock insertion delay, pose challenges in providing the required test clock frequencies [64].

1.3 Research Motivation

In this research we focus on path based delay faults, since these faults model both point and distributed delay effects. The overall objective is to provide techniques that will enhance the effectiveness of delay testing considering the various challenges described in the previous section. The following sub-sections describe different focus areas which were explored as a part of this research.

1.3.1 Impact of Dynamic Variations

One of the biggest challenge in successfully employing structural path delay tests as timing performance tests or for speed-binning is identifying the optimal set of paths that need to be tested. Paths that are more likely to fail are considered as critical. Depending on the fault model employed, critical path selection involves either comparing path delays with a threshold value [72] or ranking paths based on their delays [65]. Thus, the effectiveness of any path selection algorithm depends on the accuracy with which the path delays are estimated. It has been observed that the critical paths identified using pre-silicon tools rarely match the critical paths found in silicon [47]. This is because path delays are subject to variation from various sources, and the timing models used for delay estimation fail to capture this variation accurately. Sources of variation include device parameter variability, such as L_{eff}, V_t, etc, as well as dynamic variations such as supply noise, crosstalk or multiple input switching noise as shown in Figure 1.3.

Process parameter variations can result in different paths being the maximum delay

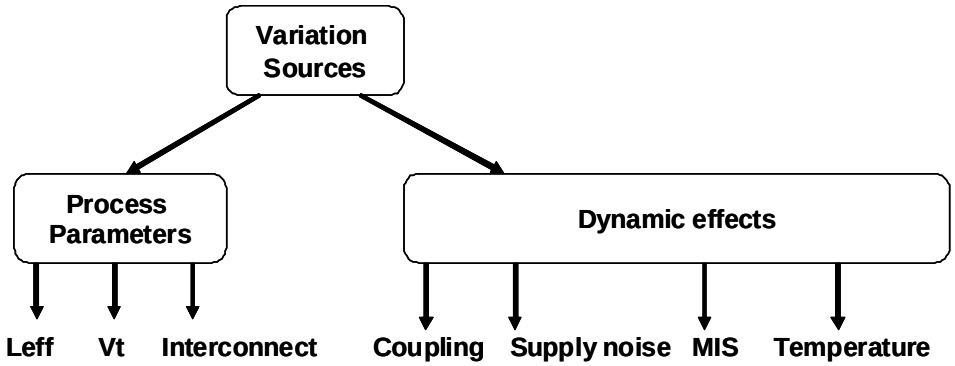


Figure 1.3: Various sources of delay variability

path in different dies. Thus path delay can be modeled as a random variable to represent the uncertainty due to process variations. Dynamic variations, on the other hand are vector dependent, and not random. Each vector that can sensitize a path will introduce a different amount of dynamic delay and hence path delay varies from vector to vector. Dynamic delay variability can also depend on the previous states or history of the circuit operation. For instance a certain sequence of operations in a circuit can cause large amount of switching, which leads to high supply grid noise and increased temperatures, which in turn can cause a given path to have much longer delays that the nominal value. An illustration of how a path delay can vary due to both process variations and dynamic effects is shown in Figure 1.4.

While previous research [77],[53] has addressed the problem of path selection in the presence of process variations, the effect of dynamic variability on path selection has not received much research focus. Since path delay is vector dependent, the set of paths that are identified as critical paths during the path selection phase depends on the vectors assumed for estimating the path delays. Most path selection algorithms published in the past do not take this factor into consideration when estimating path delays. Typically, path delay is

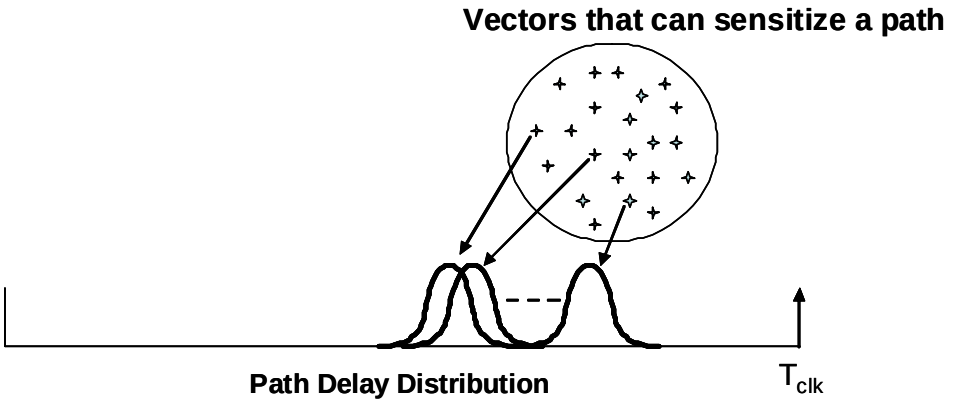


Figure 1.4: Effect of dynamic and process variability

estimated by simply adding the edge delays on the path, thereby incorrectly estimating the delay, which can result in missing some real critical paths. We believe that failure to model these dynamic variations is one of the reasons that critical paths observed on silicon often do not match the ones reported by timing verification tools. Prior works, such as [44] and [42], which have considered the dynamic or vector dependent delay variation of paths, focus only on the test generation part, while assuming that the critical paths are known.

In this research we emphasize the fact that for delay tests to be effective, it is important to first select the real critical paths for test, and hence dynamic delay effects need to be incorporated during path selection itself. The delay of a path is dependent on the vector used to sensitize it, but the test vector will not be known until the test generation phase. Thus there is circular dependency between path selection and test generation, in that test generation can happen only after paths are selected, but during path selection the vectors need to be known for accurately estimating the path delay. A simple solution to account for dynamic variability is to estimate the maximum vector delay of any path during path selection. Consider the example shown in Figure 1.5 where the delay distributions of two

paths (considering both process and dynamic variability) are shown. If the two paths are compared at their nominal delay, then the delay of P1 is larger than that of P2, but in the worst case P2 has a larger delay. Thus a critical path selection tool should rank P2 as more critical than P1. The problem then would be to estimate the worst vector delay of a path

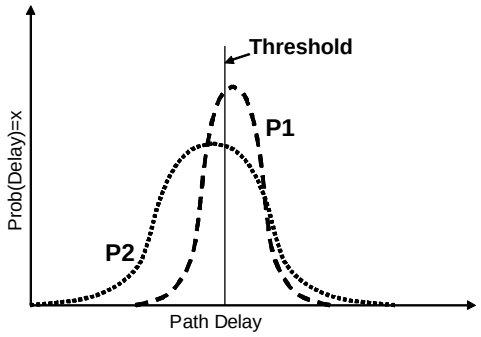


Figure 1.5: Path criticality

without being too pessimistic. Ideally, this would require estimating the total impact of all possible vector dependent delay variation sources. However, to make the problem tractable, it is simpler to analyze different sources separately. In this research, we focus on two such dynamic delay effects. In Chapter 2, we analyze the effect of Multiple-Input Switching (MIS) and develop analytical models that can be employed for incorporating the dynamic delay variation due to MIS for effective delay test generation. Chapter 3 describes an efficient technique to estimate the maximum delay of a path in the presence of coupling noise and its application in critical path selection for delay test.

1.3.2 Measuring Path Slack

The current delay test paradigm is to fix the test clock period based on the target frequency and check if the delays of the selected test paths are within the test clock period. This test strategy works if there is a guarantee that the selected test vectors introduce the

worst possible delays in the circuit. Given the large space of vectors that can sensitize any path, finding the optimal test set is a formidable challenge. Also, some dynamic effects such as supply noise depend on the previous states of the circuit and hence are difficult to reproduce during test. Thus, even though a test path meets timing during test, it is possible that during normal operation, the path delay could increase beyond the available margin due to excessive supply noise or temperature causing a timing failure. A potential solution to handle the uncertainty introduced by dynamic variations is to change the test approach. Instead of just checking if paths meet timing, delay test should involve measuring the delay margin (slack) available on paths during test as shown in Figure 1.6. A circuit can be said to meet timing specification if the available slack on paths is greater than a pre-determined value.

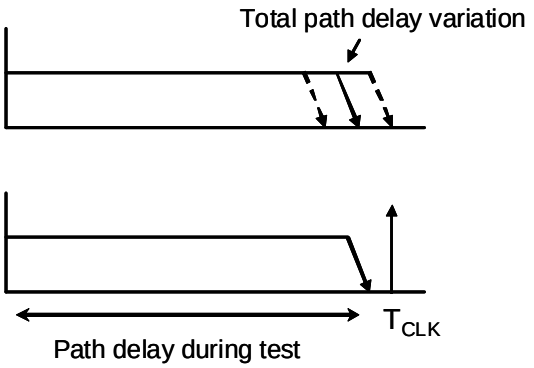


Figure 1.6: Path slack is not large enough

In this research, our objective was to design a circuit that can be used measure the delay of any path during test. In [24], a scan based vernier delay line called $MVDL$ was proposed, which can be used to measure the delay (and hence the slack) of a given path in the DUT. However, multiple copies of this circuit need to be placed in the DUT and each instance can be shared among only a few target paths. Also the MVDL needs to be placed

in close proximity with the target paths for accurate results. This incurs additional area overhead and still does not provide the capability of measuring the delay of all paths in the circuit. In Chapter 4, we discuss a new circuit scheme that provides accurate control over the capture clock frequency. It is a single control circuit that modifies the system clock and hence can target any path. The delay of any path can be measured by sweeping the capture frequency till the path fails timing. Multiple paths can also be targeted simultaneously. By providing a capture clock with programmable delay, we also facilitate at-speed and faster and at-speed testing of high performance DUTs using low cost testers. This also has direct application in detecting small delay defects as described in the next section. In Chapter 4, we discuss various applications of our circuit and also discuss how this technique can be employed in various scan schemes used for delay test.

1.3.3 Small Delay Defect Detection

Small delay defects are the defect types that do not cause enough delay increment to cause timing failures, but can affect circuit reliability and hence are undesired. Such defects, for instance, are likely to cause failures in the field, resulting in a increased field return failure rate, typically measured in *Defects Per Millon (DPM)* [61]. Field failures prove to be extremely expensive and hence, for high volume integrated circuit manufacturers, the target DPM is in the range of a few hundreds. Traditionally, reliability screening has been done using a process called burn-in, which involves running tests at high voltage, temperature and pressure conditions, causing unreliable parts to fail by accelerating the failure modes. Burn-in is a very expensive process, and it has been argued that burn-in might damage the dies due to excessive stress conditions [76]. Studies also suggest that burn-in might not be very effective in screening reliability defects in future technologies due to scaling down of

supply voltage [62]. Hence there is an increasing interest in finding alternative reliability screening procedures which can reduce or eliminate burn-in.

Delay tests that can detect defects that affect circuit reliability can potentially be used as a pre-burn-in reliability screen. Dies can be divided into three bins: 1) Defective, 2) Good but unreliable, and 3) Good and reliable. Dies for which small-delay defects are detected go in the second bin and should be sent in for burn-in. The chips in the third bin which have low probability of failure can be skipped from burn-in or sent for shorter burn-in cycles. Dies that have gross defects go in the first bin and need to be discarded. A similar approach for reducing burn-in cost was described in [79], where reliability information was extracted from wafer probe test to bin the dies with different burn-in failure probabilities.

Since there is no well defined fault model for reliability tests, the effectiveness of these test should be determined by the size and type of defects being detected. Thus, for delay testing to be effectively used as a reliability screening test, it should be able to detect very small delay changes. The primary limitation of at-speed testing is that the size of defect being detected is limited by the slack on the path as shown in Figure 1.7. For instance, if a the longest path through some node in the DUT has a large slack, many possible reliability defects can go undetected at that node.

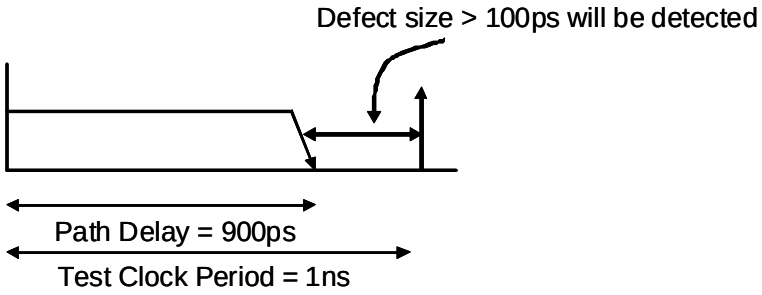


Figure 1.7: Detected defect size limited by path slack

Several research efforts in the recent past have shown how faster than at-speed testing can be used for small delay defect detection [79],[4]. However, most of these works still use the path selection strategy applicable to timing performance test, thus selecting the longest paths in the circuit. If delay test is done at rated clock frequency, then the smallest defects will be detected by the longest paths. However, if faster than at-speed testing is being done then it is important to consider the effect of process variations to determine if the delay increment observed on a path is due to random process parameters or due to the presence of small defects. The former case implies that it is a good chip running at a slow process corner, while the later means that it has a reliability defect. In this research, our objective was to develop a path selection algorithm that can be used for faster than at-speed tests. We analyze the effect of process variation on defect detection probability and then develop an efficient algorithm for maximizing the defect coverage. In Chapter 5, we analyze resistive interconnect defects in this context and show that long paths need not be the optimal paths for small delay defect detection. In Chapter 6, we discuss an efficient path selection algorithm that maximizes defect coverage when multiple fast test clock frequencies are present.

Chapter 2

Modeling the Effect of Multiple Input Switching

2.1 Multiple Input Switching Effect

Static Timing Analysis (STA) is the most common method for timing verification of digital logic circuits in the industry today. As circuit complexity increases, it becomes infeasible to check circuit timing by simulating all possible input combinations. STA tools work on the principle that if the maximum signal arrival time at any sequential or pin in the circuit is within the target value, then the circuit meets timing. Thus timing verification becomes a vectorless process and does not require extensive timing simulations. STA algorithms use a graph representation of the circuit and perform *Breadth-First-Search* based graph traversals for obtaining the maximum and minimum signal arrival times at each node. The gate level timing information is obtained from a characterized cell library which lists (or contains models for) the propagation delay of each timing arc of each gate as a function of different parameters. Each input to output path of a gate forms a timing arc. The timing simulations for characterizing the pin-to-pin delays of each gate typically assume that signal switching is happening only on the input being characterized while the other inputs are at a stable value. This is referred to as the Single Input Switching (SIS) delay of the timing arc. In reality, the propagation delay of any timing arc is affected by the signal switching at the other inputs of the gate. Thus, assuming that the other inputs are at stable values introduces an error in the delay estimation process. This estimation error in timing analysis is referred to as the Multiple Input Switching (MIS) error.

A signal transition at the input of any basic logic gate can be categorized as a *Controlling to Non-Controlling* (CTN) or *Non-controlling to Controlling* (NTC) depending on its effect on the output. For any gate, the controlling value is the one that enables one of the parallel current paths, either pull-up or pull-down, to the output node. In a NAND gate, for instance, since the pull-up network has parallel paths, the controlling value is 0. Hence a rising transition is a CTN transition while a falling transition is a NTC transition in a NAND gate. Thus, for all basic logic gates, one of the transition (rising or falling) is CTN while the other is NTC. An XOR gate is an exception, since both 0 and 1 are controlling values. If all the four possible states, Rising, Falling, Stable-at-0, Stable-at-1 are considered for an n-input logic gate, then the gate delay needs to be characterized for 4^n possible combinations. This can be reduced by considering the type of propagation. For instance, cases where one input is switching and other is at a stable controlling value can be eliminated since there is no signal propagation. When only one of the inputs is switching and all the other inputs are stable at the non-controlling value, the pin-to-pin delay can be called the SIS delay. If the inputs are switching in opposite directions, then either there is a glitch at the output (if the CTN transition arrives before the NTC transition), or no signal is propagated to the output. Figure 2.1 shows the possibilities for any 2-input gate. For an XOR gate, since both 0 and 1 values are controlling values, when multiple inputs are switching there is always the possibility of glitch. Glitches are undesirable signal propagations, when the output is required to be stable and will cause delay variation in the fanout stage. Thus the only cases for which MIS delay needs to be considered are when more than one input make transitions in the same direction. Figure 2.1 shows a chart of all possible cases for a 2-input NAND gate.

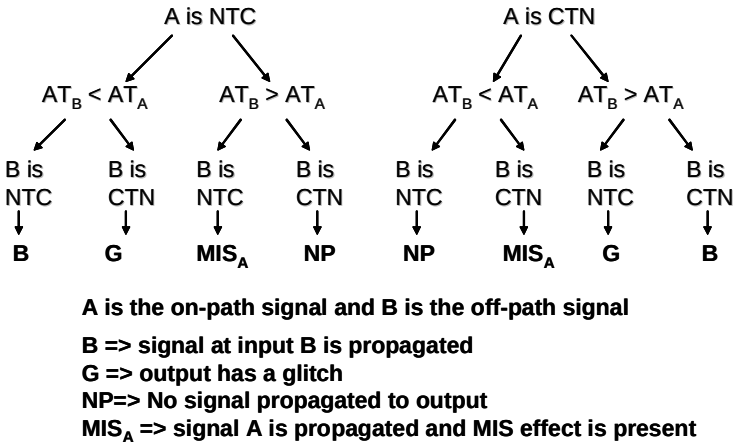


Figure 2.1: MIS cases for NAND2 gate

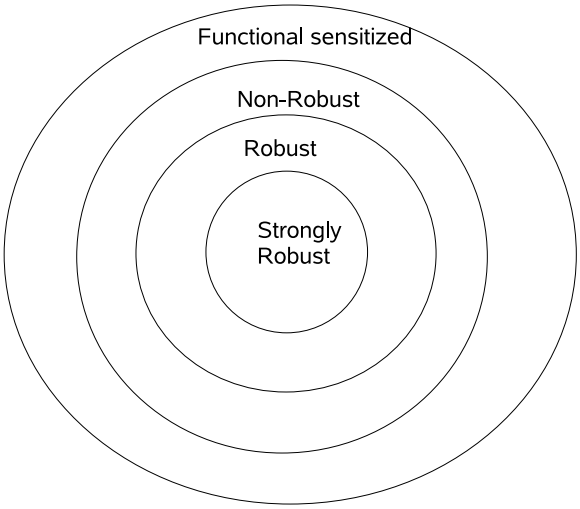
2.1.1 Prior Work

The effect of Multiple Input Switching on gate propagation delays has been studied before. A technique to convert any logic gate into an equivalent inverter to estimate the peak supply current is shown in [59], with the idea that gate delays can be estimated once the current waveform is known. However it is not clear how this model can be directly used in STA or statistical STA. In [12], the MIS effect is modeled as an error on the *Single Input Switching* (SIS) delay. Their technique consists of first selecting the dominant input and applying a delay macro-model to determine the delay w.r.t. the dominant input and further discuss a method to apply their technique for multiple input gates. In [18], a gate delay model is obtained as a function of input slews and transition times by using curve fitting to simulation results. More recently, in [6], the MIS effect has been captured by building multi-port current source models for each gate; while in [71], the MIS delay of complex gates is modeled using a high dimensional model representation. With timing analysis moving to the statistical domain to avoid conservative designs, the MIS delay effect has also been studied for statistical static timing analysis in [1]. The MIS effect has however largely been evaded

due to the added complexity in delay characterization. In this research, our objective is to derive a simple model for basic logic gates that does not require extensive characterization and uses the information from the available timing libraries.

2.1.2 Need for MIS Modeling in Delay Testing

Test vectors generated for delay testing can be classified based on the node sensitization criteria used for each path. A test vector is considered a robust delay test for a path if it can detect a defect on the path irrespective of the delays on the side input paths. The robustness criteria requires that a delay fault on a target path should not be masked by a fault on any other path. Based on this criteria, the set of vectors that can sensitize the path can be categorized as 1) *Strongly Robust*, 2) *Robust*, 3) *Non-Robust* and 4) *Functional*. A *Strongly Robust* test requires that transition be propagated only along the nodes on the path, while the side input nodes remain at stable (non-controlling) values. Thus for *Strongly Robust* vectors, path delay can be accurately estimated by simply adding the SIS delay of each edge on the path. For a *Robust* test, if an on-path node has a NTC transition, then the off-path side inputs need to be stable at the non-controlling value. If a on-path node has a CTN transition, then the off-path side inputs can either be stable and non-controlling or can have a CTN transition. A non-robust test has more relaxed constraints, that is, the side inputs (off-path) can either be maintained at stable non-controlling values or can make a CTN transition for both NTC and CTN transitions at the on-path node. Finally, the functional sensitization test vectors relaxes the constraints even more by allowing the side input to take any arbitrary value if the on-path node has a NTC transition. For a CTN transition at the on-path node, the side input should either be stable at the non-controlling value or can make a CTN transition. Figure 2.2 shows the different types of path sensitization based



(a) Venn Diagram of Test vector types

	On-Path is CTN	On-path is NTC
Strongly Robust	NCV	NCV
Robust	NCV/CTN	NCV
Non-Robust	NCV/CTN	NCV/CTN
Functional	NCV/CTN	NCV/CTN/NTC

CTN: Controlling to Non-Controlling transition
NTC: Non-Controlling to Controlling transition
NCV: Stable at Non-Controlling Value

(b) Off-path signal values for various test types

Figure 2.2: Test vector types and MIS possibilities

classifications for delay test vectors. Typically, for any given path in a fault list, an ATPG strategy first tries to find a *strongly robust* vector and only if it fails, it looks for a vector with *Robust* sensitization criteria. Finally, *Non-Robust* and *Functional* tests are used only if no robust test is found for the path.

The MIS effect introduces dynamic delay change on the path, based on the signal arrival time at the side inputs of the on-path gates. The type of test being generated i.e. *Robust* or *Non-Robust* will determine the amount of MIS effect seen on the path and hence impact the path delay. For delay testing, since the objective is to check if the circuit can meet timing in all possible cases, test generation algorithms need to identify test vectors that maximize the delay on any given test path. However, since the path selection is also dependent on the delays of the paths, it is important to model the MIS effect during path selection itself. Path selection for delay testing requires comparing paths based on their criticality and hence pessimistic delay estimates as used in STA tools could result in selecting

the wrong paths. Path criticality analysis requires estimating the path delay distribution which in turn is determined by the dynamic and process parameter variability seen by the path. Once paths are selected for test, the test vector generated to sensitize the path can be targeted towards maximizing the dynamic delay effect on the path with appropriate logic constraints. The following sections develop a simple and scalable model of the MIS effect for basic logic gates that can be used for critical path selection and optimal test generation for delay test.

2.2 MIS Modeling for NAND2 Gate

When both inputs of 2-input NAND gate are switching, the output signal Arrival Time (AT) will be determined by the dominant input. Again, as mentioned in the previous section, we consider only the cases where both inputs are switching in the same direction. We analyze the MIS effect on the timing arc from input A to output Z of a 2-input NAND gate, which can be of thought as the on-path edge. The *Arrival Time* (AT) of input A is given by T_A, and of input B is T_B. To find the effect of a signal transition at input B on the signal propagation delay from input A to output Z, T_A is set to 0, while T_B is varied. Let the output signal AT for the SIS case, i.e., when the signal is only propagated from A to Z while B is at a stable non-controlling value, be given by $T_Z^{S_A}$. Also, let T_Z^M denote the signal AT at Z when both the inputs are switching, i.e., the MIS case. We define the propagation delay from input A to output Z when only A is switching as $\Delta T_Z^{S_A}$. The impact of MIS on the propagation delay from A to Z is given by $\Delta T_Z^{M_A} = (T_Z^M - T_Z^{S_A})$. In the following two subsections, analytical models for two cases are derived, one for NTC transitions at both inputs and the other for CTN transitions at both inputs.

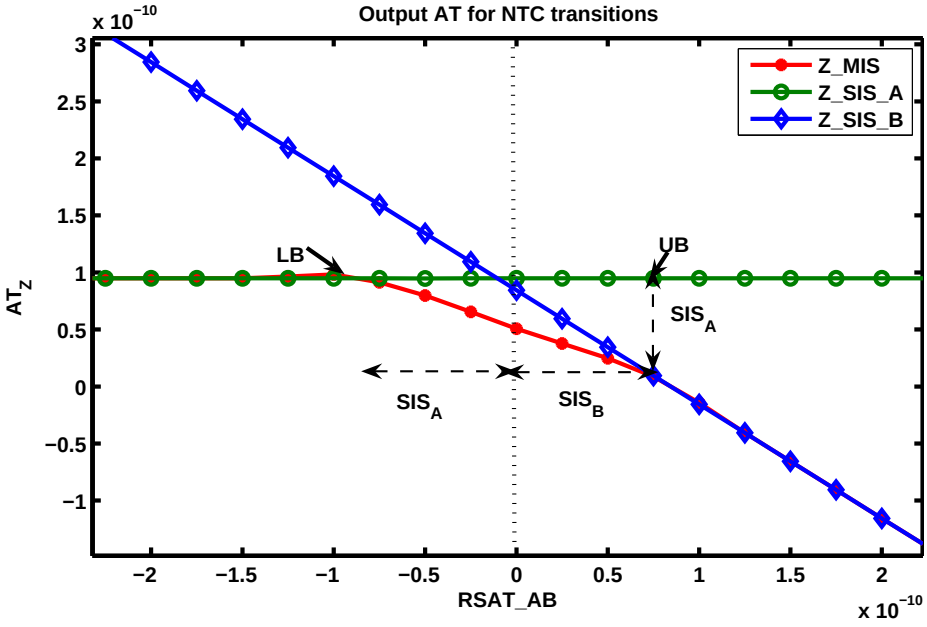


Figure 2.3: MIS effect on A for NTC transition

2.2.1 MIS for NTC

For a 2-input NAND gate, NTC transitions are falling transitions, so the pull-down network is being disabled and the pull-up network is being enabled. Since the pull-up network has parallel signal paths, the earliest arriving input becomes the dominant one in determining the output signal AT. Figure 2.3 shows the output AT as a function of input AT for NTC transitions at the inputs in three different cases. There are three curves; $T_Z^{S_A}$ and $T_Z^{S_B}$ correspond to the SIS cases for inputs A and B, respectively. The T_Z^M curve is for the case when both the inputs have NTC transitions. For all three cases, input A is assumed to arrive at time $t = 0$ while the AT of input B is varied. It can be seen that the SIS curves are asymptotes for the MIS curve and that the MIS effect occurs only when the Relative Signal Arrival Time (RSAT) between A and B is within a certain range. Let the AT difference between inputs A and B be given by $RSAT_{AB} = T_A - T_B$. In the plot shown in Figure

2.3, points LB and UB show the bounds on $RSAT_{AB}$ for which MIS effect is present. Rhe output arrival time is completely controlled by A when $RSAT_{AB} < LB$ and is completely controlled by B for $RSAT_{AB} >= UB$. For $LB <= RSAT_{AB} <= UB$ the delay of A to Z is affected by signal B and hence the MIS error needs to be considered. From the graph, it is clear that for NTC transitions, the MIS phenomenon reduces the effective pin-to-pin delay. It can also be seen that the gate delay in the MIS case is almost 50% smaller than the SIS case. Thus using only SIS delay values during timing verification will give optimistic estimates for min-delay and can fail to detect potential race conditions.

From the figure it can be seen that for the region of interest, the MIS error varies linearly with $RSAT_{AB}$. Thus, the MIS effect on the delay from input A to output Z can be modeled as

$$\Delta T_Z^{M_A} = \begin{cases} 0 & RSAT_{AB} < -LB, \\ \frac{-\Delta T_Z^{S_A}}{\Delta T_Z^{S_B} + \Delta T_Z^{S_A}} \left(RSAT_{AB} + \Delta T_Z^{S_A} \right) + c, & -LB <= RSAT_{AB} <= UB, \\ -RSAT_{AB} + \Delta T_Z^{S_B} - \Delta T_Z^{S_A}, & RSAT_{AB} > UB. \end{cases} \tag{2.1}$$

where $LB = \Delta T_Z^{S_A}$, $UB = \Delta T_Z^{S_B}$ and c is a fitting parameter that can be determined using simulations. In general, the timing libraries used in traditional STA tools have the pin-to-pin SIS delays of any gate characterized at different load-slope points. Since the model in Equation 2.1 only requires knowledge of the SIS delay values, the dependency on load and slope is effectively captured. The MIS delay effect on input A measured using SPICE simulation is compared with that estimated using the above model in Figure 2.4. It was observed that the best fitting c has a weak relation with the load and slope values, but to minimize characterization effort, a single value can be used without significant accuracy degradation. To show that the model is robust over a range of input slopes and output loads, a Monte Carlo simulation was done where the output load and the signals slopes were

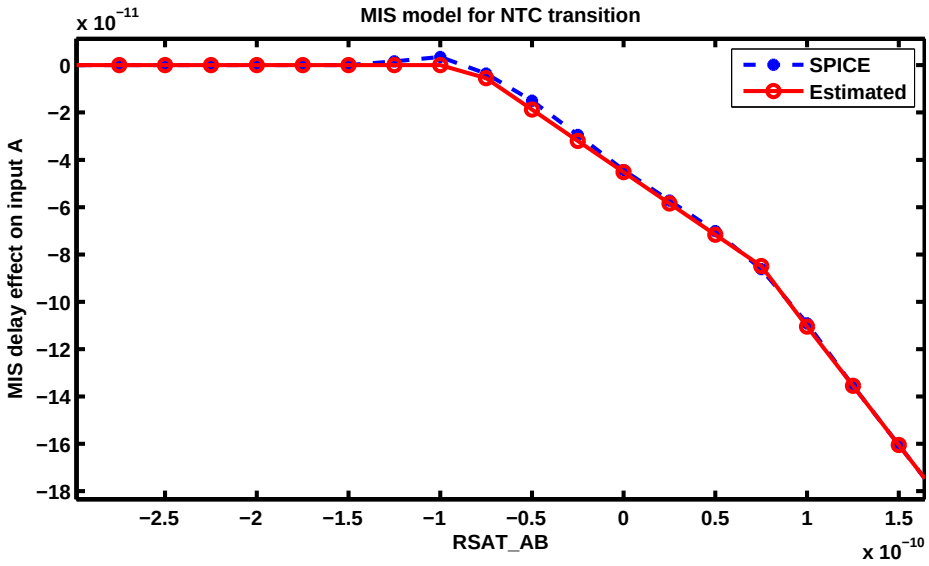


Figure 2.4: Model for $\Delta T_Z^{M_A}$, NTC transition

drawn from a random sample. Input A is assumed to arrive at time $t = 0$ and the AT of input B has a normal distribution with mean 0 and $\sigma = 33ps$. The same value of fitting constant c was used for each point. For each data point, the output AT is estimated using the MIS model presented above, and the scatter plot of the estimated values compared to those measured from SPICE is shown in Figure 2.5. The maximum estimation error was $8ps$ which was 17% of the SIS propagation delay. If the output AT is estimated using just the SIS delays of the earlier arriving signals without accounting for the MIS effect, then the error in estimating the output AT was found to be 61%. Thus accounting for MIS reduces delay estimation error by 44%.

2.2.2 MIS for CTN

A similar analysis was done for CTN transitions at the inputs and graph of the output signal arrival as a function of side input (B) arrival time is shown in Figure 2.6. In a NAND

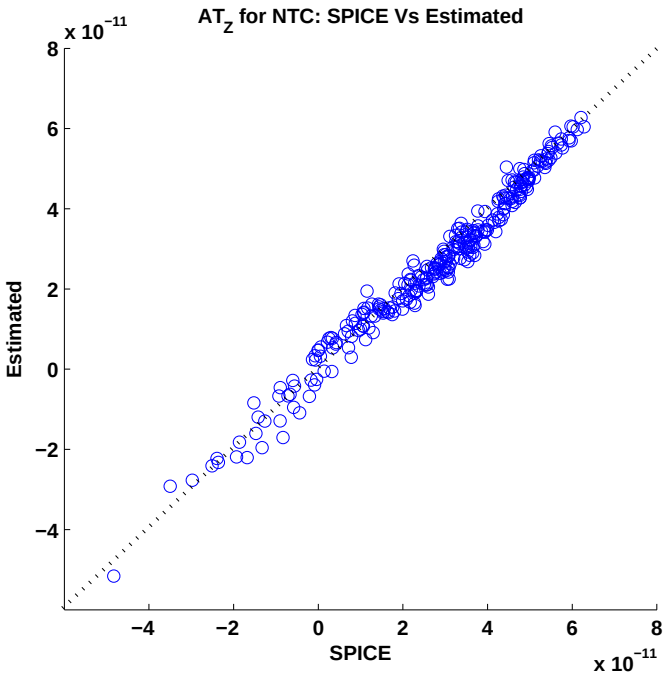


Figure 2.5: Estimate of AT_Z, variable load,slope, RSAT (NTC)

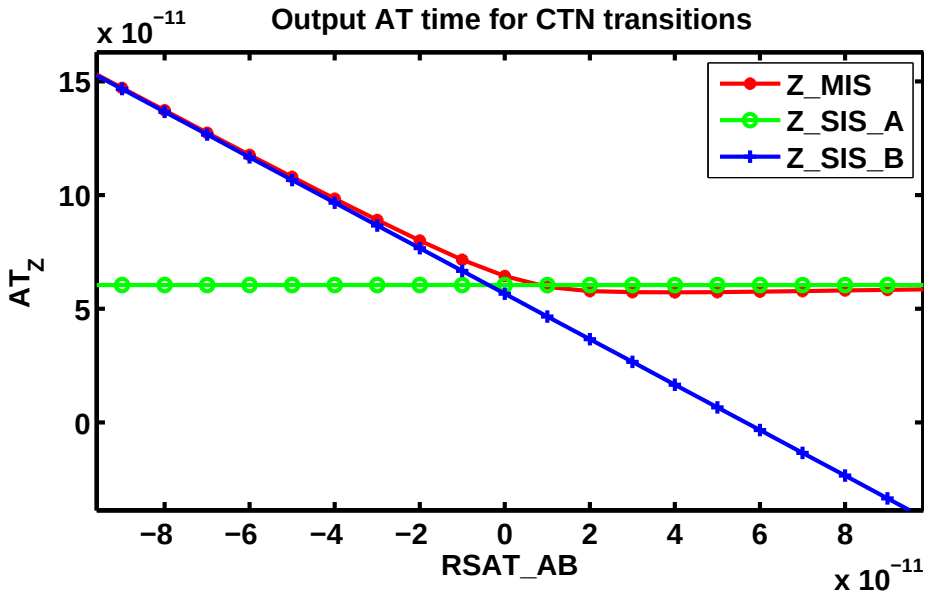


Figure 2.6: MIS effect on A for CTN transition

gate, for CTN transitions at both inputs, the pull-down network is getting activated. Thus the transition is being propagated through the series network and hence the latest arriving signal determines the output arrival time. The MIS analysis for CTN transitions requires more characterization effort since the position of the input in the series stack also matters. In general, for the input closest to the output, the SIS delay is smaller than the MIS delay since all the intermediate nodes will be already discharged if all the other transistors in the series path are ON. For an input away from the output node, the SIS delay is larger than the MIS case, since it will have to discharge all the intermediate node capacitors which have been charged up. Figure 2.6 shows the same three curves for output AT, namely, MIS case, SIS at input A and SIS at input B. It can be seen that for CTN, the MIS region, i.e., where both A and B together determine the output AT is very small and the MIS curve does not deviate much from the SIS cases. For obtaining a smooth transition between the two asymptotes, we can approximate the MIS delay effect on input A as

$$\Delta T_Z^{M_A} = \frac{1}{k} log(\mathrm{e}^{(k(\Delta T_Z^{S_A} + d_A))} + \mathrm{e}^{(k(\Delta T_Z^{S_B} - RSAT_{AB} + d_B))}) - \Delta T_Z^{S_A} \tag{2.2}$$

where k, d_A and d_B are fitting parameters that can be determined using simulation data. A similar expression was used to define the *soft-max* function in [25].

To validate the accuracy of the model, the estimated delay values are compared with the values measured from a SPICE simulation in Figure. 2.7. It can be seen that the model estimates delay values with good accuracy. The fitting parameters k, d_A and d_B have a weak dependence on input slope and output load, but in general a good empirical value can be selected without loss of much accuracy. We checked the robustness of the model by performing a Monte Carlo simulation in which the input slopes, output loads and $RSAT_{AB}$ are generated randomly. The same values of fitting parameters k, d_A and d_B were used for

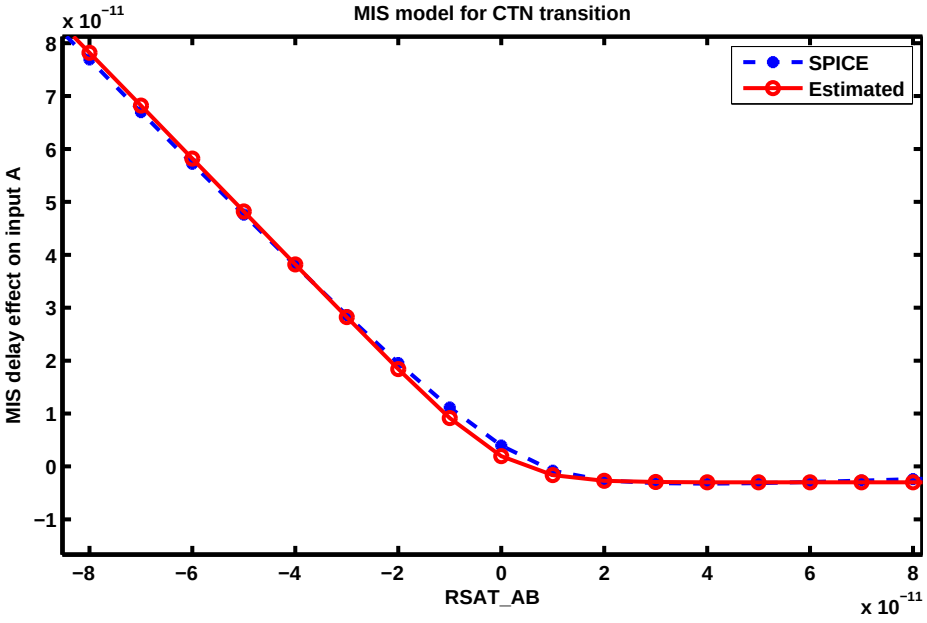


Figure 2.7: Model for $\Delta T_Z^{M_A}$, CTN transition

all the data points and the output AT is estimated using the above model. The scatter plot shown in Figure 2.8 compares the estimated values with those from SPICE measurements and it can be seen that the model is robust for a range of load and slope values. It can be seen that for CTN transitions, the SIS delay can either be an overestimate (negative MIS error) or an underestimate (positive MIS error) of gate delay depending on the RSAT values. For inputs closer to the output node, the MIS effect will tend to increase the delay, while for inputs away from the output, the MIS effect will decrease the delay. Hence the MIS error could be either positive or negative, which is different from the NTC case where the MIS error is always negative. It was also observed that the MIS error for inputs farther from the output node tends to be very small, and the MIS error is significant only for inputs closer to the output nodes. This shows that modeling the MIS behavior for CTN transitions requires more characterization effort, since the timing library needs to record the position of

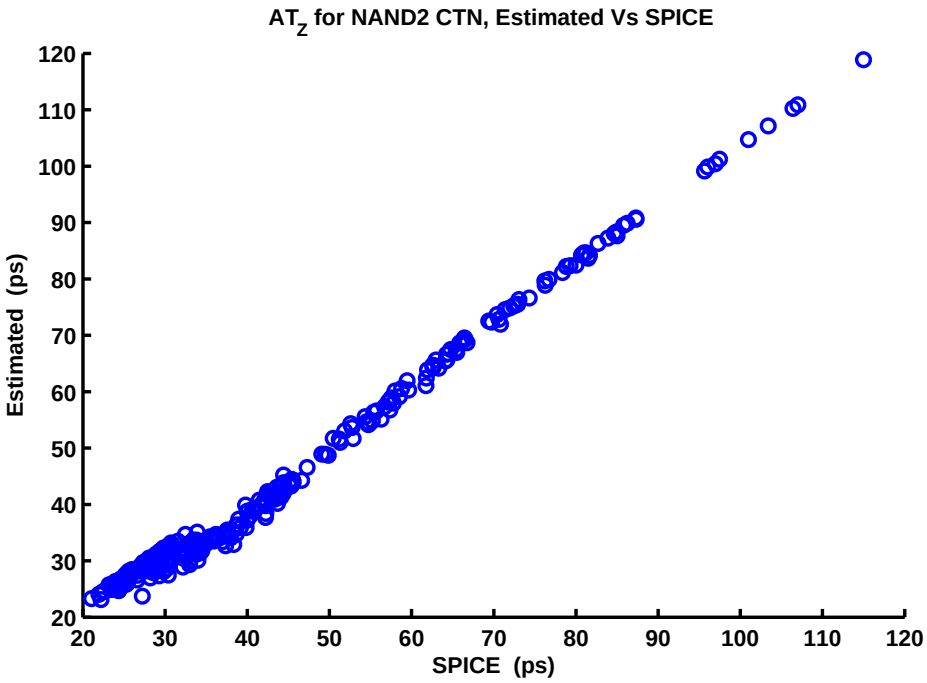


Figure 2.8: Estimate of AT_Z variable load, slope, RSAT (CTN)

an input in the series stack. From our simulations, we observed that for CTN transitions, the percentage error in delay estimate if we use SIS delay only, is much lower (around 10%) for a 2-input NAND gate. This might not be very significant considering the characterization effort required. For larger series stacks, the MIS effect for CTN transitions was also observed to be around 10%. Since characterization for CTN transitions requires more effort due to the importance of input position in the series stack as well, We believe that for CTN case, to reduce characterization effort, it could be more advantageous to compute output AT using the AT and SIS delay of the latest arriving input signal.

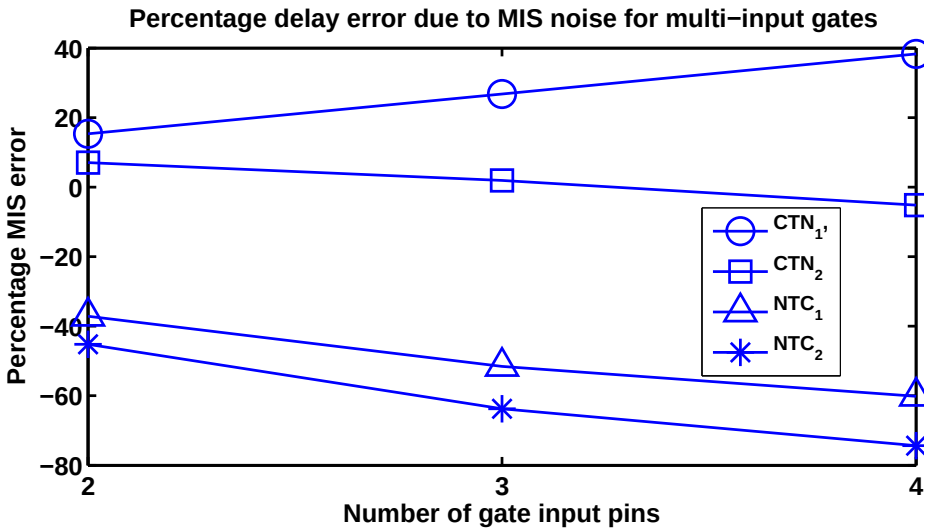


Figure 2.9: MIS effect for multi-input gates

2.3 MIS for Multi-input Gates

As the number of inputs pins of a gate increase, the MIS error also increases. Figure 2.9 shows the maximum percentage error in propagation delay caused due to MIS noise for both CTN and NTC transitions, as the number of gate inputs is increased. For each point, the MIS delay was measured for $RSAT = 0$, i.e., when all the inputs arrive at the same time. Thus the values plotted represent the maximum percentage MIS error for each case. For CTN transitions, the number of transistors in a series stack increases with the number of input pins. In the figure, CTN_1 shows the MIS error for the input that is closest to the output node, while CTN_2 is for the input farthest from the output node. Two similar curves NTC_1 and NTC_2 are shown for NTC transitions at the inputs. It can be seen that for CTN transitions at the inputs, the MIS error is significant only when the input pin is closer to the output node and the error can be as high as 40%. For the input pin away from the output node, the MIS errors are negligible ($< 10\%$). For NTC transitions however, since multiple

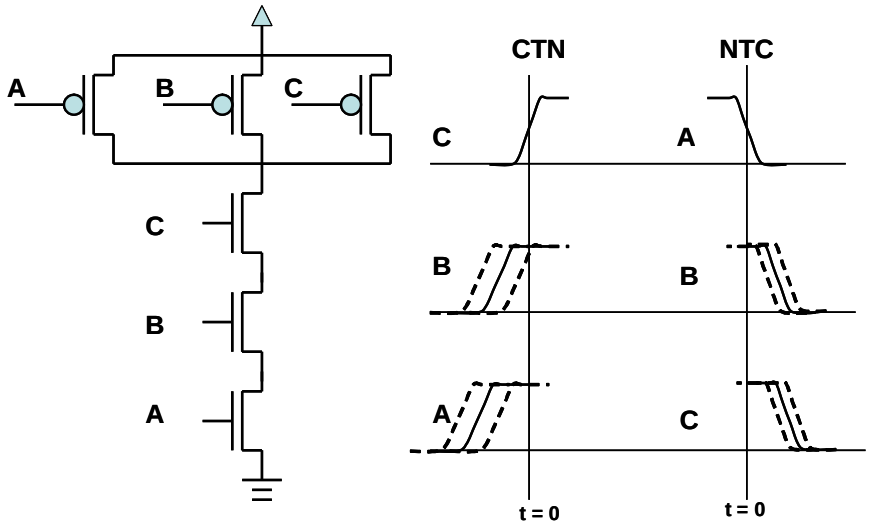


Figure 2.10: NAND3 gate

input signals mean multiple parallel current paths, the percentage delay errors due to MIS are very high starting from -45% for a 2-input gate to up to 75% for a 4-input gate. In this section, a model for the MIS error for NTC transitions is derived for a 3-input gate and the model can be easily extended to gates with more pins.

2.3.1 NTC Transitions for NAND3

For NTC transitions at all inputs on a gate, the output arrival time is determined by the earliest arriving signal. Consider a 3-input NAND gate as shown in Figure 2.10, with NTC transitions on all the three inputs and input A arriving the earliest at time $t = 0$. The later arriving signals B and C, will affect the propagation delay of the gate due to the MIS effect. Let us assume that the sequence of input arrival is A, B, C, i.e., signal A arrives the earliest followed by B and then C. Only the positive range of $RSAT_{BA}$ and $RSAT_{CA}$ therefore needs to be considered. The output AT is controlled by the dominant (earliest for NTC) signal which is input A. From the previous section on 2-input gates, we know that

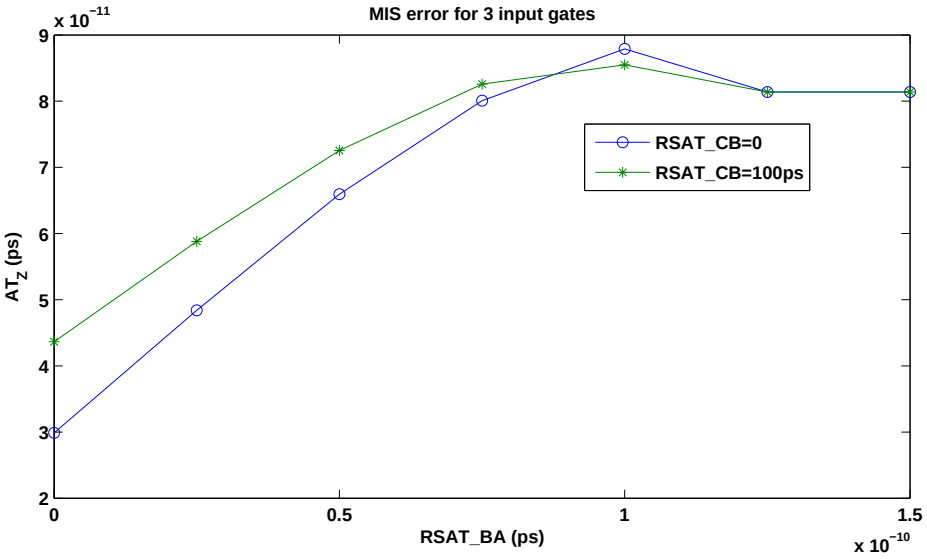


Figure 2.11: MIS error for 3-input NAND gate

signal B will affect output AT only when it arrives within a certain time window, and the same is true for signal C. If input C arrives much later than B, then only signal B influences the output AT due to MIS effect. However as C arrives close to B, both B and C together affect the output AT. The output AT with respect to $RSAT_{BA}$ is shown in Figure 2.11 for two cases. The case where $RSAT_{CB} = 100ps$ represents a simple 2-input NAND model, since input C arrives much later and hence does not affect the output arrival time. The second case is for $RSAT_{CB} = 0ps$, where both B and C affect the output arrival time due to MIS. From the figure, it can be seen that the third input affects the slope of the MIS curve. Notice that the MIS curves are similar to those presented in the previous section for a 2-input NAND gate, except that only the positive region of RSAT is considered here. Let s be the rate at which the output AT drops due to MIS (i.e., the slope), and the MIS error at $RSAT_{BA} = 0$ be given by m (i.e., the y-intercept). To estimate the MIS effect of input C, we need to determine how m varies with $RSAT_{CB}$. Once m is known, the slope of the

new curve can be easily determined. We define a generic function $F(x, xp, yp, s)$ as follows:

$$F(.) = \begin{cases} s(x - xp) + yp + \epsilon, & \text{for } 0 <= x <= xp \\ yp, & \text{for } x > xp \end{cases}$$

where ϵ is an empirical fitting constant. Thus $F(.)$ remains constant at yp for $x > xp$ and drops at the rate of s for $x <= xp$. This generic function can be used to describe the MIS curve, i.e. the output AT remains constant at $\Delta T_Z^{S_A}$ while $RSAT_{BA} > \Delta T_Z^{S_A}$ and for $RSAT_{BA} < \Delta T_Z^{S_A}$, the value drops at the rate of $\Delta T_Z^{S_A}/(\Delta T_Z^{S_A} + \Delta T_Z^{S_B})$. It was observed that the variation in m with respect to $RSAT_{CA}$ also follows a behaviour similar to $F(.)$. The value of m remains constant while $RSAT_{CA} > \Delta T_Z^{S_C}$ and for $RSAT_{CA} < \Delta T_Z^{S_C}$, the value drops at the rate of $0.5 \times \Delta T_Z^{S_A}/(\Delta T_Z^{S_A} + \Delta T_Z^{S_C})$. This shows that the input B which is closest to A has the most impact on the output AT, while input C which arrives after B has a second order impact, in that, it changes the rate at which output AT varies with $RSAT_{BA}$. The same trend can be continued for higher input gates, while taking into consideration the trade off between accuracy and effort. The complete method for estimating the MIS delay error for NTC transitions in a 3-input gate is shown below

$$T_Z^{M'} = F(RSAT_{BA}, \Delta T_Z^{S_A}, \Delta T_Z^{S_A}, \frac{\Delta T_Z^{S_A}}{(\Delta T_Z^{S_A} + \Delta T_Z^{S_B})}) \tag{2.3}$$

$$m' = T_Z^{M'}(RSAT_{BA} = 0) \tag{2.4}$$

$$m = F(RSAT_{CA}, \Delta T_Z^{S_C}, m', \frac{\Delta T_Z^{S_A}}{2 \times (\Delta T_Z^{S_A} + \Delta T_Z^{S_C})}) \tag{2.5}$$

$$s = (\Delta T_Z^{S_A} - m')/\Delta T_Z^{S_A} \tag{2.6}$$

$$\Delta T_Z^{M} = F(RSAT_{BA}, \Delta T_Z^{S_A}, \Delta T_Z^{S_A}, s) \tag{2.7}$$

where, $T_Z^{M'}$ and m' are intermediate variables. The output AT estimated using the above method is shown in comparison with values measured from SPICE simulations in Figure 2.12. Three curves are shown for three different values of $RSAT_{CA}$ and it can be seen that

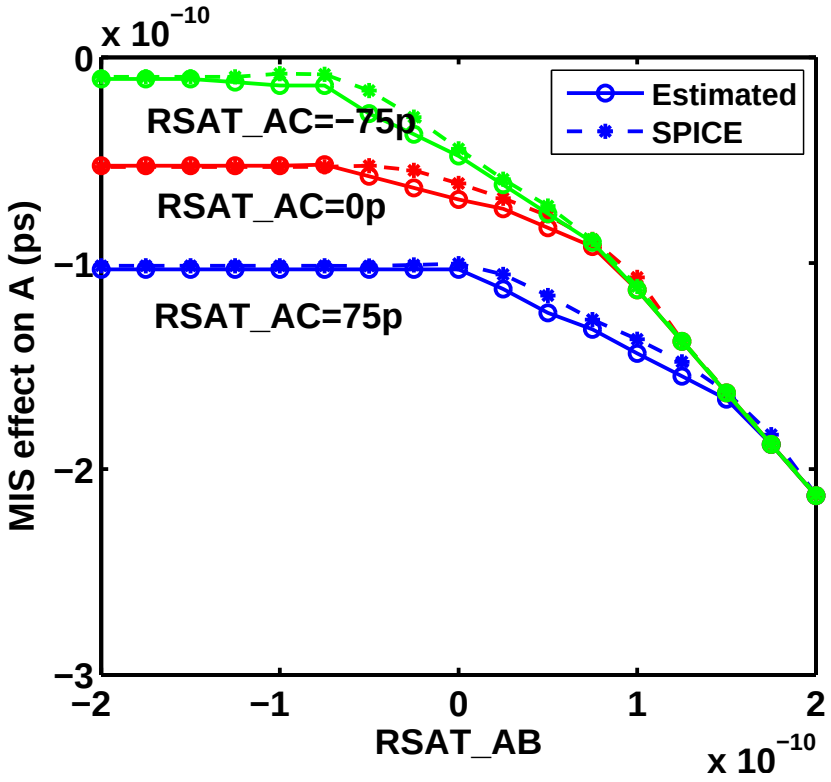


Figure 2.12: MIS model for 3-input NAND gate, NTC

the model estimates the MIS errors with reasonable accuracy. The main advantage of this model is that it does not require any extra library characterization. The MIS errors can be determined using the SIS delay values which can be obtained from traditional timing libraries. The only empirical constant here is ϵ which has a weak dependency on input slope and output load, but a single value can be used without much loss of accuracy. This can be seen from the plot shown in Figure 2.13, which shows the delay distribution of the timing arc from input A to output Z for NTC transitions at all three inputs. Monte Carlo simulations were done with input A arriving at $t = 0$, input signal slopes, output load are drawn from a Gaussian distribution. Arrival times of signal B and C are also assumed to be Gaussian ($AT_B = N(30ps, 16ps), AT_C = N(50ps, 16ps)$). The SIS delay represents the distribution

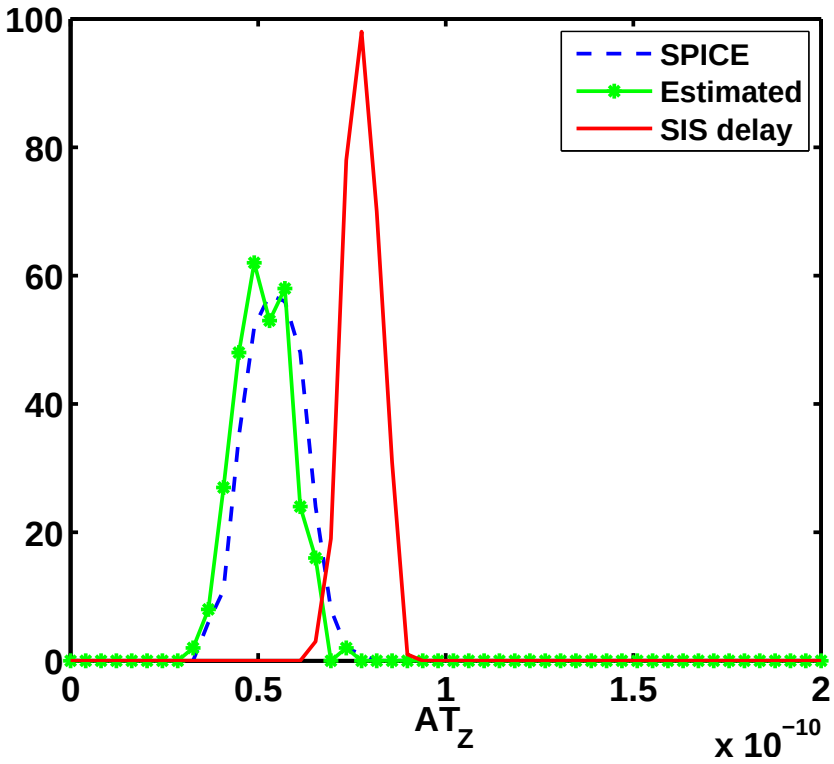


Figure 2.13: NAND3 delay distribution, NTC

estimated when MIS effect is ignored. The maximum error in estimating the output AT using the above model was 15%, while the error in delay estimation is 51% if only SIS delays are considered. Thus accounting for the MIS effect reduces error by 35%.

2.3.2 CTN Transitions for NAND3

As mentioned earlier, the MIS error for CTN transitions is negligible for inputs away from the output node and becomes significant only for inputs closer to the output node. For the following discussion, we therefore only consider the input pin closest to the output node, which in this case is input C. Let the sequence of input arrivals be A, B, C, i.e., C arrives the latest, and A and B affect the output AT due to the MIS effect. From the model for 2-input

gates in the previous section, we know that for CTN transitions, the MIS error increases exponentially as the off-path input arrives closer. For 3-inputs, both B and A will introduce MIS noise and hence affect the output AT. It was found that for CTN transitions, the total MIS error can be estimated by taking a scaled summation of the individual MIS errors as follows:

$$\Delta T_Z^{M_C} = \Delta T_Z^{M_{CB}} + 0.5 \times \Delta T_Z^{M_{CA}} \tag{2.8}$$

where $\Delta T_Z^{M_{CB}}$ and $\Delta T_Z^{M_{CA}}$ are the MIS effect on input C due to inputs B and A, respectively, and can by computed from Equation 2.2. The output AT of a NAND3 gate estimated using the above model is compared with that measured from SPICE simulation in Figure 2.14. The graph shows the MIS error with respect to $RSAT_{BA}$ for three different values of $RSAT_{CA}$. The estimated values follow the SPICE curves with reasonable accuracy. Figure 2.15 shows the results of Monte Carlo simulation for estimating the delay distribution of the timing arc from input C to output Z. The input slopes and output loads were obtained from a gaussian distribution, similarly the ATs of signals A and B were also treated as random variables, with $AT_A = N(-20ps, 7ps)$, and $AT_B = N(-30ps, 10ps)$. Signal C is assumed to arrive at $t = 0$. The maximum error in estimation using the above model is 16%, while the maximum error if only SIS delays are used was found to be 57%. Thus modeling for MIS when estimating the delay distribution of a gate reduces timing estimation errors significantly.

Thus in general, the total MIS error for multi-input gates with NTC transitions at the inputs can be obtained by iteratively computing the effect of farthest signal (signal with maximum RSAT) on the model parameters and the highest error is introduced by the closest arriving signal. For CTN transitions, the total MIS error can be obtained by taking a scaled sum of the individual MIS errors introduced by each input. For both cases, it is important to first determine the order in which the signals arrive. Traditionally, the most formidable

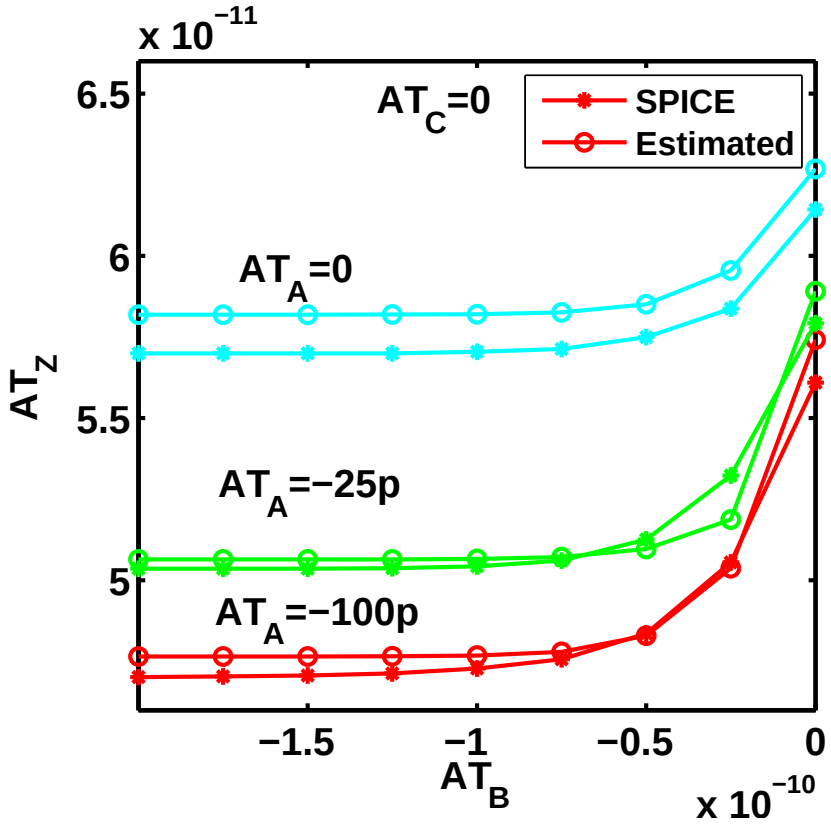


Figure 2.14: MIS model for 3-input NAND gate, CTN

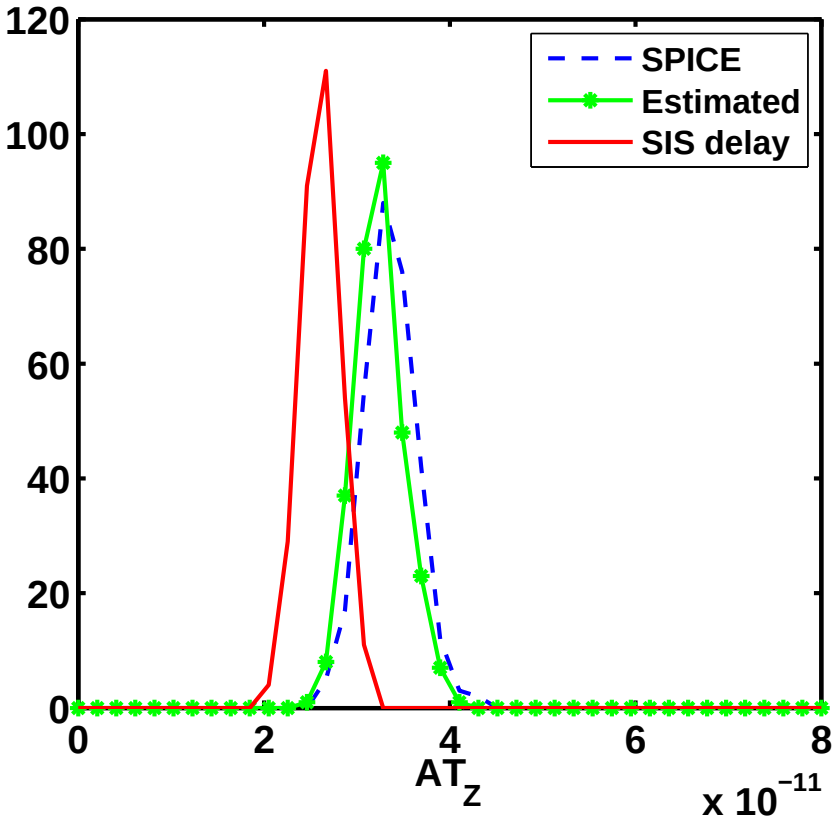


Figure 2.15: NAND3 delay distribution, CTN

challenge in incorporating MIS effect during timing analysis is the amount of characterization required. If a separate model is used, then the model constants need to computed at each load-slope point in the timing library. The analytical models derived above are obtained from values already known from the timing library, namely the SIS delays. Since SIS delays vary with output load and input slope, the model captures this dependency effectively. There are very few fitting constants which have to be characterized only once for each gate and were found to be robust over a large range of input slopes and output loads. The above model can be easily incorporated in statistical timing analysis, i.e., the output AT can be determined more accurately given the AT distributions of the side inputs.

2.4 Delay Variability Due to MIS

Most path selection techniques developed in the past do not focus on the path delay estimation problem. Path delay is simply computed by adding the SIS delays of each timing arc on the path, thus assuming that the off-path signals are stable at non-controlling value. This underlying assumption implies that only the set of strongly robust vectors is being considered, which is generally a very small subset of all possible vectors that can sensitize the path. The signal AT at the off-path inputs will vary from vector to vector, and hence in each clock period, the $RSAT$ between on-path and off-path signals will be different. Since the test vector is not known at the path selection phase, the off-path input arrival is not known. Thus $RSAT$ can be treated as a random variable and hence the path delay change induced due to MIS noise (which is a function of RSAT) will vary dynamically. Depending on the amount of MIS noise each path experiences, the delay distribution and hence path criticality will be different. Accounting for this dynamic variability is extremely important when path based analysis is done for critical path selection. For instance even if two paths have similar delays for the SIS case, if one of the paths has late arriving off-path signals (positive RSAT) for CTN transitions, its average path delay will be much higher, and if it has early arriving off-path signals (negative RSAT) for NTC transitions, its average path delay will be much lower. Note that taking into account off-path signal signal ATs is also important for detecting timingly false paths, i.e., the path for which the off-path signal is being propagated.

Figure 2.16 shows the histogram of path delay due to variation in MIS noise. The values were obtained by performing Monte Carlo simulations on a 10-stage NAND chain. The dotted line in the center is the path delay when all off-path inputs are stable and non-controlling. In the other two cases, the off-path signal ATs were sampled from random

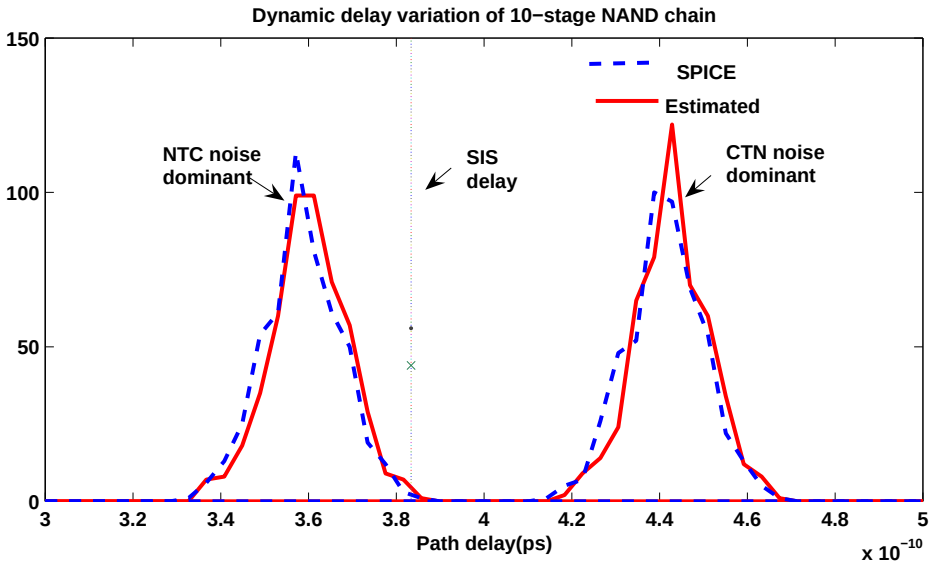


Figure 2.16: Dynamic variation on NAND chain path

distributions, and each off-path an on-path signal pair switches in the same direction. As mentioned earlier, if for stages with CTN transitions the off-path signal arrives later, then average path delay increases (CTN Noise dominant), and if the off-path signal arrives much earlier for stages with NTC transitions, then the average path delay decreases (NTC Noise dominant). In both cases, we performed a Monte-Carlo simulation using our proposed model and the estimated path delays can be seen to match those measured from SPICE very well. For our simulations the RSAT distributions between the on-path and off-path signals were assumed to be known. However estimating the RSAT distributions between any two nodes of a circuit is a non-trivial problem.

2.5 Conclusion

Path delay is subject to variability due to both process parameters and dynamic factors. During critical path selection for delay testing, the delay variability of a path needs

to be taken into account when estimating path criticality. This research was targeted towards modeling the effect of *Mmultiple Input Switching* on the path gates, which is an important factor in the total dynamic delay variability of the path. The analysis done in this research shows that computing path delays by simply adding SIS delay values can lead to inaccurate delay estimates. The delay of a path is affected by the signal arrival times at the side inputs of the on-path nodes. Signals can arrive at the side inputs through many different paths, depending on the vector used to sensitize the target path. Thus the path delay varies dynamically as a function of the Relative Signal Arrival Times (RSAT) between on-path and off-path nodes. The objective of this research was to develop a simple model that can be used to estimate the dynamic variation in path delay as a function of RSAT between different input signals. We have derived an analytical model that is based on already known data from the timing library, i.e., the SIS delays. Minimum characterization is required for the fitting constants and the model was found to be robust over a range of load and slope points. The model can be used for estimating the dynamic variability in path delay due to MIS during critical path selection.

Test generation for delay testing requires selecting vectors that maximize the delay of the selected paths. From the MIS delay models described above, it is clear that the delay of any edge on a path is reduced if both on-path and off-path signals have NTC transitions. This combination is only possible for functional sensitization. For both non-robust and functional sensitization tests, if an on-path node has a NTC transition, then the side input can make a CTN transition. This will create a glitch at the output which is undesirable since it requires more control on the capture signal to prevent latching an incorrect value. Also, in this case, if the side input arrives later than expected, then the output node will not make any transition and the path cannot be tested. Thus considering both path delay and

robustness criteria, *Non-Robust* and *Functional Sensitization* tests are the least desirable. Based on our MIS model for CTN transitions, the path delays for *Robust* vectors will tend to be higher than that for *Strongly Robust* vectors. This is because *Robust* vectors allow CTN transitions on the side inputs when on-path nodes have CTN transitions, while *Strongly Robust* test requires the side inputs to be stable in all conditions. Thus to maximize delays of the path being tested, *Robust* tests should be more preferable than *Strongly Robust* tests.

Chapter 3

Path Selection Considering Coupling Noise

3.1 Introduction and Motivation

Coupling capacitance is the parasitic capacitance present between two neighboring interconnect lines as shown in Figure 3.1. With aggressive scaling, while device dimensions are shrinking steadily, interconnect cross-section dimensions are not reducing at the same rate. At the same time, with high amount of functional integration on a single chip, interconnects are becoming longer. In addition, the aspect ratio of wires is increasing, making the wires taller, while the spacing between the wires is reducing, thus increasing the coupling capacitances. With longer interconnects, larger wire aspect ratios and smaller wire spacing, coupling capacitance is becoming the dominant component of the total wire capacitance. It has been reported that coupling capacitance can be as high as 80% of the total wire capacitance [66]. The wire (node) which is switching is the aggressor and the one at which noise pulse is injected is called the victim. This coupling noise, also known as crosstalk, can cause signal integrity issues which can lead to transient functional failures. For instance, if the noise pulse is large enough, it can potentially cause switching in the following stages, and an incorrect value could be captured in a downstream storage element. Developing tests for detecting crosstalk induced transient logic failures is a well researched problem [20]-[35].

Coupling induced noise or crosstalk can occur when the victim node is quiet and only the aggressor is switching. However, when the victim signal is also switching, then the

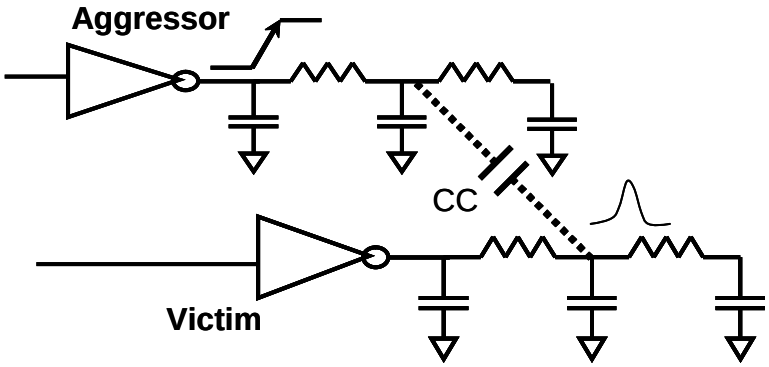


Figure 3.1: Victim-aggressor coupling

parasitic coupling capacitance can change the signal propagation delay at the victim node. The delay change can be explained using the *Miller Coupling* phenomenon which changes the effective capacitance of a capacitor when the voltages at both ends of the capacitor are varying. Depending on the direction in which the victim and aggressor signals are switching, the effective capacitance can increase or decrease. For instance, if both victim and aggressor are switching in the same direction, then the effective capacitance seen at both nodes reduces, since the nodes are aiding each other. Thus the victim propagation delay will be reduced. On the other hand, if victim and aggressor are switching in opposite directions, then the effective capacitance seen at both nodes increases, there by increasing the propagation delay at the victim node.

In Static Timing Analysis (STA), the delay effect of coupling is typically captured by replacing the coupling capacitance between the two nodes with an equivalent capacitance to ground. This requires scaling the physical coupling capacitances by a factor called the *Miller Coupling Factor* (MCF), to obtain the effective capacitance to ground seen at the victim node.

$$CC_{eff} = MCF \times CC_{phy} \tag{3.1}$$

Figure 3.2 shows three different cases of victim and aggressor switching combinations and the corresponding MCF values that are used. If the aggressor is quiet, then there is no effect on the victim and hence MCF is 1. If the aggressor is switching in the same direction as the victim, then the MCF is less than unity, thus reducing the effective capacitance seen at the victim node. This case needs to be considered when analyzing min-delay failures. For

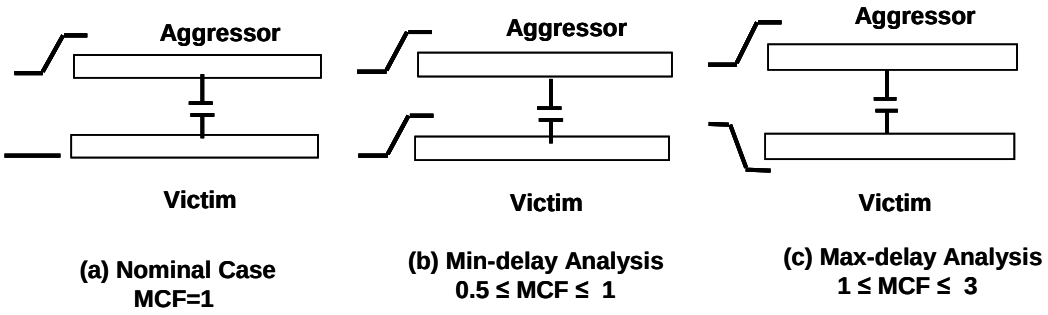


Figure 3.2: MCF values for different cases

max-delay analysis, we only consider the case when the victim and aggressor are switching in opposite directions. It has been shown that the worst case MCF can be as large as 3 [19] but $MCF = 2$ is generally used as the worst case to avoid excessive pessimism. In reality, not all nodes which have parasitic capacitances will affect circuit delays. A signal transition at the aggressor node can affect the victim signal propagation delay only if the victim and aggressor switching events have temporal proximity. The amount of delay change is a function of the Relative Signal Arrival Times (RSAT) between the aggressor and victim nodes and also the slew rates of the two signals. The delay change is maximum when the difference in the Arrival Times (ATs) of the aggressor and victim is minimum (RSAT = 0). Thus the actual MCF could be anywhere in the range of 1 to 3. The objective of static timing verification is to estimate the upper bound of signal propagation delay along any path in the circuit and therefore, STA tools use worst case estimates of delays. Thus if the

signal switching windows of the victim and aggressor nodes overlap, then a pessimistic value of MCF is used to scale the coupling capacitance.

Since coupling noise incurs a large impact on circuit timing, it is important to model the effect of coupling noise during delay test generation. In [48], a crosstalk induced delay fault is modeled by treating each aggressor path as an individual fault. This however, leads to a very large fault list. The initial set of critical paths is assumed to be known and fault coverage is then reported as the number of target faults for which test vectors are found. In [44], a genetic algorithm based search technique was used to find a test vector that maximizes coupling noise, but coupling noise was not considered during the path selection phase. Also, the problem with selecting the noise sources without considering the logical constraints as done in [44] is that it can result in selecting too many noise sources or the wrong ones. Thus, in most of the previous works, path selection and test generation are treated as independent processes and the focus is given primarily on the test generation part, while assuming that the critical paths are known.

In this research, we show that it is important to consider coupling induced delays during the path selection phase itself, as it will affect the set of paths selected for test generation. If coupling induced delay change is not modeled during path selection, path delays will be under-estimated (optimistic estimate), which can result in missing some real critical paths from the test set. On the other hand, even though static timing analysis tools model the delay impact of coupling noise, these tools cannot be used for critical path selection for delay tests due to the pessimism involved. The primary reasons are as follows,

- Most STA tools perform block based analysis while critical path selection require path based analysis. STA does not take into account circuit logic constraints and hence

assume that all aggressors on a path can be active together. In reality only a subset of aggressors can be active at any given time

- The same Miller Coupling Factor (MCF) is used to scale all coupling sites if victim and aggressor timing windows overlap at that site, while in reality the MCF at each coupling node will be different for different victim and aggressor signal paths

Using such pessimistic path delay estimates can result in not only selecting many non-critical paths but also missing some real critical paths due to incorrect path ranking. In [44], the top K aggressors on a path are selected for test generation, based only on performance sensitivity, without considering the logic constraints on the aggressors. This increases the search space to find the optimal test vector and can also result in targeting the wrong vector space. We apply both logic and timing constraints at the various coupling sites on a path to find the subset of aggressors that can be active together and introduces the maximum coupling noise. Our methodology thus reduces the pessimism in path delay estimation and also drastically reduces the search space for optimized test vector generation. The main features of our methodology are as follows.

- The maximum delay at each coupling site is computed by determining the minimum arrival time difference between aggressor and victim signals as described in Section 3.2 .

- For any path, the aggressors are classified as 'critical' or 'relaxable' during the path selection phase as shown in Section 3.3 .

- An efficient technique is used to find the worst aggressor combination by solving a constrained maximization problem.

The results are shown in Section 3.4 followed by the conclusion in Section 3.5.

3.2 Maximum Coupling Noise at a Site

During critical path selection, paths should be compared at their maximum delays considering dynamic variations. For this, we need to estimate the maximum delay change that can be caused at any coupling site. In [2], an analytical model of the delay change at the victim node as a function of the signal arrival times and signal slews was derived, and is referred to as the Delay Change Curve (DCC). The DCC showed that the delay change is maximum when the $RSAT$ between aggressor and victim is minimum. The following subsection describes a graph based algorithm to estimate the minimum $RSAT$ for a given victim-aggressor pair. Given the minimum $RSAT$ estimate, we use a regression based model to determine the corresponding MCF. Section 3.2.2 gives the details of the MCF modeling technique. The estimated MCF can then be used to scale the physical coupling capacitance to obtain the effective load seen at the victim due to coupling noise. This effective load capacitance can then be used to estimate the new delay at the victim node using standard timing libraries.

3.2.1 Estimating Minimum RSAT

The circuit to be processed is represented as a Directed Acyclic Graph, with each pin being a node and pin-to-pin arc being an edge. A *Source* node is added with outgoing edges to all the primary and pseudo-primary inputs and a *Sink* node is added with incoming edges from all primary and pseudo-primary outputs. Our objective is to determine the maximum delay increment on a path considering coupling noise. Thus the path under consideration will be the victim path, and hence the signal arrival time at the victim node can be computed

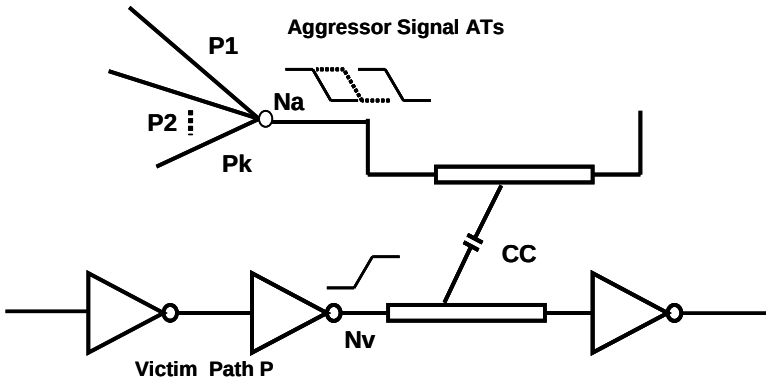


Figure 3.3: Multiple aggressor paths

accurately. On the other hand, the signal arrival time at the aggressor node is not known. As shown in Figure 3.3, the aggressor signal has multiple possible arrival paths. For any edge with an aggressor, the amount of delay increase introduced due to coupling noise depends on the signal Arrival Time (AT) difference between victim and aggressor. The maximum coupling noise occurs when the RSAT between aggressor and victim is minimum. Since the victim arrival time is known, to estimate the minimum RSAT, we need to find an incoming path at the aggressor node, whose delay is closest to the victim AT.

Let the victim arrival time be the *target* delay. We developed a simple graph traversal based algorithm for estimating the minimum $RSAT$. The pseudo code of this algorithm is shown in Figure 3.4. Given a *target* delay at the aggressor node, the objective is to find an incoming signal arrival time that is closest to the *target* delay. Signal AT at any node will depend on the incoming path delays, and it is impractical to trace back all possible incoming paths. The timing window at any node gives the bounds on the incoming path delays at that node and can be easily computed by performing a single block-based traversal of the graph as a part of pre-processing. The timing window at any node n, gives the minimum and maximum signal AT at that node, [minAT(n), maxAT(n)]. Starting with the aggressor

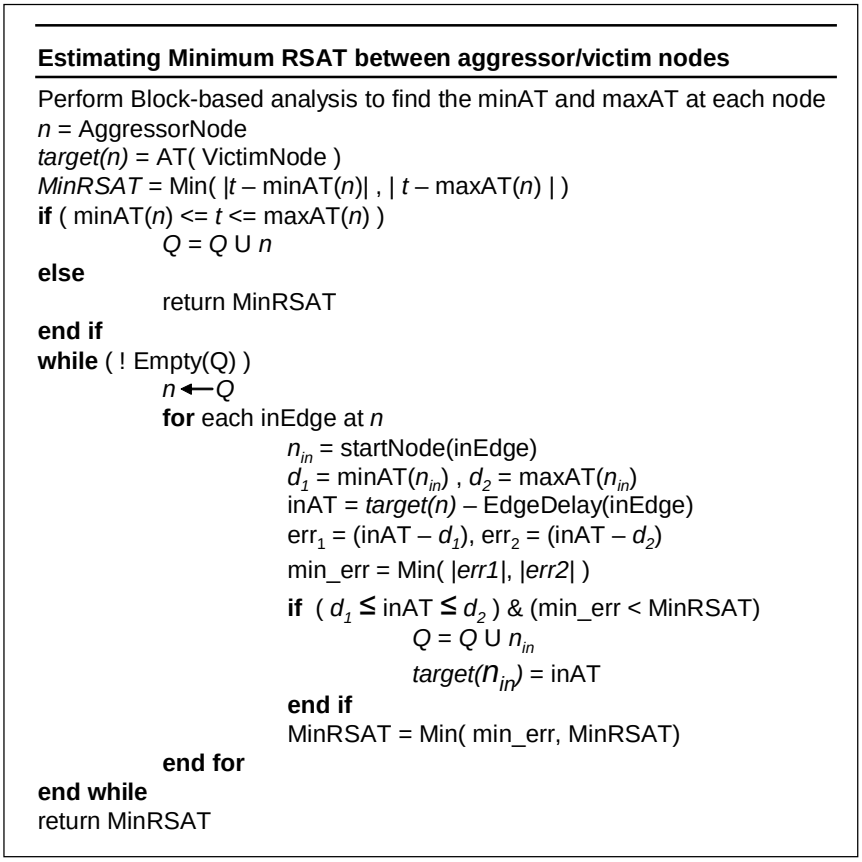


Figure 3.4: Min-RSAT estimation algorithm

node, the graph is traversed towards the *Source* such that the minimum RSAT estimate is refined.

At any node, the next *candidate* nodes to be visited are the incoming nodes. Given a *target* delay at any node n, we compute two error values, $err_1 = (target - minAT(n))$ and $err_2 = (target - maxAT(n))$. The best estimate of the minimum RSAT at this node is then given by $minRSAT(n) = Min(|err_1|, |err_2|)$. Starting from the aggressor node, we trace back along its incoming nodes, and at each node, the estimate of minimum RSAT is updated. The candidate node which gives the smallest $minRSAT$ is the next node to be

visited. We maintain a sorted list of nodes that are to be visited next. A candidate node is ignored if its target delay is outside its timing window, which is the case if both err_1 and err_2 have the same sign. A candidate node is added to the queue, if its $minRSAT$ estimate is less than or equal to the current $minRSAT$ estimate. The algorithm stops when the queue is empty, which means that the $minRSAT$ cannot be reduced any further.

Consider the graph shown in Figure 3.5. The delay of each edge and the timing window at each node are shown. The target at node n_{10} is 13 and the node traversal order to arrive at the $minRSAT$ estimate of 0 is also shown. In this particular case, since all values are integers, the search can also stop if $minRSAT$ is estimated to be 0. Once the minimum $RSAT$ value possible between the victim and aggressor nodes is known, the worst case MCF can be derived using the MCF prediction model as described in the next section.

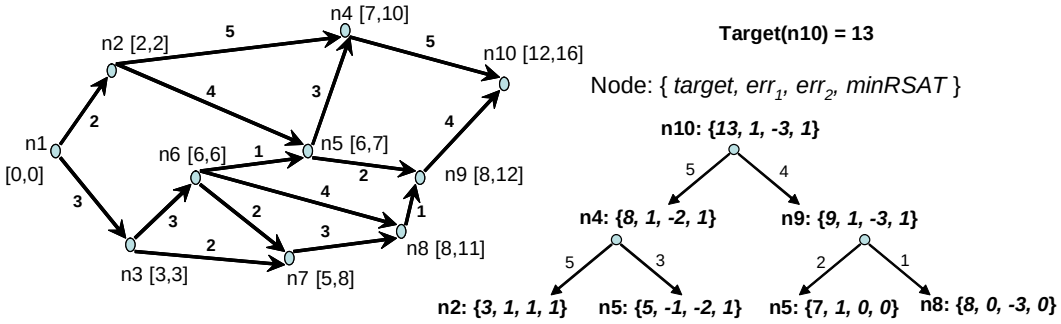


Figure 3.5: Estimating minimum RSAT

3.2.2 Estimating MCF

In [2] a detailed analysis of coupling noise is presented and a Delay Change Curve (DCC) is derived based on the noise pulse injected into the victim when the aggressor switches. The DCC gives the change in delay of the victim as a function of RSAT. The

analytical approximations proposed in [2] involve approximating the noise waveform by a two-piece model consisting of a linear ramp and exponential decay after the peak voltage. Delay parameters are then extracted and the DCC is obtained by curve-fitting to a one-pole model. The limitation of this model is that it uses a two-segment RC model and requires extracting curve fitting parameters to generate the DCC. Thus the delay change estimate is tied to the delay model and wire model being used. In this work, instead of using the delay change curve, we model the MCF directly as a function of RSAT and signal slews. The advantage of modeling MCF as compared to delay, is that it is independent of the delay model used for computing wire delay. In [45], an analytical multi-regional model for MCF variation with respect to RSAT was obtained by curve-fitting results of SPICE waveforms. The authors of [45] use a sensitivity based approach to estimate the delay variance considering RSAT as a Gaussian distribution and show that as compared to the model in [2], their model has better accuracy in predicting wire delay variance when RSAT is a random variable, but the errors are still as high as 30%. Analytical models entail approximations and curve fitting parameters and hence can cause inaccurate estimates.

In this work, we use Multivariate Adaptive Regression Splines (MARS) [27] to construct a non-linear model for MCF as a function of RSAT and victim and aggressor slews. $MARS$ is a non parametric regression algorithm that builds a prediction model for a dependent variable by fitting piece-wise splines (basis functions) in the independent predictor variables (input parameters). The $MARS$ algorithm is widely used for modeling circuit behavior in analog testing. In our case, the dependent variable is the MCF which has a strong dependency on $RSAT$ and the input slews. For obtaining the regression model, SPICE simulations are first run to generate training data points in a bounded parameter space. The input parameters, namely $RSAT$, victim slew (v_{slew}), aggressor slew (a_{slew}) and the

wire load capacitance (C_{load}) are generated randomly and the propagation delay of the arc is measured both when aggressor switching (D_w) and when the aggressor is quite ($D_{w_{nom}}$). The arc delay without any coupling capacitance ($D_{w_{cc=0}}$) is also measured for each point. Let CC_{eff} be the effective coupling capacitance resulting due to aggressor switching and CC be the physical coupling capacitance at any node. Then

$$MCF = \frac{CC_{eff}}{CC} \tag{3.2}$$

The path delay change due to coupling capacitance can be written as:

$$D_w = D_{w_{cc=0}} + \frac{\partial D_w}{\partial CC} \times CC_{eff} \tag{3.3}$$

Similarly, the nominal path delay is given by

$$D_{w_{nom}} = D_{w_{cc=0}} + \frac{\partial D_w}{\partial CC} \times CC \tag{3.4}$$

Thus MCF can be calculated as:

$$MCF = \frac{(D_w - D_{w_{cc=0}})}{(D_{w_{nom}} - D_{w_{cc=0}})} \tag{3.5}$$

Equation 4 is used to measure the MCF when training data needed for the MARS model is generated using SPICE simulations. We develop a MCF prediction model by characterizing a fixed length WL_m wire which is represented with a 4-segment pi-model and three possible locations for coupling capacitors are shown in Figure 3.6. The advantage is that the same model can be used for estimating the delay change of a longer wire by dividing the wire into segments of length WL_m and predicting the MCF for each coupling capacitor separately depending on the segment location. For each segment, depending on its location in the entire wire, the values of $RSAT$ and C_{load} given to the model have to be adjusted.

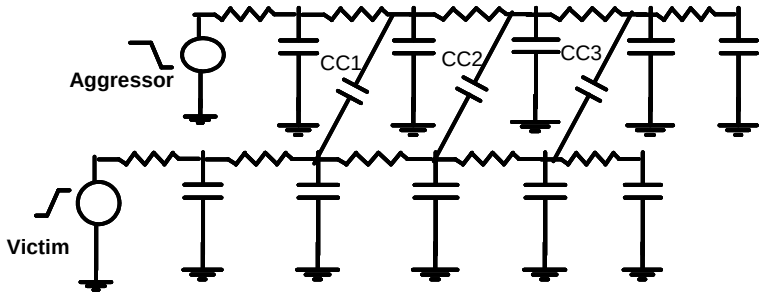
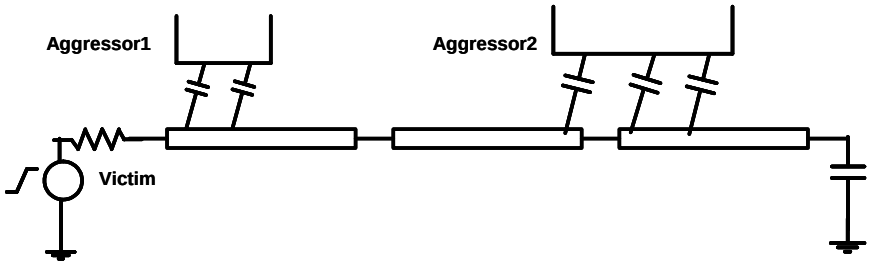


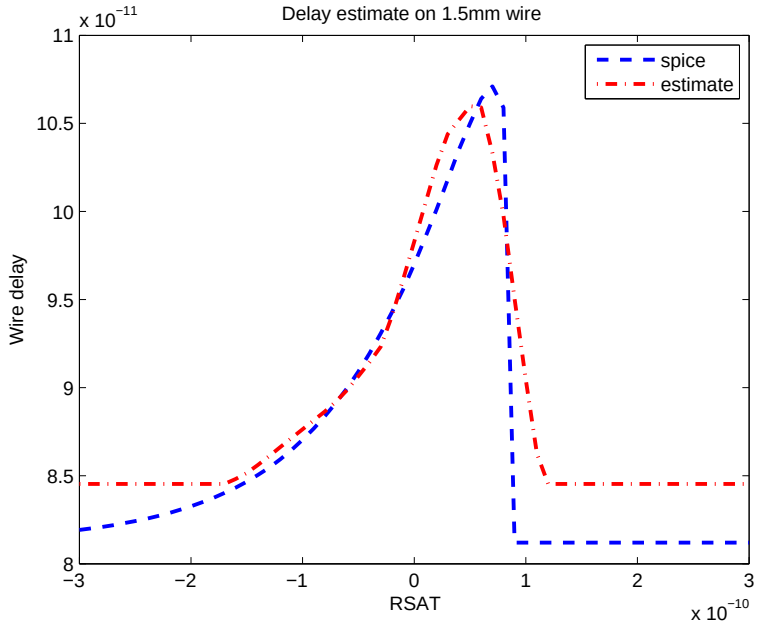
Figure 3.6: Circuit used to obtain MCF prediction model using MARS

MCF prediction model was obtained using MARS for each of the three coupling capacitors shown in Figure 3.6, since the coupling effect on far end nodes will be different from the near end nodes due to slew degradation. For multiple partially coupled aggressors, the MCF due to each aggressor is predicted by computing the corresponding $RSAT$, slews and load seen by that wire segment. The advantage of using such a regression model is that it is simple, accurate and independent of the wire model and delay models being used and thus can easily fit within an existing timing and noise methodology.

To show that the model is robust over a range of $RSAT$ values, we obtained a MCF prediction model by characterizing a wire of length 500μ and used that to estimate the coupling delay of a wire of length $1.5mm$ with two aggressors acting simultaneously as shown in Figure 3.7(a). All simulations were done for a 0.13μ technology. Table 3.1 compares the wire delay distribution obtained using Monte Carlo simulations in SPICE with those obtained using the MARS model for different values of $RSAT$ variance. The results show that wire delay variance increases with $RSAT$ variance and our model is able to estimate this change with good accuracy. For estimation using the MARS model, $RSAT$ values were enumerated and the MCF value was obtained at each point. This is much faster than SPICE simulations since it only requires evaluation of a single equation at each point. The



(a) Test circuit



(b) Estimated Vs Simulated

Figure 3.7: Wire delay estimation using MARS model for MCF

Table 3.1: Wire delay distribution estimate

σ_{RSAT} $\mu_{RSAT} = 0$(ps)	SPICE (μ, σ)(ps)	Estimate (μ, σ)(ps)	Error μ μ (%)	Error σ (%)
10	(102,1.4)	(105,1.6)	2.4	13.5
20	(103,2.8)	(104,3.0)	1.7	4.9
30	(103,4.1)	(103,3.8)	0.8	6.0
40	(102,5.2)	(102,4.6)	0.05	10.0
50	(101,6.2)	(101,5.4)	0.3	12.9

estimated MCF was then used to scale the corresponding coupling capacitance and the wire delay was then estimated using the D2M model [5]. Figure 3.7(b) shows the curves for wire delay with respect to $RSAT$ obtained using SPICE and the MARS model when only the second aggressor is switching.

3.3 Threshold Based Path Selection

In this section we describe a path selection algorithm considering coupling noise for the threshold based fault model, where the objective is to select all paths with delays above a given threshold value. Section 3.3.1 describes the pre-processing steps that need to done before the actual path selection begins. The main path selection loop is described in 3.3.2.

3.3.1 Pre-processing

Block Based Analysis: This consists of the standard breadth-first traversal of the graph to obtain the signal Arrival Time (AT) windows at each node. The AT windows (also known as switching windows) are used for estimating the minimum RSAT at any coupling site as described in the Section 3.2.1.

Threshold Assignment: Given the specified global threshold, we derive thresholds at each node of the graph. Threshold assignment is done by performing a breadth-first traversal of the graph starting from the *Sink* node. The threshold at any node is the difference between the global threshold and the delay of the longest path from that node to the *Sink*.

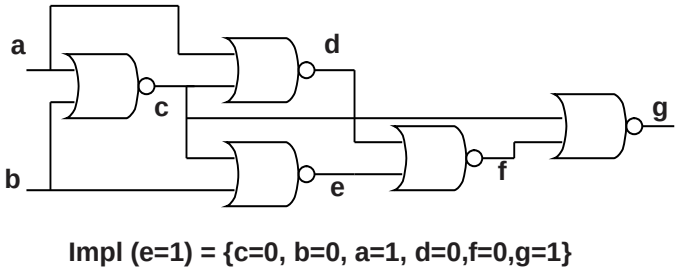


Figure 3.8: Example of static implications

Static Logic Implications: Static implications is a technique that can be used to derive logic values at different nodes in a circuit, by graph traversal without having to perform simulations. An example of how a single node assignment can determine values at various other nodes in a circuit using static implications is shown in Figure 3.8. In the illustration, if node e is assigned a value of 1, forward implications will evaluate $f = 0$. Also, backward implication at e will give $b = 0$ and $c = 0$. Since both c and f are 0, g has to be 1. Since c is 0, and b is 0, a has to be 1, which implies $d = 0$. Using static logic implications for proving false paths is a well known technique [65]. We derive the static logic implications at each node in the circuit using the algorithm given in [82], which include the forward, backward and extended backward implications. For each node, we derive two implication sets $impl_0$ and $impl_1$, where $impl_0$ and $impl_0$, are the list of implications when this node is assigned a 0 and 1 respectively. Generally, static implications generation could be time consuming; however, since it is based on only the circuit logic, it has to be computed only

once and reused for every run with a different timing library, parasitic data or threshold assignment.

3.3.2 Path Selection

The path selection procedure consists of levelized graph traversal, with path pruning at each level. The graph is traversed in a breadth-first order, starting with the *Source* node. To begin with there is a single path, containing only the *Source* node. At any node, new paths are created by adding fan-out edges to the incoming paths. The delay of the new path

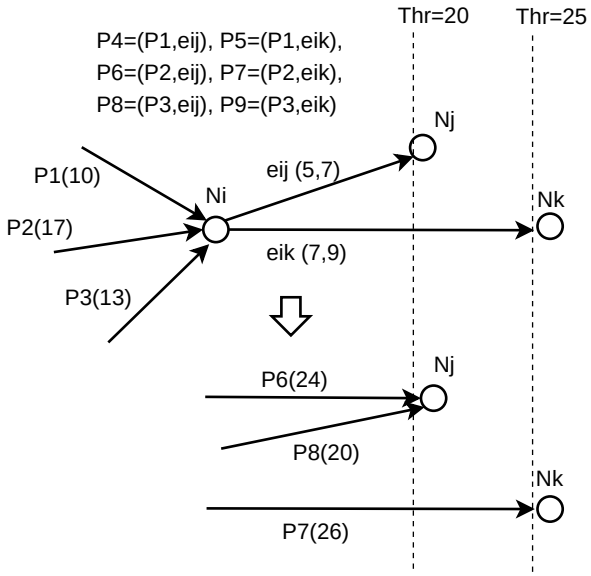


Figure 3.9: Path selection at a level

is the sum of the incoming path delay and the edge delay. The edge delay is first computed without considering the effect of coupling noise. If the delay of the new path is greater than the threshold at the next node, then the created partial path is a potential critical timing path. If the delay is less than the threshold and the new edge has aggressors, then the edge delay is recomputed considering coupling noise, since it is possible that in the presence of

coupling noise the path becomes critical. In the illustration shown in Figure 3.9, the values next to the edge are the edge delays with and without coupling. The path P_8 is formed by incoming path P_3 and edge e_{ij}, and its nominal delay is 18 which is less than the threshold for level j. However in the presence of coupling noise, the delay of edge e_{ij} will be 7 and hence delay of P_8 becomes 20, and hence P_8 can be added as a critical path at node N_j. If the path becomes critical only in the presence of coupling, then the corresponding aggressor is marked as $critical_{aggr}$. Thus a $critial_{aggr}$ is one which has to be active for a given (partial) path to become critical. Next, node assignments required for robust sensitization of the path are derived. If the path is critical only in the presence of coupling noise (i.e. it has $critical_{aggr}$), then the aggressor node assignments (transition opposite to that at the victim) are also added to the path sensitization constraints. For each of these node assignments, the corresponding static implication sets are then searched to find if there are any conflicts. If there are any conflicts between the static implications and the node assignments required for robust testing of the path, the path can be declared as robustly un-testable and hence removed from the list, else it is added to the list of incoming paths of the end node. Thus, newly created paths can be eliminated based on two factors,

1. The path delay is less than the current threshold,
2. The node assignments required for robust testing of the path has static implication conflicts,

If an edge has an aggressor which is not a $critical_{aggr}$, then it is added to the list of *relaxable* aggressors for the path, provided that the aggressor node assignment has no static implication conflicts with the other node assignments required for testing the path. This is the first level of screening which removes aggressors that can never be active along with the

path. The delay of a path being added to the list is always updated to include the worst coupling on the newly added edge. Once all nodes at the current level have propagated their incoming paths, the incoming paths on these nodes are deleted. Since new paths are created at each level, to keep memory usage in check, if the number of incoming paths is greater than a certain acceptable value at any node, then further path elimination is done by calling the SAT based ATPG process for the partial paths at that node and deleting paths that are not true.

The pseudo code of the overall algorithm is shown in Figure 3.10. The next section describes the usage of SAT solver for path based test generation.

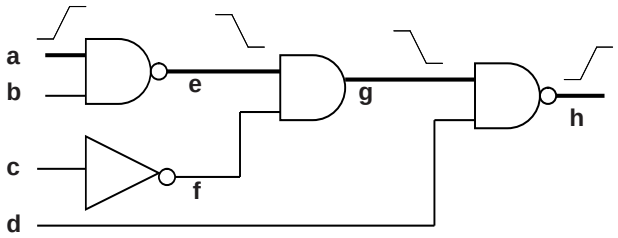
Path Selection Algorithm

```
for each incoming path P_k at node n_i
    for each outgoing edge E_ij to node n_j
            new_path = append(P_k, E_ij)
            PD = PathDelay(P_k) + EdgeDelayWithoutCoupling(E_ij)
            PD_cc = PathDelay(P) + EdgeDelayWithCoupling(E_ij)
            if ( PD > Threshold(n_j) )
                 curr_assgn = RobustSensitization(new_path)
            else if ( PD_cc > Threshold(n_j) )
                curr_assgn = RobustSensitization(new_path) U AggressorConstraints
            else
                delete new_path
            end if
            if ( !StaticImplicationsConflict(curr_assgn) )
                PathDelay(new_path) = PD_cc
                AddIncomingPath(n_j , new_path)
            else
                delete new_path
            end if
    end for
end for
```

Figure 3.10: Path selection algorithm

3.3.3 Identifying the Worst Aggressor Subset

The initial path screening phase retains all paths that have their maximum delay greater than the global threshold and have no static implication conflicts. The initial delay estimate for these timing critical paths assumed all (relaxable and critical) aggressors to be active. This means that each aggressor on path must be switching in the direction opposite to its corresponding on-path victim node. However the circuit logic constraints might not allow all these aggressors to be active at the same time. Hence the maximum path delay corresponds to the case when the subset of aggressors that together induce the maximum possible delay is active. This problem is a constrained satisfiability problem which requires both satisfiability and optimization. For each selected path, we use a SAT solver to first find if it is robustly testable, and if it is, we formulate and solve a Weighted Partial Max SAT (WPMSAT) problem to find the worst subset of aggressors that can be active together.



$\mathbf{Path\_CNF = (a'_s)(a_v)(b_v)(e_s)(e'_v)(f_s)(f_v)(g_s)(g'_v)(d_s)(d_v)(h'_s)(h_v)}$

Figure 3.11: Example of path CNF

Modern SAT solvers such as zchaff [54] enable incremental SAT solving, which is an efficient method of solving multiple SAT instances that differ in a small number of clauses. This is done by assigning clauses with *GroupIDs*. Groups of clauses can then be added and deleted to create new SAT instances. This method leverages the clauses learned from the clauses which are common among the various SAT problems. In our case, the circuit

constraints are common for all paths. The constraints required for robustly testing any path are formulated in a group of clauses called the $Path_{CNF}$. For any node n, two variables, n_s and n_v are created to represent the node state in $V1$ and $V2$ (the two vectors required for delay test) respectively [17]. For example, consider the circuit in Figure 3.11. To robustly sensitize path $a - e - g - h$, the side inputs b, f, d also need to assume certain values. The clauses corresponding to the constraints on the on and off-path signals are shown in the figure. The constraints corresponding to the $critical_{aggr}$ are also added to the $Path_{CNF}$ since the path cannot be critical without these aggressors switching. After the SAT solution is obtained, all the clauses in the $Path_{CNF}$ group are deleted before processing the next path. The following subsection describes how to obtain the constrains on the relaxable aggressor so as to maximize the coupling noise.

The Weighted Partial Max SAT is a constrained satisfiability problem, in which certain constraints (clauses) are hard while the rest of the constraints are soft or relaxable. Each relaxable clause that is not satisfied incurs a certain cost to the solution. The objective then is to find a solution that minimizes the cost. For any path, the original circuit clauses and the clauses present in $Path_{CNF}$ are considered as hard constraints, while the clauses corresponding to the relaxable aggressor nodes are treated as soft constraints. Each aggressor has a weight associated with it depending on the maximum delay it can introduce on the path. The aggressor weights are computed in the path selection phase itself. The objective is to find a solution such that the total cost due to relaxing aggressor clauses is minimized. A simple method of implementing this is by adding a new *relax variable* corresponding to each aggressor (relaxable) clause. The aggressor weight becomes the corresponding variable's cost and the problem can be formulated as follows.

Given a Boolean formula ϕ with n variables $x_1, x_2, \ldots, x_n$ with cost $c_i \geq 0$, find a

variable assignment $X \in 0,1^n$ *such that* X *satisfies* ϕ *and minimizes*

$$C = \sum_{i=1}^{n} c_i x_i \tag{3.6}$$

where $x_i \in 0,1$ *and* $1 \leq i \leq n$.

In our case, only the relax variables will have a cost $c_i > 0$, while all other variables will have $c_i = 0$. Thus this problem can also be seen as a MinCostSAT problem [28] with only a few variables with non-zero cost. The MinCostSAT problem has been well researched and typically a branch-and-bound search technique such as A^* search is used to find a solution. Such a technique prunes the search space based on two criteria: either there is a conflict, or the minimum estimated cost (CurrentCost + LowerBound) of the sub space is higher than the best known solution (Upper Bound). The algorithm is complete, i.e. it gives the optimal solution if the lower bound is never overestimated. This means that the estimated lower bound should never be greater than the actual lower bound, otherwise we might prune a sub space that could have the optimal solution.

In [28], it was shown that for problems with many zero-cost variables, it is actually beneficial to use no lower bound at all. Since our problem has only a few variables with non-zero cost, the lower bound of a sub-space is computed by assigning all unassigned variables to 0, and hence the lower bound is always 0. Once the clauses in the $Path_{CNF}$ are added to the clause database, the SAT solver is first called to ensure that the hard constraints are satisfiable. If they are, then the relaxable clauses corresponding to the aggressor constraints are added. For instance, let a given path have three aggressors $a1, a2, a3$, and the coupling noise and hence path delay be maximized if the aggressors have the following assignments $a1 = R, a2 = F, a3 = R$. Since each aggressor node has two variables associated with it, a total of 6 relax variables $r1, r2 \cdots, r6$ will be required corresponding to the 6 relaxable

clauses as follows:

$$(a\bar{1}_s + r1)(a1_v + r2)(a2_s + r3)(a\bar{2}_v + r4)(a\bar{3}_s + r5)(a3_v + r6) \tag{3.7}$$

Thus each aggressor has two clauses and two relax variables associated with it. Assigning any of the relax variables to 1 would mean that the aggressor constraint has been relaxed. Thus finding an optimal solution would require searching through all possible assignments of the relax variables to find a satisfiable and minimum cost solution. The branch-and-bound search algorithm shown in Figure 3.12 performs a best-first search that prunes sub spaces that are inconsistent. The relax variables are sorted, and in each step the next best variable is selected for decision making. A state is either a partial assignment of the relax variables or a complete *Solution* and the cost represents the aggressor clauses that have been relaxed. A state is *Consistent* if the sum of the cost of the state and the lower bound, is lower than the last known best solution (UpperBound) and there are no conflicts. Checking the consistency of a state would therefore require branching and making decisions over the relax variables within the SAT solver. We do this by adding extra clauses that enforce the given state on the relax variables. For instance in the above example, to check the consistency of the state $r1 = 0, r2 = 0, r3 = 1, r4 = 0, r5 = X, r6 = X$, the following new clauses need to be added.

$$(\bar{r1})(\bar{r2})(r3)(\bar{r4}) \tag{3.8}$$

This state corresponds to the partial assignment state where aggressor $a1$ has to be active, $a2$ can be relaxed and $a3$ has not been assigned. If this state is consistent, the search will decide on whether $a3$ needs to be relaxed and if the cost is acceptable. This method of adding extra clauses to check the consistency of any state has the advantage that it is independent of the variable decision strategy used by a particular SAT solver and can be applied to any solver.

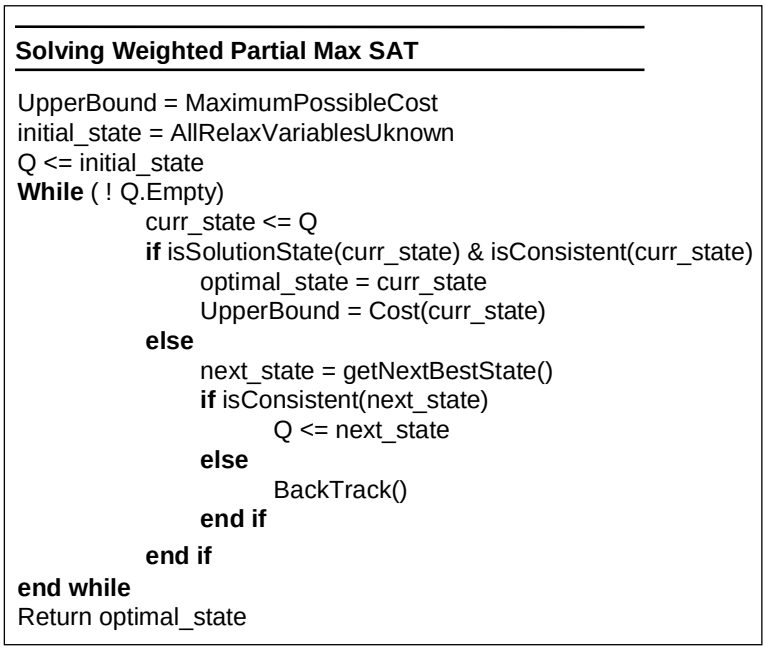


Figure 3.12: Algorithm for solving WPMSAT

3.3.4 Reducing the Search Space

Since we solve an optimization problem for each path, the runtime can increase significantly. A path with k aggressors has 2^{2k} possible solution states. This search space can be drastically reduced by doing some preliminary elimination. The first step is to check if each aggressor can individually be activated along with the path constraints. This requires k SAT calls, but reduces the search space significantly. Next, the relaxable clauses corresponding to the V1 vector only, are added to the SAT instance and the MinCost solution is obtained. This eliminates the aggressors that have conflicts in the V1 vector assignment and the maximum number of SAT calls in this step is $2^{k'}$, where k' is the number of aggressors left after the first elimination. Finally, the MinCost problem is solved again by considering relaxable clauses corresponding to the V2 vector of the remaining aggressors. The algorithm

returns with a solution that has minimum cost and hence maximum utility, which in our case is an optimal aggressor combination that introduces maximum delay on the path. Since the initial path delay was estimated assuming that all relaxable aggressors are active, the path delay is recomputed by considering only the aggressors from the optimal combination. If the path delay is still greater than the global threshold, then it can be retained as a critical path, else it can be discarded. In this research, the objective was to select the critical paths by estimating the maximum path delay considering coupling noise. The test vector generated here is for the best aggressor combination assuming minimum RSAT for each aggressor and the logic constraints on the aggressor signal arrival paths are not considered to keep the problem tractable.

3.4 Experimental Results

The framework described above has been implemented in C++. A Timing library for a set of standard cells was generated by characterizing pin-to-pin delays at different input slopes and output load values using the 130nm BPTM model. The experiments were done for some of the ISCAS89 benchmark circuits. Each test circuit was synthesized using the standard cell library, and a commercial automatic place and routing tool was used for physical synthesis. The wire length and coupling capacitance information for each net was then extracted from the layout. Figure 3.13 shows the block diagram of the complete framework. For these experiments, the coupling capacitances which form more than 50% of the total wire capacitance were selected as potential coupling sites. For each circuit, the global threshold was assigned to be 70% of the maximum delay at the sink node found during the initial static timing analysis. We use the zchaff [54] solver for generating the test vectors for the selected paths. Table 3.2 shows the observed results for each test circuit.

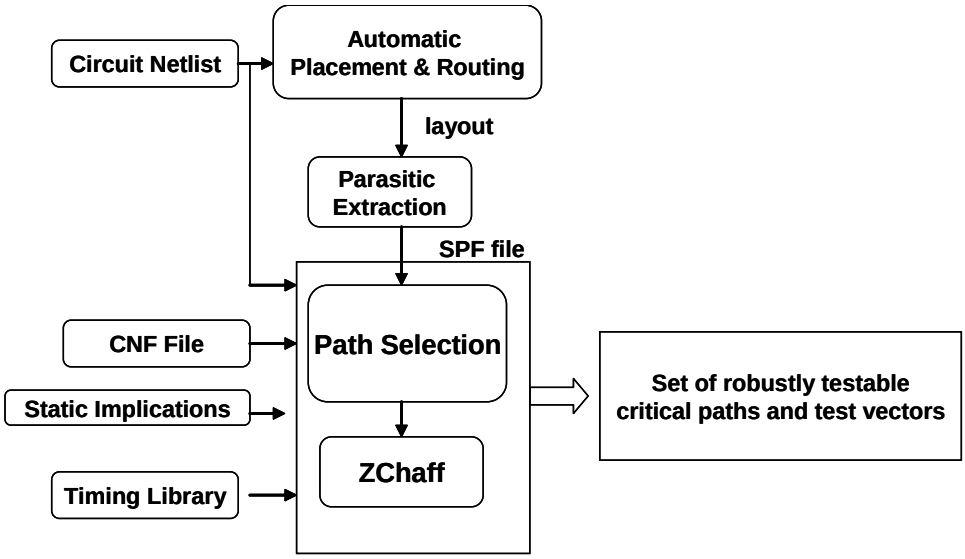


Figure 3.13: Complete path selection framework

The first column is the number of critical robustly testable paths selected ($\#CP$) using our algorithm. The number of aggressors will be different for each path and for each circuit and we have listed the average number of total aggressors per path (NTA), average number of relaxable aggressors obtained after eliminating aggressors with static implication conflicts (NRA), and finally the average percentage of active aggressors found in the optimal aggressor combination after solving the WPMSAT problem (PA). The table also shows the average number of calls to the SAT solver per path (N_{SAT}) required to find the optimal aggressor combination and the corresponding computation time for test generation. The results show that for any path, only a small subset of aggressors can be active at any given time. Thus unlike [44], where the top few aggressors are selected, our method finds the worst subset of aggressors that can be active at any time.

To emphasize the importance of estimating the 'real' worst case path delay during path selection, we compare the number of critical (testable) paths selected using our al-

Table 3.2: Per-path aggressor statistics

Circuit	$\#CP$	Per Path				
		NTA	NRA	PA (%)	NSC	T(s)
s510	58	12.7	6.0	34	12	0.03
s641	57	16.4	14.6	65	32	0.11
s713	54	26.7	18.3	59	33	0.14
s953	228	8.2	3.4	28	5	0.03
s1196	176	12.2	6.2	25	9	0.04
s1238	94	12.6	6.6	30	10	0.04
s1423	3469	22.9	11.13	30	20	0.72
s1488	304	8.5	4.5	26	6	0.05
s3271	4671	9.7	6.2	33	9	0.27
s3330	718	6.8	5.3	52	10	0.21
s3384	1891	19.6	10.7	42	20	0.62
s5378	1883	4.8	3.9	74	9	0.41
s9234	274	8.3	4.9	67	9	0.32
CP: CriticalPaths, NTA: NumTotalAggressors NRA: NumRelaxableAggressors,PA: PercentActiveAggressors NSC: NumSATcalls, T: Time						

gorithm with two methods. The *optimistic* case represents the methods which ignore the effect of coupling noise during path selection as done in [77], [48], [72] . Intuitively, using such *optimistic* estimates could result in missing some critical paths and is confirmed by our results. The *pessimistic* case is the one where static timing analysis based methods are used for estimating the maximum path delay with coupling noise. Using such *pessimisitic* estimates can result in categorizing many non-critical paths as critical and hence increasing the test time. Table 3.3 shows the number of paths selected using the three methods, the percentage of critical paths that will be missed if we use *optimistic* path delays, and the percentage of non-critical paths that will be selected if *pessimistic* delay estimates are used. For the pessimistic estimates, the MCF at each coupling site was estimated using the minimum-RSAT, which is still less pessimistic than using a constant MCF of 2 for all

Table 3.3: Comparison with previous methods

Circuit	OurMethod	Optimistic		Pessimistic	
	$\#Paths$	$\#Paths$	Missed (%)	$\#Paths$	NCP (%)
s510	58	48	17	69	16
s641	57	47	17	69	17
s713	54	41	25	80	32
s953	228	197	13	278	18
s1196	163	142	19	194	9
s1238	94	73	22	117	19
s1423	3063	2829	18	3779	8
s1488	304	290	4	343	11
s3271	4671	4469	4	5089	8
s3330	594	718	17	759	5
s3384	1891	1719	9	2147	12
s5378	1883	1419	24	2055	8
s9234	274	212	22	326	16
NCP: Non-Critical Paths					

coupling sites. Note that in the (*optimisitc*) case, simply reducing the threshold for critical path selection is not a good solution since it can result in selecting a lot of non-critical paths. Also with the current trend of large percentage of paths being near critical [78], the number of paths being selected will increase drastically. In order to minimize the test time, it is therefore important to select the real critical paths. Since pessimistic estimates do not take into account aggressor logic constraints, it can lead to incorrect ranking of paths resulting in missing the real critical paths. Path ranking is important when the top K critical paths are to be selected for test. This effect again will be more prominent in circuits with a large number of near-critical paths. Modern circuits with high timing optimization during synthesis, tend to have many near critical paths, which emphasizes the importance of employing our path selection technique. Table 3.4 shows the percentage of paths missed if we use the

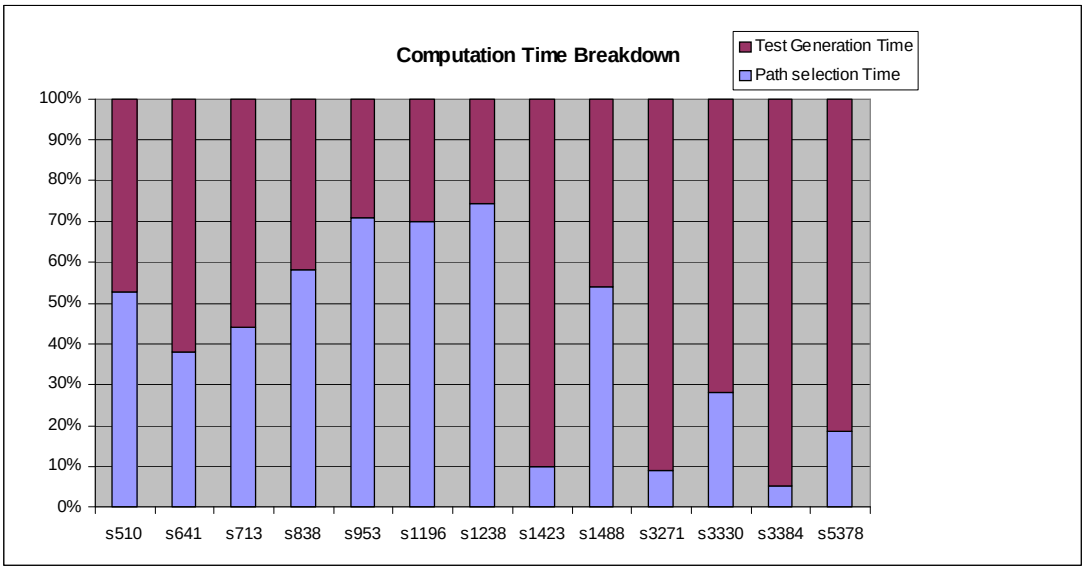
Table 3.4: Top 100 testable paths

Paths missed					
	s3271	s3330	s3384	s5378	s9234
Optimistic	16	9	10	17	10
Pessimistic	23	10	13	7	8

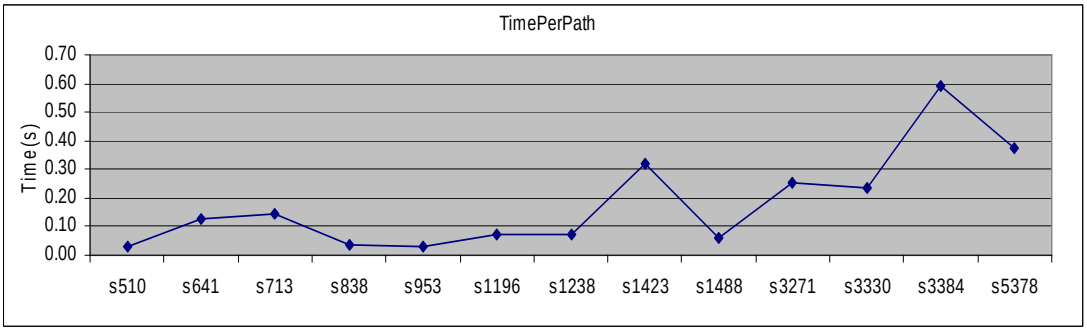
pessimistic delay estimates as compared to our technique for selecting the top K paths for few of the circuits. The total computation time required for each circuit can be broken down into path selection time and test generation time as shown in Figure 3.14(a). Figure 3.14(b) shows the average per path computation time required for each circuit.

3.5 Conclusions

During critical path selection for delay testing, it is very important to estimate path delays accurately. Coupling noise is one of the important factors contributing to dynamic path delay variation. As a part of this research, we have developed a methodology to estimate the maximum path delay with coupling noise considering both logic and timing constraints. The problem of estimating the maximum coupling delay for a given path is formulated as a Weighted Partial Max SAT problem and an efficient technique to solve this problem using clause relaxation is presented. Our experimental results show that ignoring aggressor logic constraints when estimating path delay with coupling noise can not only result in selecting many non-critical paths, but also in missing some real critical paths.



(a) Computation time distribution



(b) Computation time per path

Figure 3.14: Computation time

Chapter 4

Design for Accurate Delay Test and Characterization

4.1 Introduction

AC scan test methods involve setting the circuit to a certain state using scan, generating transitions on the paths to be tested and then conducting an at-speed capture to verify if the timing is met. Since delay test is a parametric test, the effectiveness of delay testing depends not only on the number of circuit nodes for which tests are obtained but also on the size of delay defects that can be detected using the given tests [4]. It has been shown that in nanometer technologies, circuits have an increased susceptibility to small delay defects [62] and hence delay tests need to be targeted for detecting very small delay changes. For path based tests, the size of delay defects that can be detected using any path depends on the timing slack of the path being tested.

Most ATPG based solutions focus on identifying the longest testable paths through any fault site and generating test vectors that maximize the path delay [65]. Since longer paths have smaller slacks, timing violation can be caused by smaller defects, which makes defect detection easier. However, the effectiveness of this method is inherently limited by the distribution of path delays in the test circuit. For instance, for some nodes, even the longest paths can have large slacks, thus failing to detect defects below a certain size [4]. Another approach to reducing the slack interval during test is to increase the frequency of the capture clock [7]. In [79], a test scheme to detect small delay defects by performing multiple captures in the slack interval was described but solutions to provide faster than

at-speed test frequencies were not discussed. A promising solution for generating fast test clock frequencies is to generate the test clocks on the Device Under Test (DUT) itself.

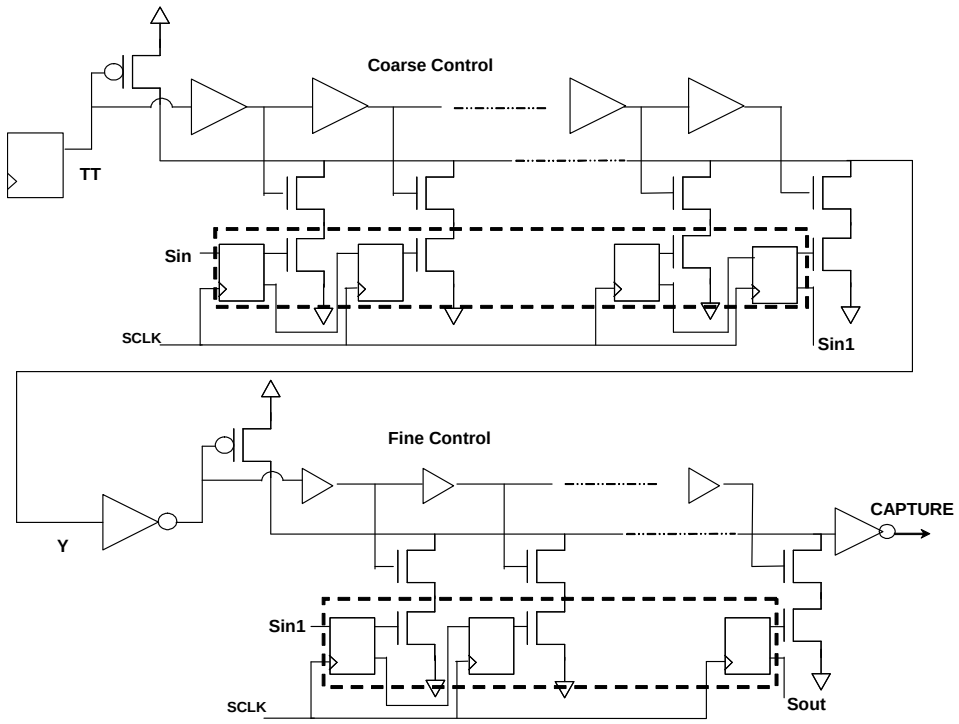
In [64], a switching circuit was shown that allows a programmed number of PLL clocks to be sent on the system clock during delay test. This allows at-speed broadside test, also known as Launch on Capture (LOC) scan test, but not faster than at-speed test. A similar scheme for multiple clock domains was described in [10] and a technique that can facilitate at-speed test for LSSD designs was proposed in [36]. Another technique that uses an on-chip clock chopper and gating logic to obtain a fast test clock using slow tester clock was proposed in [60], but it requires additional pins on the chip. Most on-chip test clock generation schemes proposed in the past obtain the test clock using the on-chip PLL. This facilitates at-speed delay testing, but cannot be easily extended to faster than at-speed tests since this would require re-programming the PLL clock. In [57], a clock control circuit that uses the PLL clock for performing faster than at-speed capture during delay test was discussed. It consisted of a clock chopping circuit where control registers called 'chop registers' are programmed to generate predetermined chopped waveforms using the PLL clock. The limitation of this scheme is that it involves resetting the PLL at the beginning of each pattern and also requires extra select pins be added to the chip. Also, the dynamic range of the capture clock provided using such a technique is very limited. In [37] a technique that uses multiple time shifted clocks and complex selection logic for obtaining faster than at-speed tests was described. However, an important limitation of this scheme is that it can be used only in Broadside delay test. This issue also persists in most of the techniques that derive both the launch and capture pulses from the on-chip PLL. This is because in the LOC method, both the launch and capture clocks need to be applied after the Scan Enable (SE) signal has been disabled and the system is in functional mode. For the Launch on Shift

(LOS) method, on the other hand, the launch clock needs to be applied while the system is in scan mode, and the capture clock should be applied when the system is in functional mode. Thus, if the at-speed test is to be done for the LOS method using the PLL clocks directly, switching the scan-enable between the launch and capture events, becomes a challenge. It has been shown that the Launch on Shift (LOS) method can give much better fault coverage and pattern count than the Launch on Capture method [36]. Therefore, it is desirable to develop a design solution that allows both LOS and LOC based AC-scan test to be used for at-speed and faster than at-speed delay test.

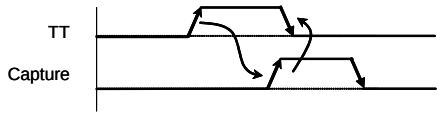
In this work, we describe a technique which facilitates programming the required test clock frequency within the test vector itself. The technique can be applied to any of the scan based delay test methods, namely Enhanced Scan (ES), Launch On Shift (LOS) and Launch On Capture (LOC), with minimal area overhead. The test clock frequency can be controlled with good resolution, allowing not only at-speed but *faster than at-speed* test. The technique provides arbitrary control over the test clock period such that during test, the real path delay (and hence the slack) can be measured, instead of just checking if the path under test meets timing. The proposed technique is valuable not only for detecting small delay defects but also for timing characterization during silicon validation and debug. Another advantage of having such a scheme is that tests need not be restricted to long paths, as the slack on the test paths can be controlled by setting the test clock frequency.

4.2 Capture with Programmable Delay

Delay test consists of launching a transition at a start point (input or scan flop) and then capturing the propagated signal at an end point (output or scan flop). The test clock frequency is determined by the delay between the *launch* and the *capture* clock signals.



(a) Schematic



(b) Timing Diagram

Figure 4.1: Programmable Capture Generator (PCG)

The exact application of the launch and capture clocks is highly dependent on the type of ac-scan test method being used (ES, LOC or LOS). The objective of our proposed scheme is to be able to have arbitrary control on the test clock frequency by controlling the generation of the capture signal from the launch clock. The schematic of the Programmable Capture Generator (PCG) circuit is shown in Figure 4.1(a) and the basic timing diagram is shown in Figure 4.1(b). The input to the circuit is the *TT (Test Trigger)* signal and the output is the *CAPTURE* signal. A rising transition on the *TT* signal should coincide with the *launch*

event which triggers the circuit to generate the $CAPTURE$ signal after a programmable delay. The PCG circuit consists of two delay lines, one using coarse delay buffers and the other using fine delay buffers. The required delay between the launch and capture signals can be programmed using a series of scannable flops that need to be set to a one-hot code during scan mode. The two delay lines ensure a good dynamic range of the test clock, without having to use too many buffer stages. As long as TT is 0, the output is not affected due to the changing register values during the scan operation, and the $CAPTURE$ signal will remain stable at 0. When TT is asserted to 1, the NMOS corresponding to the flop which has the value 1 stored, gets a rising signal after a certain delay through the delay line. The node Y drops to 0, which acts as the trigger to the fine delay controller. The fine control operates exactly the same as the coarse delay part, so that the capture signal has a rising transition after the programmed delay.

4.3 Incorporating PCG in Delay Test

In this section, we discuss the techniques to incorporate the PCG circuit into various ac-scan test frameworks. Typically, the system functional clock $FCLK$ is provided by an on-chip PLL while the scan clock $SCLK$ is provided by an external ATE. When the Scan Enable (SE) signal is 0, the system is in functional mode with $FCLK$ being sent on the system clock tree, while for $SE = 1$ the system is in scan mode and all system flops get $SCLK$. Figure 4.2 shows a general block diagram of a system that uses the PCG circuit for controlling the delay test. The CLK_SELECT_LOGIC block controls the clock signal that goes to the system flops and its implementation details along with the TT_GEN logic block is dependent on the type of AC-scan method being used, namely ES, LOS or LOC. Since the PCG circuit will provide the required system clock during delay testing, it is important

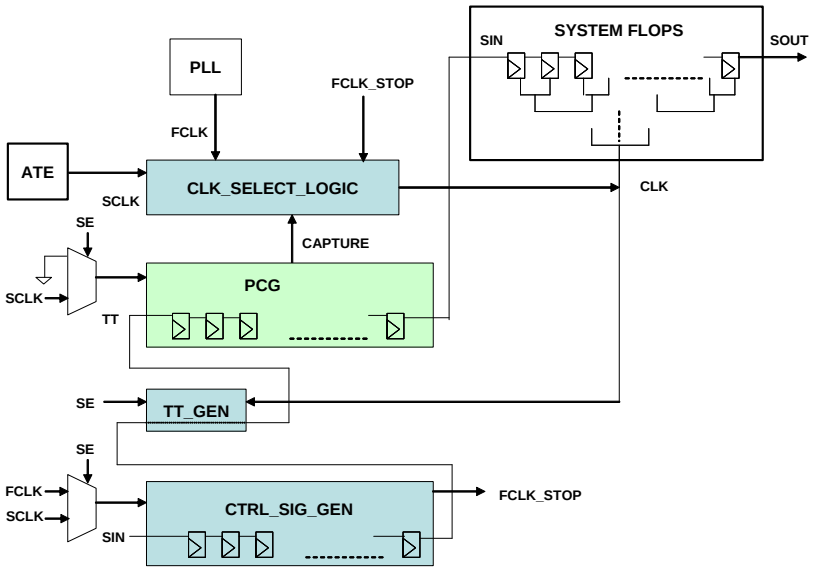


Figure 4.2: System using the PCG

to block the functional clock from being sent on the clock tree during delay teasing. A simple way to do this is by having an additional control pin called *DTEST_MODE*, which can be used to block the functional clock. If an external pin is not desired, then a control signal can be generated internally using the *CTRL_SIG_GEN* block. The schematic of the *CTRL_SIG_GEN* circuit is shown in Figure 4.3. It consists of a decrement counter that can be programmed to an initial value during scan. The counter starts down counting when the Scan Enable (SE) signal is de-asserted. The output of the counter is the *FCLK_STOP* signal which is used in the *CLK_SELECT_LOGIC* block. The *FCLK_STOP* signal is used to stop the *FCLK* signal from being sent to the system clock tree for a given interval after the *SE* signal is deasserted. The counter should be programmed such that *FCLK_STOP* signal remains high for a long enough time, so that *SE* signal can be asserted again and the results can be scanned out. The following sub-sections discuss the details of the control scheme in the context of different AC-scan methods.

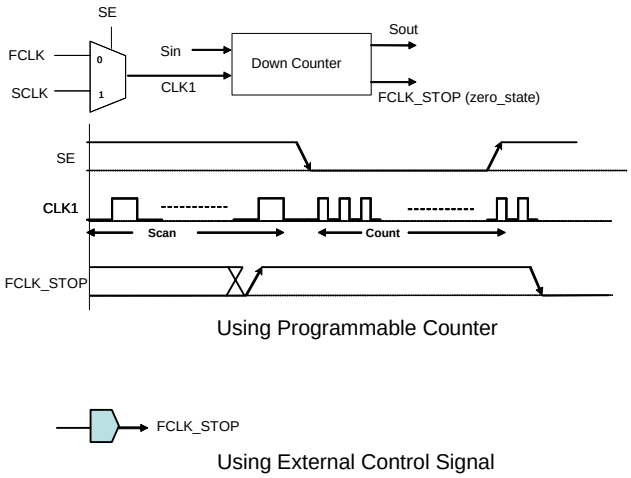


Figure 4.3: Generating the $FCLK\_STOP$ signal

4.3.1 Enhanced Scan

In the Enhanced Scan scheme, each scannable flop is a Scan-Hold-Flip-Flop, which provides an extra control signal called $Hold$ that can be used to hold the flop output to the previous value while new value is being stored in the flop. The advantage of Enhanced Scan is that the inputs to the combinational logic do not switch during the scan operation. The delay test procedure involves scanning in the V1 vector first in scan mode and then applying the V1 values by de-asserting the $Hold$ signal, followed by the scanning in of the V2 vector. During the scan operation, again the $Hold$ signal is held at 1 and when V2 vector is completely scanned in, the SE signal is de-asserted to enter the functional mode and the $Hold$ signal is also de-asserted to launch the required transitions for delay testing. A fast clock signal needs to be generated at this point to capture the results of the input transition. In order to use the PCG circuit for generating the fast capture clock, the TT signal needs to be generated exactly at the launch event. For Enhanced Scan, this can be done by having the TT register as a part of the system scan chain and connected to the

system clock as shown in Figure 4.4. The TT register should be set to 0 in $V1$ and 1 in $V2$, so that at launch, a rising transition is generated on TT as required. Figure 4.5 shows

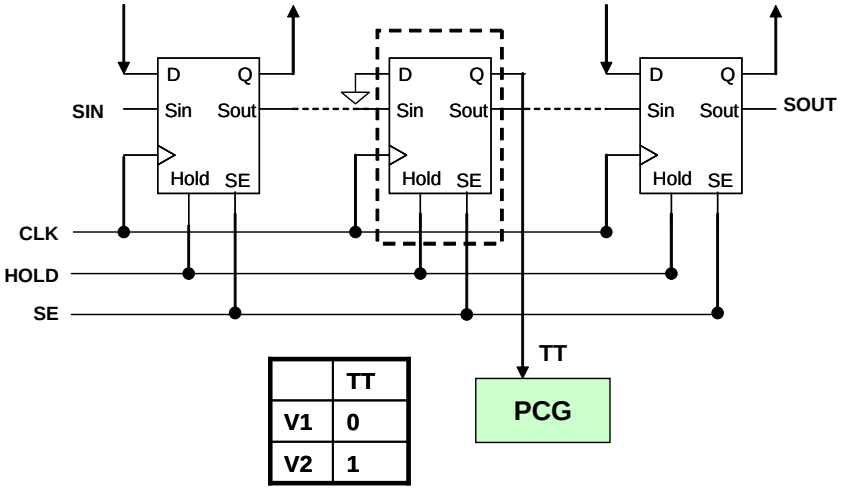


Figure 4.4: Trigger generation for Enhanced Scan

the details of the CLK_SELECT_LOGIC block and the corresponding signal diagram for the Enhanced Scan scheme. When $SE = 1$, the system is in scan mode, and when $SE = 0$, the $FCLK\_STOP$ signal blocks the functional clock and instead the $CAPTURE$ clock generated from the PCG circuit is sent on the system clock. The rising transition on TT is caused due to the launch event ($HOLD$ is de-asserted), which causes the rising transition on the $CAPTURE$ signal. Since the TT register also gets the system clock, while it is in functional mode, it captures a 0, which in turn generates the falling transition on the $CAPTURE$ signal. Here, for simplicity we assume that the SE and $Hold$ signals are tied together so that the SE signal can be used as the indication for launching $V2$. The amount of time the $FCLK$ will be blocked can be controlled by programming the down counters inside the CLK_SELECT_LOGIC circuit. This should give ample time for the ATE to assert the SE again and scan out the results that were captured.

4.3.2 Launch on Shift

The disadvantage of Enhanced Scan is that it entails a high overhead in terms of area, since each flop needs to have the extra hold latch. Also the test time is almost doubled since both $V1$ and $V2$ need to be scanned in completely. In addition to that, it requires an extra *Hold* signal to be routed throughout the system to each scannable flop. The Launch-on-Shift strategy alleviates these problems by adding an extra constraint that the $V2$ vector be derived by shifting $V1$ by one position. Thus after $V1$ is scanned in, a single clock pulse needs to be applied on $SCLK$ to obtain the $V2$ vector. This last shift clock also corresponds to the launch event (launching the required transitions). Immediately after that, the SE signal needs to be de-asserted to go to functional mode, and a fast clock signal has to be provided for capturing the results in functional mode. This introduces new design requirements, in that the SE signal needs to switch between the launch and capture events (at-speed). In [30] a new flop structure was proposed to alleviate this problem, but the cost of the solution increases with the increasing number of scan flops in the design. We describe a technique for incorporating the PCG circuit in the Launch-On-Shift test method, while relaxing the stringent timing requirements on the SE signal. The concept is similar to that in [3] in that the required scan enable is generated on-chip. The requirement for the LOS method is that the last shift ($V2\_CLK$) should happen in scan mode while the capture should be done in functional mode. To satisfy this requirement, we generate the $V2\_CLK$ on-chip, instead of applying it on the $SCLK$ signal. The primary idea is that the external SE signal (from the ATE) should be de-asserted after the $V1$ vector is scanned in, but internally, the system (DUT) is still maintained in scan mode using the $\widehat{SE}$ signal. The $\widehat{SE}$ signal is obtained by delaying the SE signal obtained from the ATE. During this extended scan mode, the last scan clock pulse ($V2\_CLK$) is generated on-chip and sent on the clock network. Thus

the SE signal can de-asserted any time after $V1$ is scanned in and the $V2\_CLK$ will be generated only after SE is de-asserted. The control circuit used to generate the $V2\_CLK$ signal is shown in Figure 4.6. The trigger signal (TT) for generating the capture signal needs to be asserted exactly when $V2$ is launched. This can be achieved by having the TT register and a preceding DMY register as a part of the system scan chain as shown in Figure 4.7. The TT register is made using the Scan-Hold-Flip-Flop that is used in Enhanced Scan and the $Hold$ signal is tied to the SE signal. This prevents the capture signal generation from getting triggered during the scan operation. Initially, when $SE = 1$ and $V1$ is being scanned in, the output of TT will remain stable since $HOLD = 1$ on the TT register. The $V1$ vector should set the DMY register to 1 and TT to 0. After $V1$ is scanned in, the SE is de-asserted which also disables the $Hold$ signal on TT. After the SE signal is de-asserted, the $V2\_CLK$ pulse is generated and is sent to the system clock tree, while the system is still in scan mode. The last shift caused by $V2\_CLK$ causes the value from the DMY register to be shifted into TT, thus generating a rising transition on TT which triggers the PCG circuit. The $\widehat{SE}$ signal is de-asserted with the falling edge of the $V2\_CLK$, which ensures that the system goes to functional mode, and sends out the $CAPTURE$. Implementation of the CLK_SELECT_LOGIC and the signal diagram for the LOS scheme are shown in Figure 4.8. While $SE = 1$, the system gets $SCLK$, and after SE is de-asserted, the $V2\_CLK$ signal is sent out on the system clock network (launch) while the circuit is still in scan mode ($\widehat{SE} = 1$). The system goes to functional mode when $\widehat{SE} = 0$, but $FCLK\_STOP$ signal selects the $CAPTURE$ signal generated from the PCG circuit for a fast capture. One limitation of this scheme is that the capture clock period, is limited by the pulse width of the $V2\_CLK$. In our implementation, the $V2\_CLK$ is derived from the $FCLK$. Thus if a 50% duty cycle is assumed for $FCLK$, then the maximum capture clock frequency that

can be obtained is $2 \times FCLK$. Various circuit tricks can be done to reduce the pulse width of $V2\_CLK$, to allow faster capture clock frequencies, but we do not address this problem here.

4.3.3 Launch on Capture

In the Launch-on-Capture (also known as Broadside) scheme, the SE signal does not have to be fast. The $V2$ vector is derived by capturing the results of the $V1$ vector. Thus after the $V1$ vector is scanned in, the SE signal is de-asserted and two fast clock pulses need to be generated in functional mode. The first one captures the results of $V1$ and simultaneously launches $V2$, and the second pulse captures the results of $V2$. In order to incorporate the PCG circuit in the LOC test method, we need to add a circuit that allows exactly one $V2$ pulse at functional clock speed followed by the programmable capture pulse. The circuit used to generate the $V2\_CLK$ signal in the LOS scheme is also used here, except that the $V2\_CLK$ is provided in the functional mode. We again use the TT and the DMY registers, except that only the DMY register is part of the scan chain while the TT register is a regular flop as shown in Figure 4.9. While $V1$ is scanned in and $SE = 1$, the output of the TT register remains 0. The $V1$ vector should set the DMY output to 1, and at the $V2\_CLK$ (launch), the TT register gets the 1 from the DMY register, thus providing the required trigger signal for generating the $CAPTURE$. After SE is de-asserted, the $FCLK\_STOP$ signal, is used to prevent the $FCLK$ from being sent out and instead the $V2\_CLK$ followed by the controlled $CAPTURE$ is given to the circuit. The SE signal needs to be asserted again before the $FCLK\_STOP$ signal goes 0, and then scan out the results.

4.4 Applications of PCG

The PCG circuit provides precise control over the capture edge and hence can be used for measuring the slack of any path. The following sub sections describe the potential applications in more detail.

4.4.1 Detecting Small-delay Defects

An important parameter to measure the effectiveness of delay test is the size of delay defects that can be detected (defect coverage) [4]. It has been observed that certain interconnect based defects such as resistive opens and bridges cause very small delay increments and hence can easily escape traditional at-speed test [79] but affect circuit reliability. Even if the longest path through each node is selected for test, a significant number of paths could be intermediate length paths or short paths [4] and hence have low defect coverage. It was shown in [79] that the defect coverage can be greatly improved by performing multiple captures in the slack interval. This would require the capability of performing faster than at-speed tests. Using the PCG technique, the test clock frequency can be programmed as a part of the test vector, thus enabling faster than at-speed tests. Since the capture clock frequency can be controlled, the size of defects that can be detected is not limited by the path length. The PCG technique provides the ability to measure path length by sweeping the capture clock, so that paths with lower delay variability can be selected to improve defect coverage. A practical limitation of the PCG technique is that each test path could require a different test frequency set up and hence it limits the number of paths that can be tested in parallel. This can become a challenge in employing the PCG technique for high volume manufacturing tests as it can increase the test time due to lower test compaction. However, since we are no longer limited to using long paths only, path selection algorithms can be

developed to select paths with similar delays that can be tested together. Part of our future work will focus on developing efficient algorithms that exploit the trade-off between defect size coverage and test compaction to facilitate the application of the PCG technique in high volume manufacturing tests.

4.4.2 Measuring the Path Slack

Circuit timing is affected by various factors such as process parameters, supply voltage noise, capacitive coupling and temperature. Each factor introduces variability in circuit delay, and the complex interactions between various parameters is difficult to model. Ideally, to ensure that a circuit meets timing, it needs to be validated at all possible worst case timing scenarios. However, due to the increasing delay variability and large number of parameters involved, identifying the real worst cases has become an intractable problem. Traditional delay test mechanisms check if a given path meets timing for the applied test vector, but cannot check if the path will meet timing for all possible vectors and all possible process corners. For instance, if the timing target is $1ns$, and the delay of a critical path is $950ps$, then it still passes the test, but the path has a very small slack and it is possible that for some vector that introduces a large supply voltage noise, the path fails. Thus to ensure that a given set of paths meet timing, they need to be tested over for different worst case test vectors.

Although ATPG based methods have been suggested for identifying the worst case vectors [42] [44], the effectiveness of such methods is limited by the accuracy of the delay models being used. A better solution would be to use statistical delay models to estimate the bounds on path delays considering various sources of variation. The timing test should then involve measuring the path slack and checking if the slack is large enough to allow the

possible variations in the path delay. This will increase test confidence and relax the need to generate tests for all possible operating conditions. Implementing such a methodology would require the capability of measuring the slack of any path during test.

In [24], a Modified Vernier Delay Line (MVDL) circuit was proposed, which can be used for measuring path delay during post silicon debug and validation. However, multiple MVDL circuits would be required in practice and connections between the MVDL circuits and the circuit flops introduces additional routing overhead. In addition to that, the MVDL circuit can only measure path delays, but does not provide control over the test clock frequency. Thus even if path delay can be measured, it cannot detect if the path would fail due to variation in clock skew. The PCG technique facilitates path delay measurement, with much lower area overhead as compared to the MVDL method. Path delay (and hence the slack) can be measured by sweeping the test clock frequency till the path fails timing. Delays of multiple paths can be measured simultaneously by grouping together paths with similar delays. The measured path slack can then be compared with the required estimated value to validate the timing of the path.

4.5 Simulation Results

The input to the PCG circuit is the TT signal and the output is the $CAPTURE$ signal. A rising edge on the TT signal (launch event) triggers a rising edge on the $CAPTURE$ signal (capture event) and the frequency of the test clock is determined by the delay between TT and $CAPTURE$. If the propagation delay of the coarse buffer is d_c and the fine buffer is d_f, then the arrival time of the $CAPTURE$ signal is given by

$$T_{cap} = d_o + m \times d_c + k \times d_f; \forall m = 1, N_c, \forall k = 0, N_f \tag{4.1}$$

where d_o is the constant offset delay, N_c is the number of coarse delay stages and N_f is the number of fine delay stages, and m, n are the values programmed in the $CaptureDelay$ register. Thus the maximum capture frequency is limited by $d_o + d_c$, the resolution of the frequency depends on the fine buffer delay d_f, while the range is determined by the number of stages. We implemented the PCG circuit with 10 coarse buffer stages and 3 fine buffer stages using 130nm technology. The coarse delay buffer delay was designed to be $d_c = 100ps$, while the fine buffer delay was $d_f = 25ps$. Since the buffers will be propagating only one type of transition (rising in our case), the buffer delays were optimized by using skewed inverters. We have designed and implemented the test control logic required for using the PCG technique in each of the three ac-scan methods. The signal waveforms obtained for Enhanced Scan, LOS scan and LOC scan are shown in Figures 4.11, 4.12 and 4.13 respectively.

The resolution of the capture clock frequency is the smallest step size between any two test clock periods that can be generated using the PCG circuit. For the PCG circuit shown here which uses a chain of delay buffers, the capture clock resolution is the delay of the fine buffer, $\triangle T_{cap} = d_f$. In practice, the achievable resolution is highly dependent on the process variability. If only interchip or chip-to-chip variation is considered, then the delays of all buffers increase or decrease together and hence, even though the actual capture periods change, the step size, $\triangle T_{cap}$, remains the same. On the other hand, for intra-chip variation, since the process parameters can vary for any two devices on a single chip, each buffer in the PCG could have a different delay, introducing uncertainty in the step size. Note that since the PCG is a very small circuit, the parameter variability across the buffer chain will not be very large due to spatial correlations. For the designed PCG circuit, Figure 4.14 shows the variation in capture clock periods at various settings due to inter-chip variation. L_{eff} was assumed to have a Gaussian variation $N(\mu_L, \sigma_L)$, with $3\sigma_L = 0.1\mu_L$ for the simulation. To

show the effect of intra-chip parameter variations on the step size of the capture clock, Monte Carlo simulations were done with intra-chip variation of $3\sigma_L = 0.05\mu_L$. Figure 4.16 shows the standard deviation of the capture clock period plotted against the nominal capture clock period, varied in coarse steps. In addition to the variability introduced by the PCG itself, there could be mismatch in the clock paths to the launch and capture flops which makes the skew variable. Note that this component of variability is also present in traditional at-speed test methods where capture clock is obtained from on-chip PLL since those methods also use the system clock network.

Thus, process variations introduce uncertainty in the actual clock arrival time, which affects the effective capture clock resolution as shown in Figure 4.15. If the designed step size (resolution) is smaller than the total variation, then in the overlap region, it is difficult to determine the actual capture delay with certainty. Even though clever circuit tricks can be used to reduce the step size $\triangle T_{cap}$ obtained from the PCG circuit, the actual resolution is limited by the delay variability on the clock. During design, it is therefore important to characterize for process variability and select the buffer sizes such that $\triangle T_{cap}$ is larger than the expected variation on the clock. Note that we can always obtain the ideal resolution by performing multiple measurements. For the capture generator circuit shown in Figure 4.1(a), the constant offset delay d_o is determined by the NMOS after the coarse buffer, the delay through the PMOS after the fine buffer, and the final buffer required for boosting the drive strength of the capture signal.

4.5.1 Delay defect detection

As mentioned in the previous section, the PCG circuit can be used to detect very small delay defects by performing faster than at-speed tests. For comparison, we synthesized

the ISCAS89 s15850 benchmark circuit using 130nm technology and extracted the longest true path through each node of the circuit. Figure 4.17 shows the size of defects that can be detected using the traditional at-speed test compared to that obtained using the PCG technique for the top 1000 paths. The at-speed test assumed a clock period of $7ns$, which allowed a slack of $460ps$ on the most critical path. Since the same clock period is used for all paths, the detectable defect size increases as the delay of the test paths decreases. When using the PCG technique, to detect the smallest defect, the capture clock period needs to be set as close as possible to the nominal delay of the test path. To make sure that any random delay increments due to process variations are not categorized as delay defects, for a path with delay distribution $N(\mu, \sigma)$, the capture clock is set such that $T_{cap} > \mu + 3\sigma$. It can be seen that while the defect coverage can suffer using traditional at-speed methods if the selected test paths have large slacks, it remains unaffected using the PCG technique.

4.5.2 Path Delay Measurement

The PCG circuit can be used to measure the delay of any path by sweeping the capture clock frequency. The number of iterations required to perform path delay measurement and the measurement resolution is highly dependent on the delay variability on the path. Delay variation is the uncertainty between the actual path delay and the expected or the nominal value. This uncertainty could be due to process parameter variation, or dynamic effects such as coupling noise, supply noise etc. The resolution of measurement is the smallest change in path delay (from its expected value) that can be detected by varying the capture frequency. In the ideal case, when there is no uncertainty, the measurement resolution should be equal to the step size of the capture clock $T_{res} = \triangle T_{cap}$ and capture needs to be done only once by setting the correct test clock period. In the presence of variability, however, if only

Table 4.1: Iterations Vs measurement resolution for s35932 path

Num. iterations	Average Resolution (ps)
1	63.1
2	33.3
4	14.5

one iteration is to be used, then the test clock period has to be set with sufficient margin, which increases the effective T_{res}. To obtain the same amount of resolution as the ideal case, multiple iterations will be required. Thus there is a trade-off between the measurement resolution (T_{res}) and the number of iterations (measurements) that need to be done. For a test path P, if the total variability in path delay $(-3\sigma, +3\sigma)$ is given by σ_P, then the resolution is given by

$$T_{res} = \begin{cases} \triangle T_{cap} & \text{if } \sigma_P < \triangle T_{cap}, \\ k \times \triangle T_{cap} & \text{if } \sigma_P >= \triangle T_{cap}. \end{cases} \tag{4.2}$$

where $k = \lceil \frac{\sigma_P}{\triangle T_{cap}} \rceil$. The measurement resolution can be improved by performing multiple iterations, varying the capture clock period till the path fails. As an example, we simulated a path from the ISCAS89 s35932 circuit along with the PCG circuit described above. The simulation was done for L_{eff} having a Gaussian distribution $N(\mu_L, \sigma_L)$ with $3\sigma_L = 0.1\mu_L$. The simulation is repeated for different settings in the PCG circuit to change the capture clock period, starting with the most conservative value. Table 4.1 lists the average resolution of path delay measurement for different numbers of iterations. It can be seen that as the number of capture iterations are increased, the path delay can be measured with better precision.

4.6 PCG Implementation on Silicon

The PCG circuit was implemented with 12 coarse buffer stages and 5 fine buffer stages using 45nm process technology. Figure 4.18 shows the schematic design of the coarse buffer and fine buffer stages. For the coarse buffer stage, we had to add an extra NMOS in the pull down path to prevent charge sharing between node Y and the intermediate nodes. The top NMOS is also controlled by the select signal coming from the scannable select register. We observed that the interconnect and parasitic capacitances affect the delay resolution that can be obtained using the PCG. Thus, it is very important to have a optimized layout for this circuit. In our implementation, we were able to obtain a coarse resolution of 125ps and fine resolution of approximately 25ps after layout. Figure 4.19(a) shows the capture periods observed at various coarse delay settings and Figure 4.19(b) shows the same for all the fine delay settings, at a fixed coarse setting.

Due to space constraints, we chose to only implement the *Enhanced Scan* scheme with a very small DUT. The DUT is not a real circuit but a set of paths with different delays. The basic objective was to be able to test the designed PCG at various delay settings and also have paths with similar delays so the PCG accuracy can be tested. Since, the library provided did not include an Enhanced Scan flop, we added an extra latch in front of each scannable flop, and the enable of these latches were tied to the SE signal. Thus the flop outputs will not change during scan operation. The DUT was constructed using a chain of inverters and ex-or gates, with 5 inputs and 5 outputs and a total of 25 paths which can be robustly tested. Thus, there is a path from each input to each output and for robust testing, we switch only one input at a time. In reality, different path delays will be observed if multiple inputs are switched together, but there will be glitches on the signal path. The delays of the 25 paths were measured from the launch point, which for Enhanced Scan occurs

when the SE is de-asserted. The delay distribution of the paths in the DUT is shown in Figure 4.20. All the simulation results shown are using post-layout extracted data. The complete layout dimensions were $72\mu \times 56\mu$, of which, the capture generator is $69\mu \times 10\mu$, the PCG select register is $72\mu \times 2\mu$ and the DUT is $72\mu \times 32\mu$.

4.7 Conclusion

The ability to perform faster than at-speed test can be invaluable for detecting small delay defects that can easily escape traditional delay test and hence cause reliability issues. One of the main challenges in providing fast test clocks is the limitation on the clock frequencies that can be provided by external ATEs. We presented a programmable on-chip capture signal generator circuit that can be used to perform both at-speed and faster than at-speed tests. The advantage of such a scheme is that the delay test frequency can be programmed as a part of the test vector itself. This provides flexibility in path selection for delay testing, since small delay defects can be detected even using shorter paths by selecting the appropriate capture period. This new technique can be very useful during post silicon validation and debug since it provides the capability of measuring the delay (or slack) of any path in the circuit. The overhead in terms of area and design effort is minimal and it can be easily incorporated in any of the current scan-based delay test methodologies.

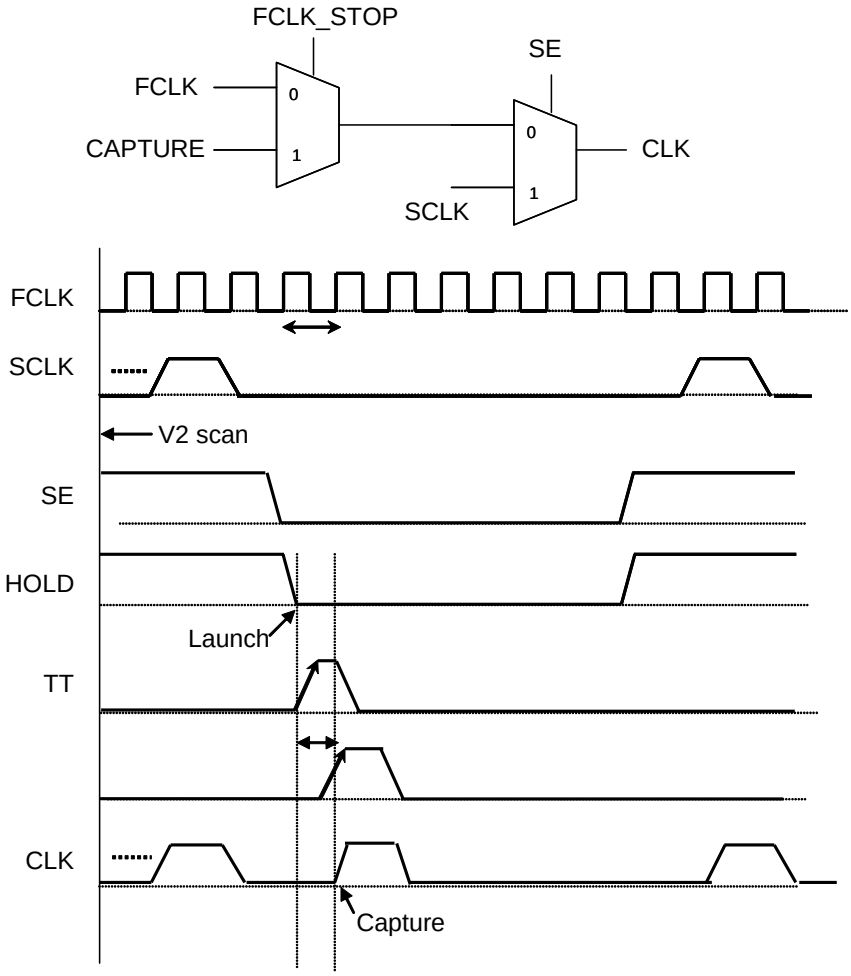


Figure 4.5: Enhanced scan

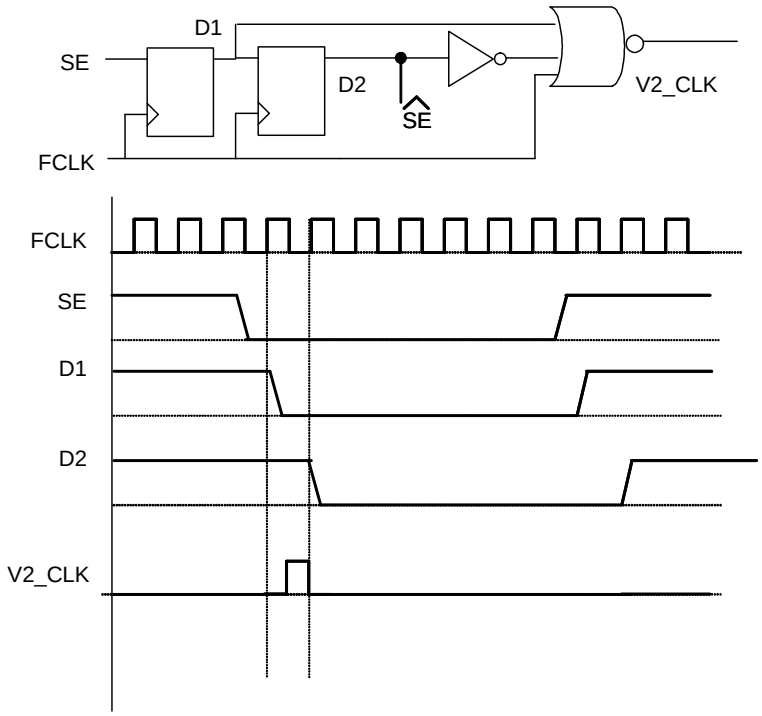


Figure 4.6: Generating $V2\_CLK$ and $\widehat{SE}$ signals

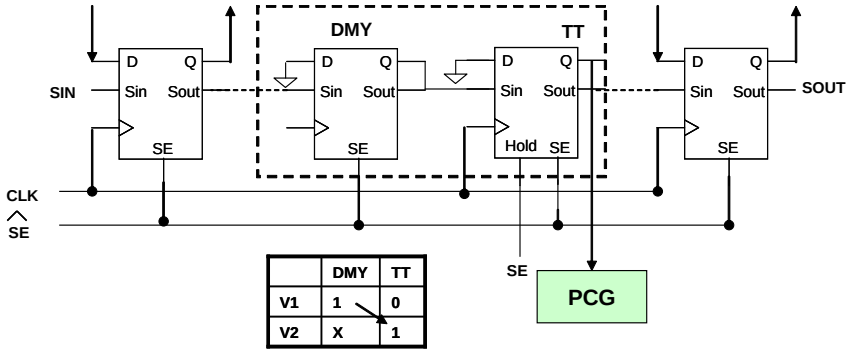


Figure 4.7: Trigger generation for LOS

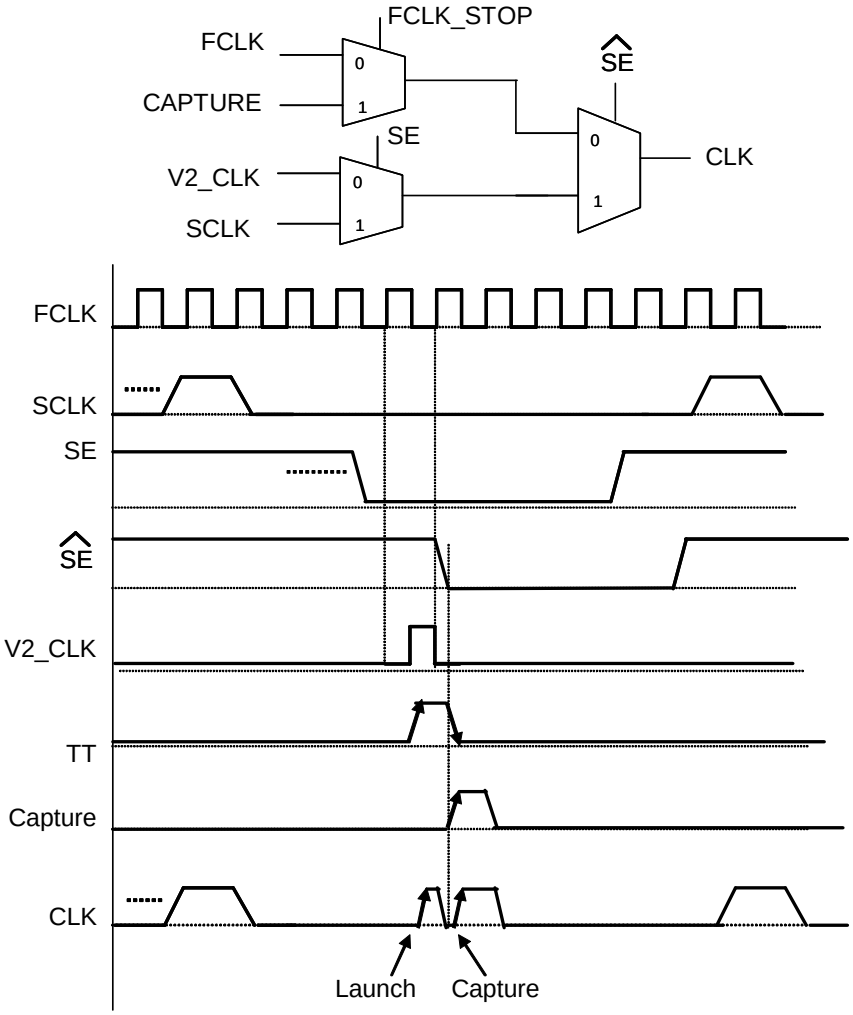


Figure 4.8: Launch on Shift

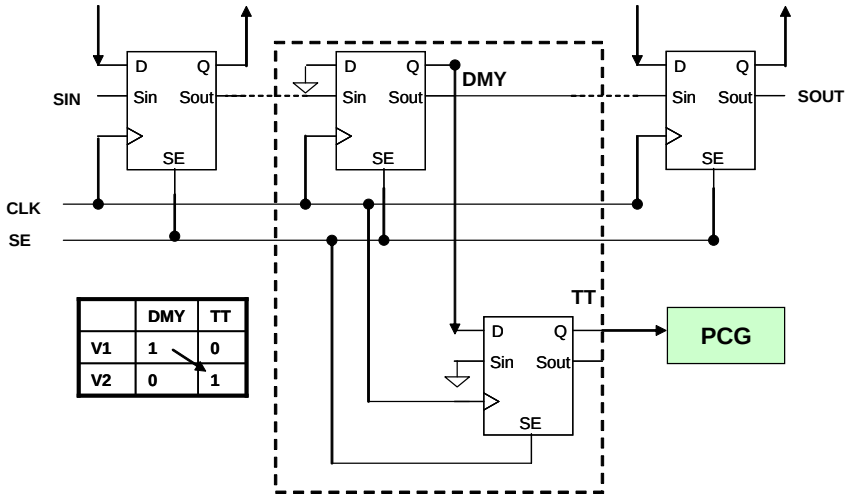


Figure 4.9: Generating Trigger for LOC

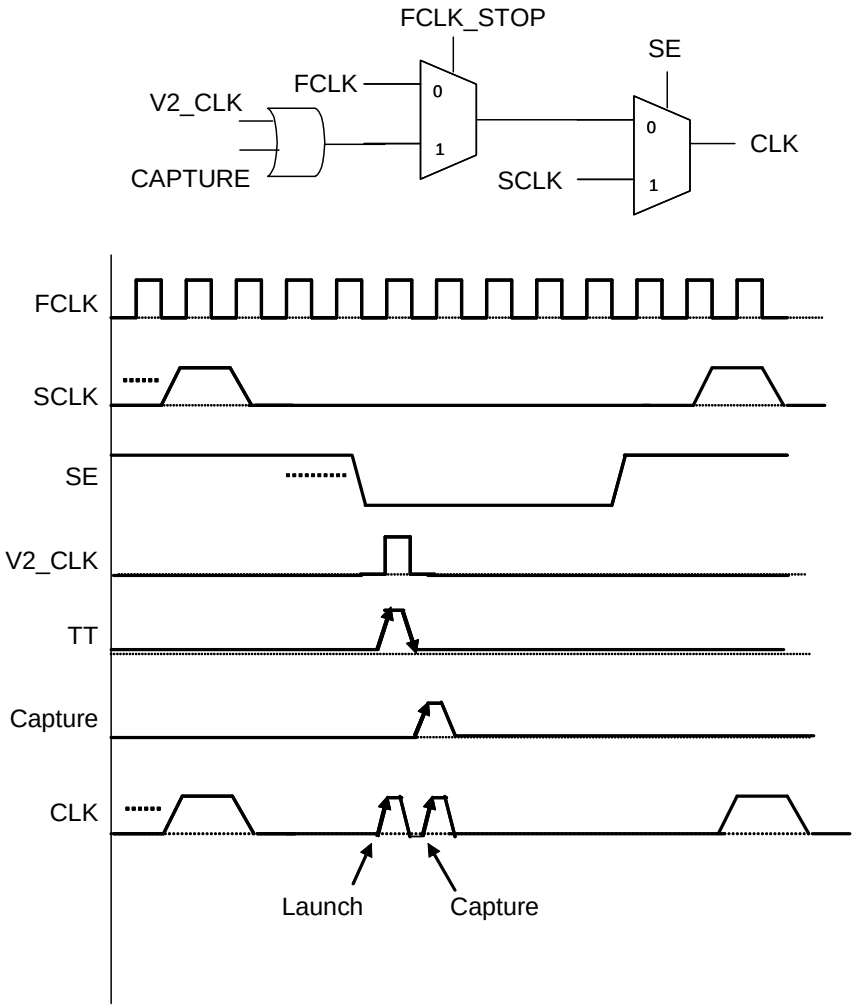


Figure 4.10: Launch on Capture

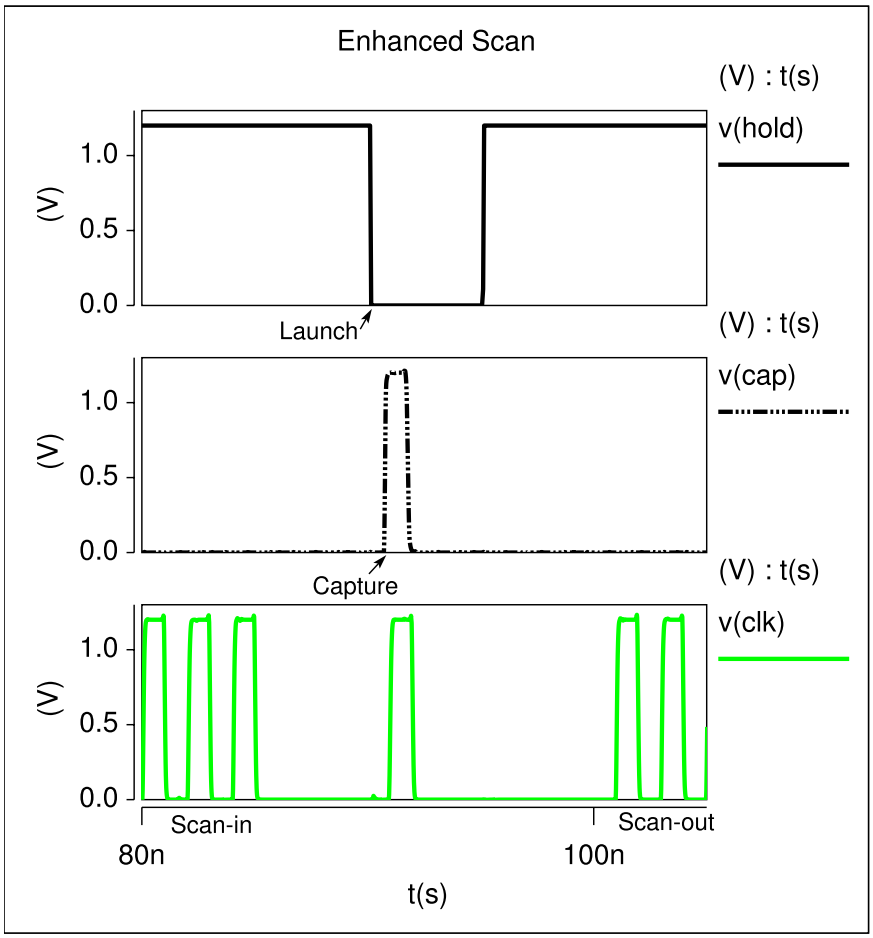


Figure 4.11: Simulated waveform for Enhanced Scan

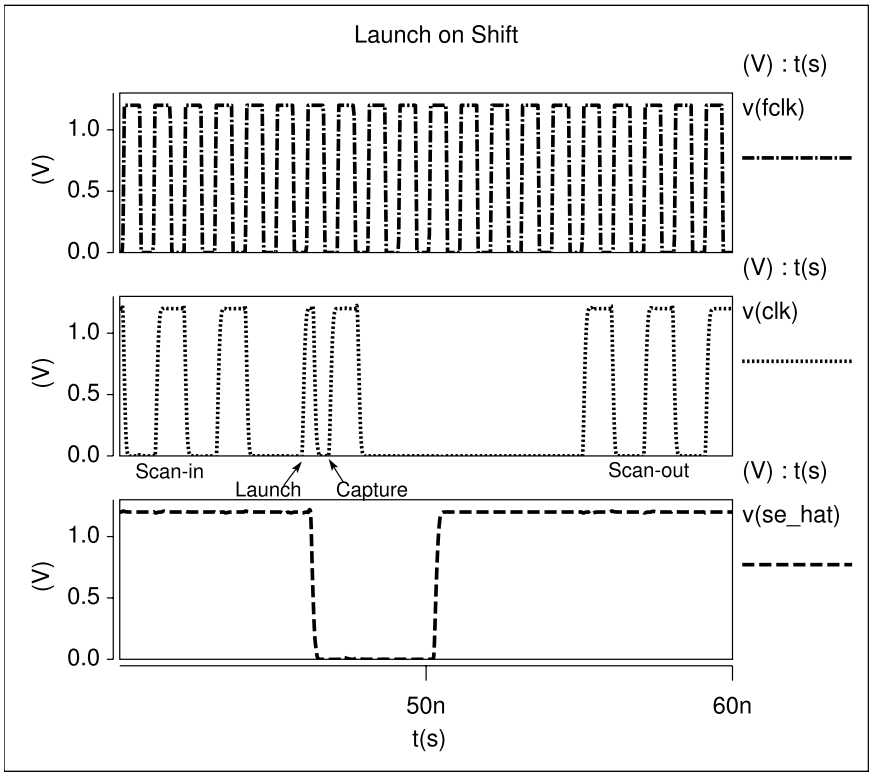


Figure 4.12: Simulated waveform for LOS

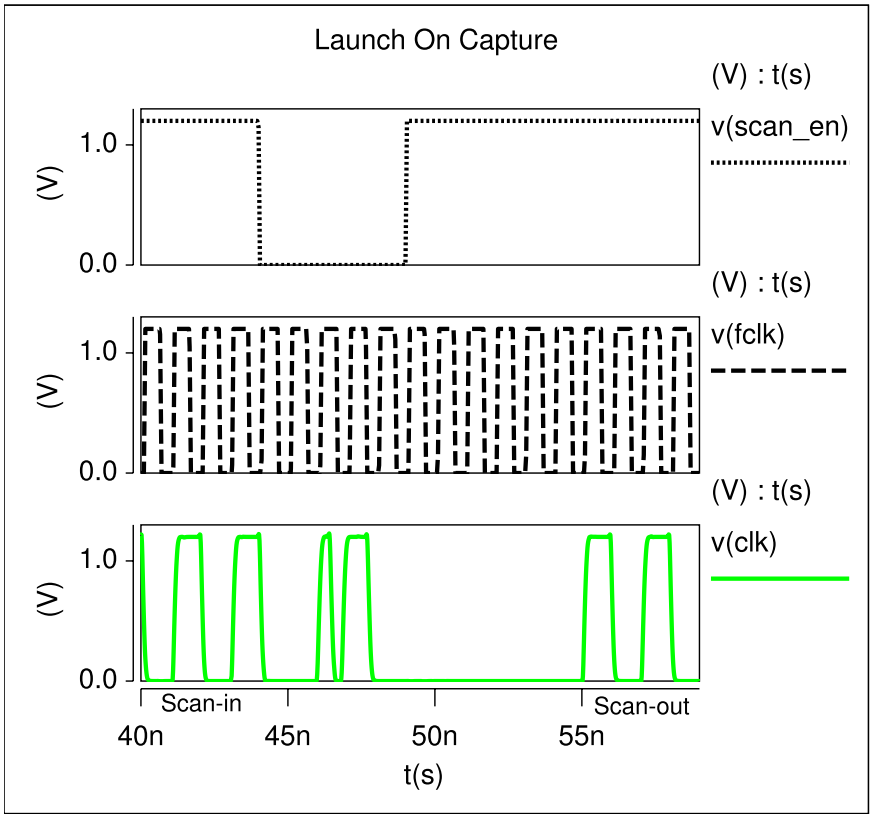


Figure 4.13: Simulated waveform for LOC

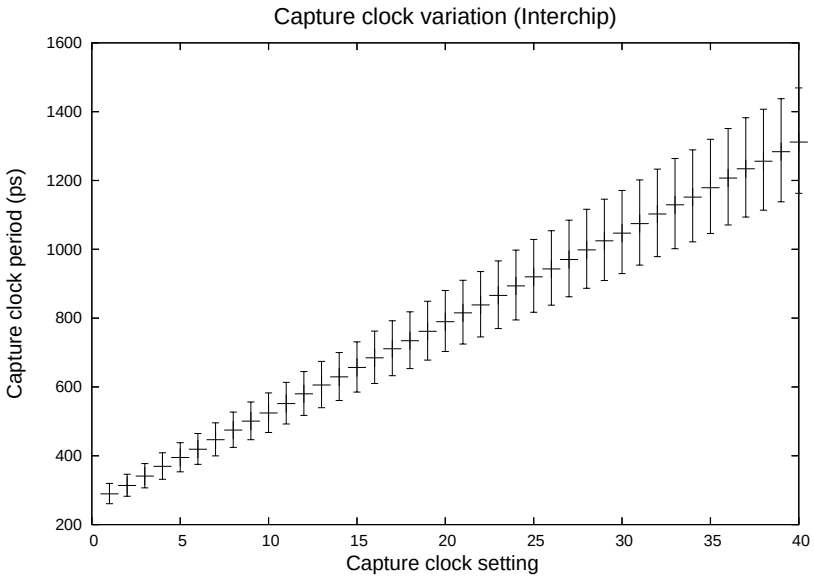


Figure 4.14: Capture clocks for inter-chip variation

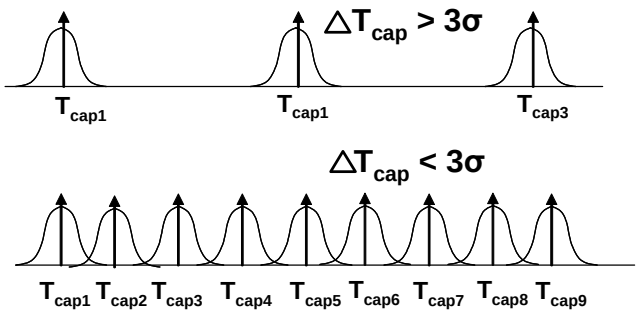


Figure 4.15: PCG resolution

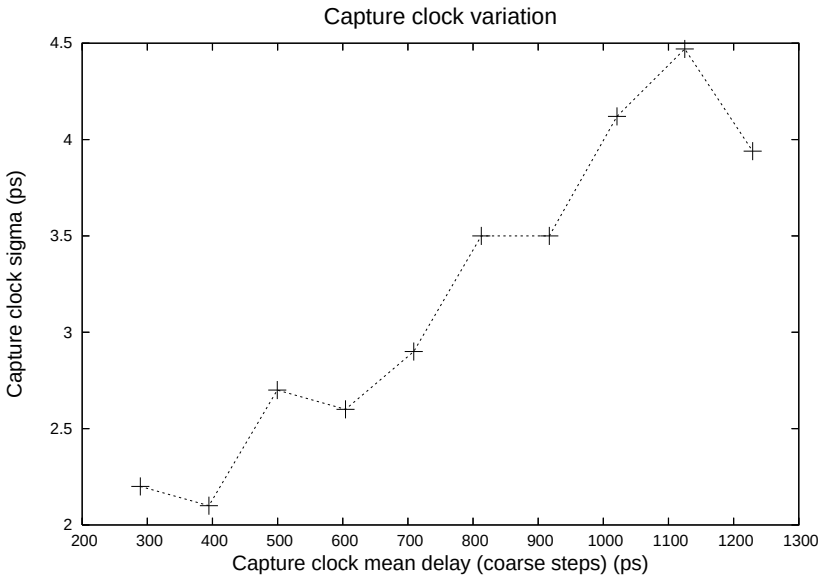


Figure 4.16: Effect of intra-chip variation on PCG

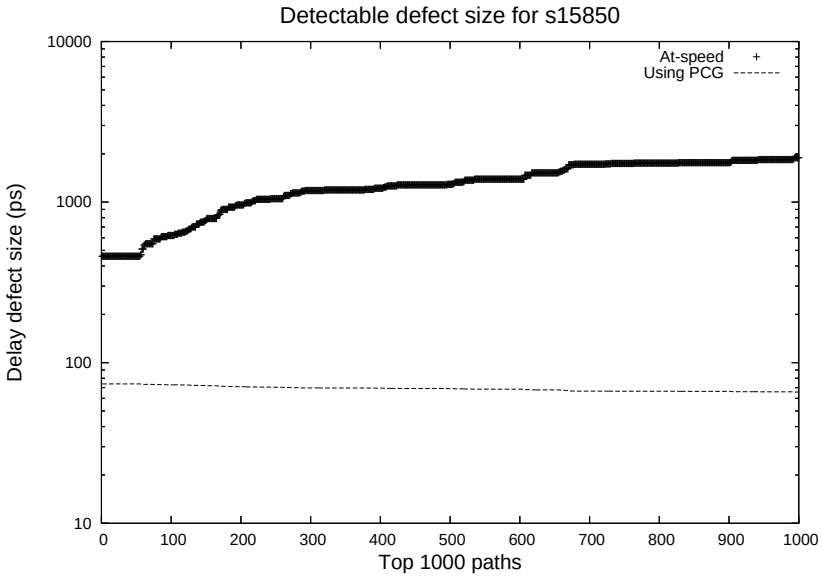
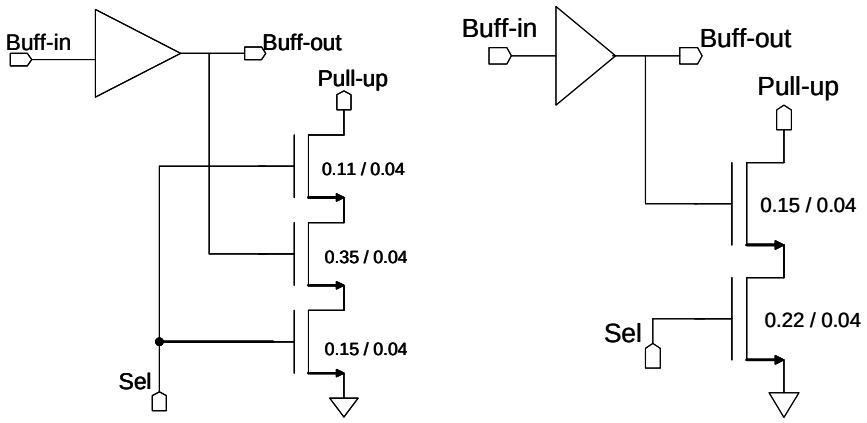
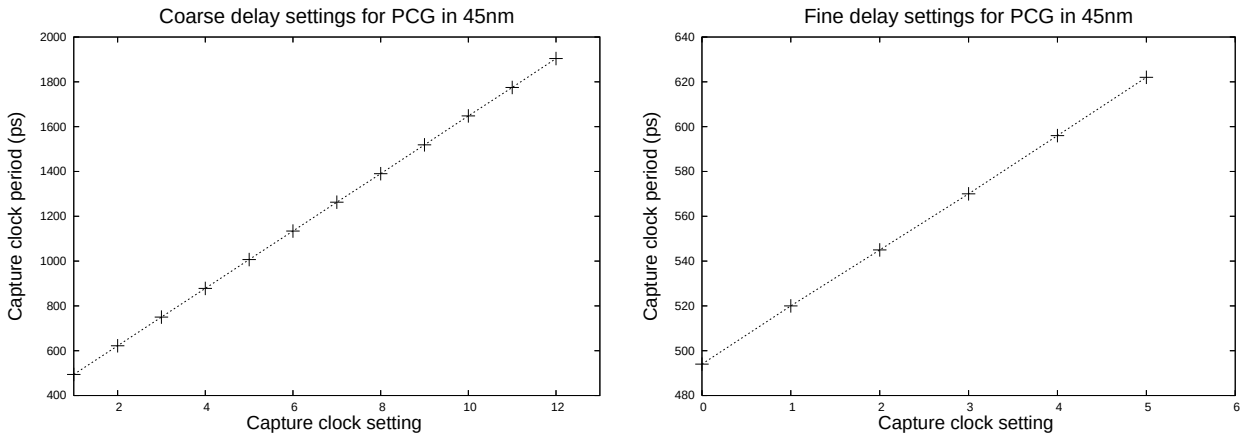


Figure 4.17: At-speed test Vs variable capture-speed test



(a) Coarse delay stage (b) Fine delay stage

Figure 4.18: Delay stages for PCG



(a) Coarse steps (b) Fine steps

Figure 4.19: Capture delay for PCG in 45nm

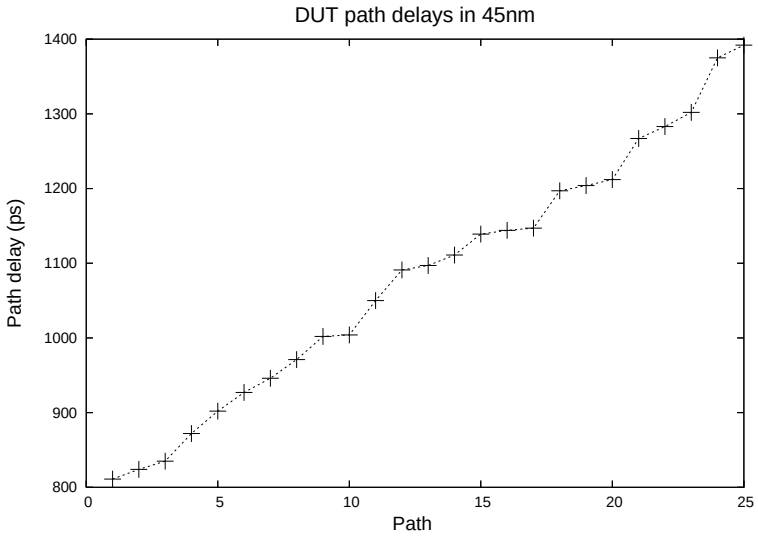


Figure 4.20: DUT path distribution

Chapter 5

Small Delay Defect Detection

5.1 Introduction

It has been observed that high-impedance interconnect defects form a large portion of the defects that escape tests [62]. Such defects are therefore very likely to cause field failure, thus increasing the DPM numbers. Resistive interconnect defects cannot be reliably detected using parametric tests such as Iddq [67], Min-Vdd tests [73] or Very Low Voltage(VLV) tests [32] and can be detected only conditionally in normal delay tests [49], [16]. The detection of resistive open defects depends on the size of the resistance, location of the defect and whether the path delay is wire dominated or device dominated [74]. The need to detect these defects is compounded by the fact that interconnect defects are becoming more prevalent in modern VLSI circuits because of the increased number of vias [61] and higher routing densities. In this research we model high-resistance interconnect defects as small-delay defects and the use delay testing as a reliability screen for detecting these defects.

The primary challenge in detecting small-delay defects is that the delay increment caused by these defects is less than the path slack, while traditional delay test methods only detect defects that are greater than the slack interval. In [79] a new delay test method to detect small-delay defects, called 'Delay Detection in Slack Interval' (DDSI) was proposed. This method suggests that multiple high frequency captures in the slack interval could be used to estimate, to a certain accuracy, how much the path delay differs from its nominal

value. The main idea of this test method is to estimate the actual path delay of a given signal path, rather than just checking if it meets a certain timing-constraint. As part of this research, we have developed circuit solutions that facilitate accurate control over the capture clock to enable path delay measurement during test, the details of which will be discussed in Chapter 4. The analysis discussed in this chapter assumes that a PCG like circuit is being used for faster than at-speed capture and hence the actual path delay of a signal path can be known during test time.

If the path delay during test exceeds an estimated value (using pre-silicon timing analysis), then it can be taken as an indication of the presence of a defect. However, due to process variations, path delays are non-deterministic and hence it is difficult to determine if the delay increment observed on a path is due to random process parameters or due to the presence of small defects. Addressing this issue is the primary focus of the research. Given a technique to estimate path delay at test time, a test approach is proposed, that would maximize the probability of detecting small delay defects caused by resistive interconnect. Section 5.2 analyzes high-impedance interconnect defects and their effect on path delay, Section 5.3 discusses the effect of process variations in detecting these defects. Section 5.4 describes the proposed method of detecting these defects by selecting minimum delay variance paths. Section 5.5 provides simulation results, followed by conclusions in Section 5.6.

5.2 Resistive Interconnect Defects

Resistive interconnect defects can be modeled as an additional resistor connected between the two circuit nodes on the signal path. The following sections discuss the various factors that affect the behavior of resistive interconnect defects such as defect size, location

and wire length.

5.2.1 Defect Size

Defects with resistance greater than $10M\Omega$ are considered strong opens and can be modeled as stuck-open faults, while defects with resistance less than $10M\Omega$ are considered weak opens. If the interconnect defect is a spot defect, then it is very difficult to detect partial opens, since the change in resistance is not significant until there is a complete open. On the other hand, if the defect is spread over the length of the interconnect, then defect resistance would be higher even for partial opens, and hence more detectable. For a wire of length 200μ, width 0.32μ and sheet resistance of 0.08 (0.18μ technology), the wire resistance is 50Ω. Under the influence of process variations, the wire resistance may vary by 10% as shown in Figure 5.1a. Thus any resistance value higher than the expected variation could mean a potential reliability hazard.

The plot in Figure 5.1b shows three cases which compare the change in wire resistance depending on the percentage of wire that was affected by the defect. The line for $L_{defect} = 50\%$, represents a case where the defect is affecting 50% of the wire-length i.e., a distributed defect, while the $L_{defect} = 5\%$ represents a spot defect. The plot shows the variation of wire resistance as a function of defect size, which is useful for modeling small-delay defects.

5.2.2 Defect Location in Path

The change in the path delay due to presence of a resistive interconnect defect also depends on the defect location. Two NAND-chain circuits of length 6 and 16 were simulated for observing the effects due to defect location. Figure 5.3 shows the path delay of the NAND-chain circuits as a function of defect size for two defect locations at the extreme ends

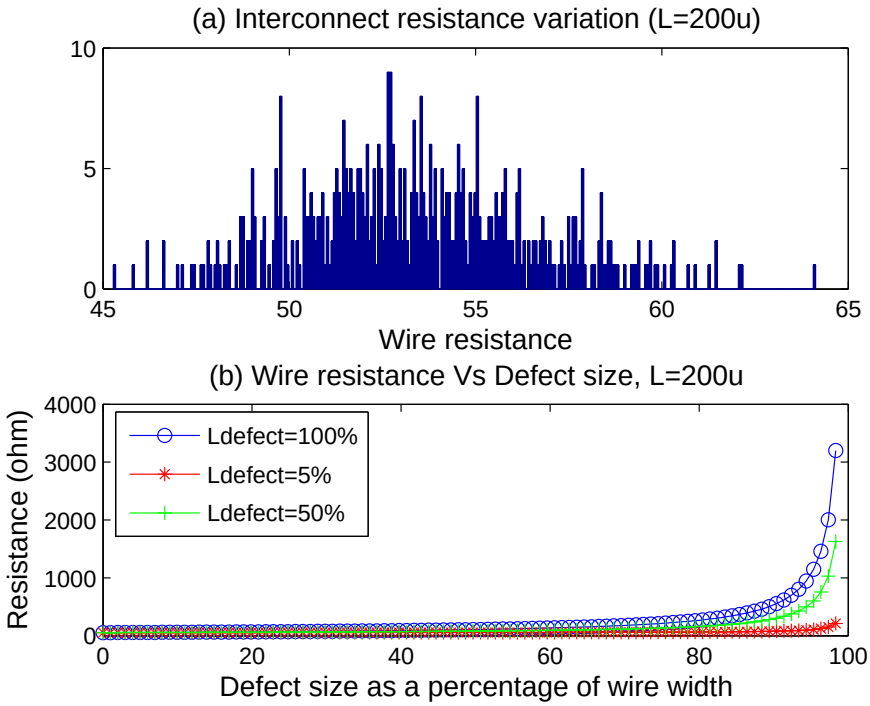


Figure 5.1: Wire resistance as a function of defect size

of the path. The NAND-chain circuit is shown in Figure 5.2, and the defect locations that were observed are shown as $n1$ and $n2$ nodes. The plot shows that defects at the sink end

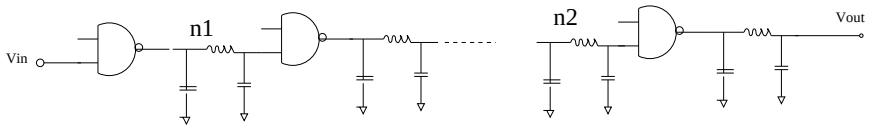


Figure 5.2: NAND-chain circuit

(node $n2$) of the path cause substantial delay change for much smaller defects as compared to those at the source end (node $n1$). Also, for a defect at the sink end, the shorter path (6 stages) has a much better delay response to a resistive open as compared with the longer path (16 stages). Thus the delay response of a path to a resistive defect depends on both the defect location and path length. Note that in the larger circuit, the delay behavior will

also be affected by the fanout load at that defect location.

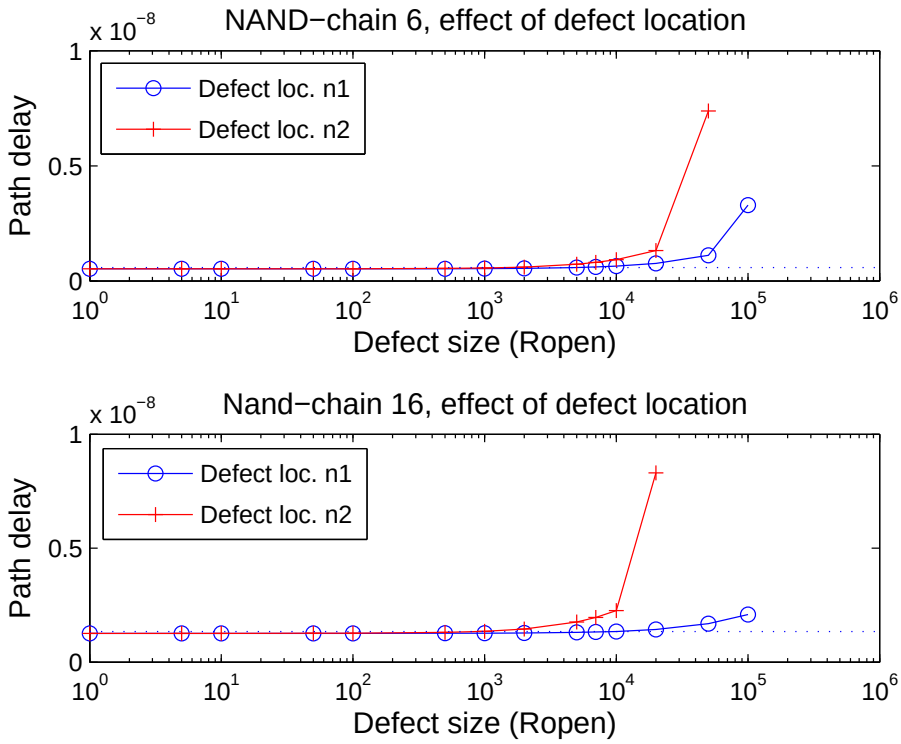


Figure 5.3: Effect of defect location

5.2.3 Wire length

For a wire delay dominated path, a small change in wire resistance would be reflected easily in the path delay, while on a path dominated by device delay, the wire resistance should be increased by a substantial amount for it to cause a detectable change in the path delay. The plot in Figure 5.4 shows the path delay as a function of defect size for two different NAND-chains, and the wire length at the defect site is varied from 50μ to 200μ. It can be seen that the knee of the path delay curve shifts to the left as the wire length increases. This means that as the wire length at the defect site increases, the path delay responds much

more to defects and hence it becomes easier to detect smaller resistive interconnect defects.

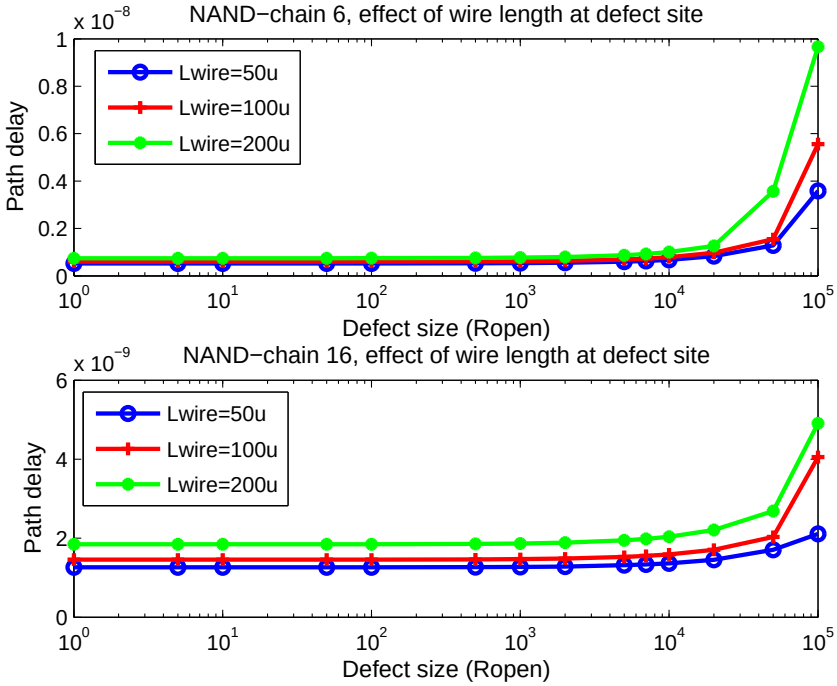


Figure 5.4: Effect of wire length at defect site

5.3 Issues Due to Process Variation

Process variations can be classified as inter-chip, where the process parameters vary from chip to chip but are constant within a single chip, and intra-chip, where the process parameters show variation within the chip itself. Thus, if a process parameter is modeled as random variable with mean μ to capture the variations within chip (intra-chip), then inter-chip variation will cause a shift in μ from chip to chip. It was shown in [79] that the effect of inter-chip variation can be canceled, if during test time the mean delay of any path is taken as the average of the measured path delays from the 'known good' neighboring dies.

Thus instead of determining the nominal or mean path delay from the static timing analysis, the mean path delay is determined from the 8 neighboring good dies, which will cancel the effect of inter-chip delay variation. In the analysis that follows, only intra-chip variation has been considered, with the assumption that inter-chip variation can be canceled using similar averaging techniques.

Figure 5.5 shows the delay distribution for the NAND-chain circuits due to L_{eff} variation, with and without resistive interconnect defects. The simulation was done for 0.18μ technology, with L_{eff} having a Gaussian distribution with $3\sigma/\mu = 10\%$. The path delay distribution is assumed to be Gaussian with mean μ and variance σ for the defect-free case. A resistive interconnect defect will shift the path delay mean to the right, while the

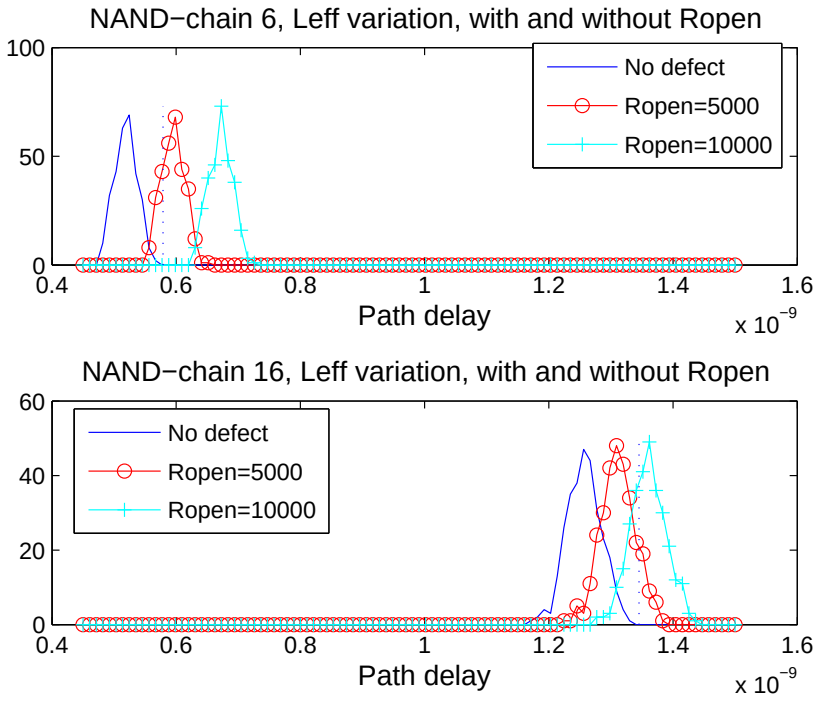


Figure 5.5: Effect of process variations

variance remains unchanged. If there is a mechanism to estimate the actual path delay and $\mu + 3\sigma$ is taken as the threshold of safe delay, then any delay increment greater than this threshold will represent a small-delay defect. In Figure 5.5, the threshold is shown using a dotted line. It can be seen that for a path with 6 stages, an interconnect defect with resistance of 10000Ω can be detected easily, but a small percentage of circuits with defect size of 5000Ω can go undetected. On the other hand, for a path with 16 stages, a large percentage of defective circuits can go undetected, since the delay distributions of the defective path overlaps with that of the non-defective one. If the measured path delay is in the overlap region, then it is difficult to tell if the delay increase is due to process variations or due to the presence of a defect. It is very important to distinguish between the two cases since the first case means the circuit is good but runs at a slow process corner while the latter means that there is a potential reliability hazard in the circuit.

Since the delay increment caused by a defect depends not only on the defect size but also on the defect location and wire length, many small-delay defects that can become reliability hazards would go undetected. The next section formulates the problem more formally and discusses the proposed solution which targets the selection of paths that would reduce the overlap region during test.

5.3.1 Path Delay Variance

In this section a theoretical analysis is provided to prove that the probability of detecting resistive interconnect defect on a line in the presence of process variations can be increased significantly by selecting the path with minimum delay variance to test the defect. A delay defect present on a path is detected if the measured path delay is greater than the threshold for maximum delay on the path, typically chosen to be $\mu + 3\sigma$ of the original path

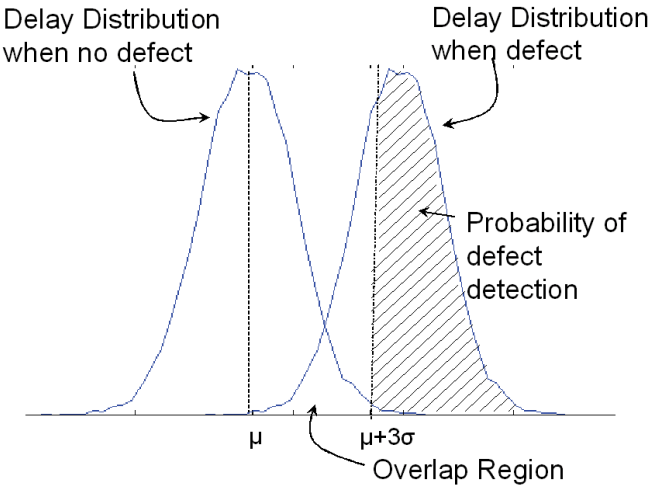


Figure 5.6: Probability of detecting a delay defect

delay. In Figure 5.6, the probability of detecting a defect is the area in the shaded region. If the path delay distribution for a defect-free case is $N(\mu, \sigma)$ and the delay increase caused due to the presence of resistive interconnect defect is d_r, then the delay distribution of the defective path becomes $N(\mu_d, \sigma_d)$. It is assumed that the distribution of the delay remains Gaussian, and this was observed from the simulations (Figure 5.5). Also, since a resistive defect will only shift the distribution by d_r, the mean of the new distribution is $\mu_d = \mu + d_r$. Thus, the probability that the defect is detected is

$$P(defect\,is\,detected) = 1/\sqrt{2\pi}\sigma_d \int_{\mu+3\sigma}^{\infty} e^{-(x-\mu_d)^2/2\sigma_d^2} dx \tag{5.1}$$

which can also be written as

$$P(defect\,is\,detected) = 1/\sqrt{2\pi}\sigma_d \int_{\mu_d+3\sigma-d_r}^{\infty} e^{-(x-\mu_d)^2/2\sigma_d^2} dx \tag{5.2}$$

This can be expressed in terms of the Cumulative Distribution Function (CDF) $F(.)$ for the path delay with defect.

$$P(defect\,is\,detected) = 1 - F(\mu_d - dr + 3\sigma) \tag{5.3}$$

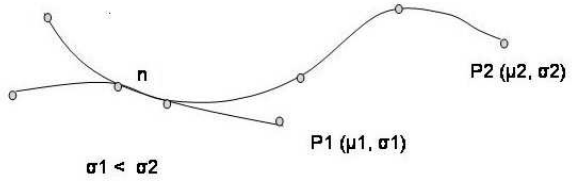


Figure 5.7: Two paths having a common net n

Since the CDF is a monotonically increasing function, the probability of detection reduces as $(\mu_d - d_r + 3\sigma)$ increases. Here d_r is the delay introduced by the resistive open and depends on the defect location, the size of the defect and the length of the defective wire. Thus for a given defect, *the probability of detecting the defect on the path is higher for a path with lower delay variance*. Consider two paths P_1 and P_2 that share a common net n as shown in Figure5.7. Path P_1 has delay distribution $N(\mu_1, \sigma_1)$ and P_2 has a delay distribution of $N(\mu_2, \sigma_2)$, with $\sigma_2 > \sigma_1$. If there is a resistive open defect R_d on n, then the probability of detecting R_d using path P_1 is higher than the probability of detecting R_d using path P_2 since the path delay variance of P_1 is lower. This effectively means that for testing delay-defects on a given net, selecting the minimum-variance path through that net will detect much smaller defects than that possible through other paths. In the next section it is shown that the paths with lower mean delay have lower variance. Hence the probability of defect detection is higher for paths with lower delay.

5.4 Proposed Approach

Selecting a minimum delay-variance path through a given net is a non-trivial problem. The path delay variance depends on various factors including parameter variations, spatial correlations, parameter correlations and die locations. The delay of any given path (PD) is the sum of the propagation delays of each of the logic-stages and the interconnect delay.

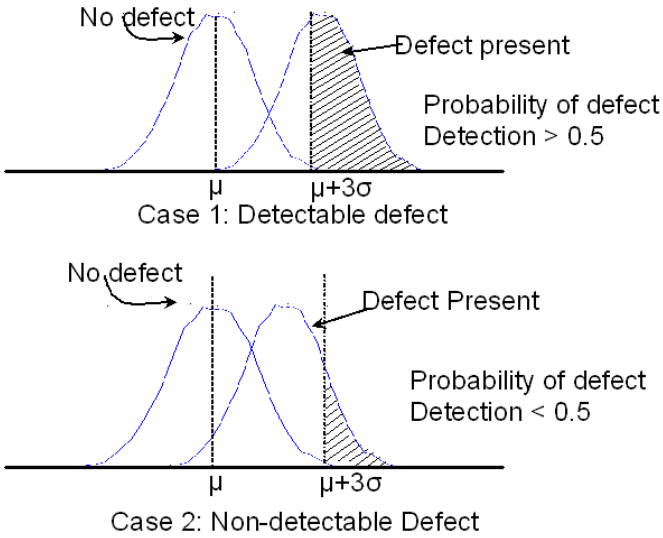


Figure 5.8: Detectable defect is one that has at least 0.5 probability of detection

Here only robust vectors are used to test the paths, hence not considering re-convergent fanouts, and the path delay can be expressed as a simple summation.

$$PD = \sum_{i=1}^{N} \tau_{p_i} + \sum_{i=1}^{N} \tau_{w_i} \tag{5.4}$$

Here τ_{p_i} is the delay of stage i and τ_{w_i} is the wire delay at stage i. For simplicity, all stages can be assumed to have equal delay, which gives

$$PD = N\tau_p + \sum_{i=1}^{N} \tau_{w_i} \tag{5.5}$$

Assuming device variations dominate the wire-width variation, the absolute variance of the path delay increases as the number of stages N, on the path increases,

$$E(PD^2) = N^2 E({\tau_p}^2) \tag{5.6}$$

Thus the minimum variance path through a net can be approximated as the shortest path

through the net.To quantify the defect-sizes that can be detected using the proposed method, the following definitions are made.

Detectable Defect: *A detectable defect is defined as a resistive interconnect defect that has at least a* 50% *probability of detection.*

Two cases are shown in Figure 5.8; the defect in case 1 is detectable since the probability of detection is 50%, while case 2 shows a non-detectable defect which has $< 50\%$ probability of detection. All defects that have $\geq 50\%$ probability of detection will also be classified as detectable.

$\mathbf{R_{dmin}}$: *R_{dmin} for a net n and a path P through n, is defined as the smallest detectable-defect on n that can be detected using path P.*

This means that any defect of resistance higher than or equal to R_{dmin} will have a probability of detection 0.5 or more and hence will be classified as detectable. From equation (4), the point at which the probability of detection is 0.5 is given by

$$P(defect\,is\,detected) = 0.5 \tag{5.7}$$

$$\Rightarrow F(\mu_d + 3\sigma - d_r) = 0.5 \tag{5.8}$$

$$\Rightarrow F(\mu + 3\sigma) = 0.5 \tag{5.9}$$

Note that the $(\mu + 3\sigma)$ point is the threshold for defect detection. Thus R_{dmin} is the value of the resistive defect that will change the delay distribution such that the median (in case of Gaussian also the mean) is equal to the threshold. A path with a defect of size R_{dmin} will shift the path delay distribution such that the new mean will be equal to the threshold used for defect detection. The R_{dmin} value for a path through a given net can thus be easily determined by sweeping the resistance value of the defect and finding the value at which the

path delay exceeds the threshold. Figure 5.9 shows that the path variance and the R_{dmin} value increases as the number of stages on path increase.

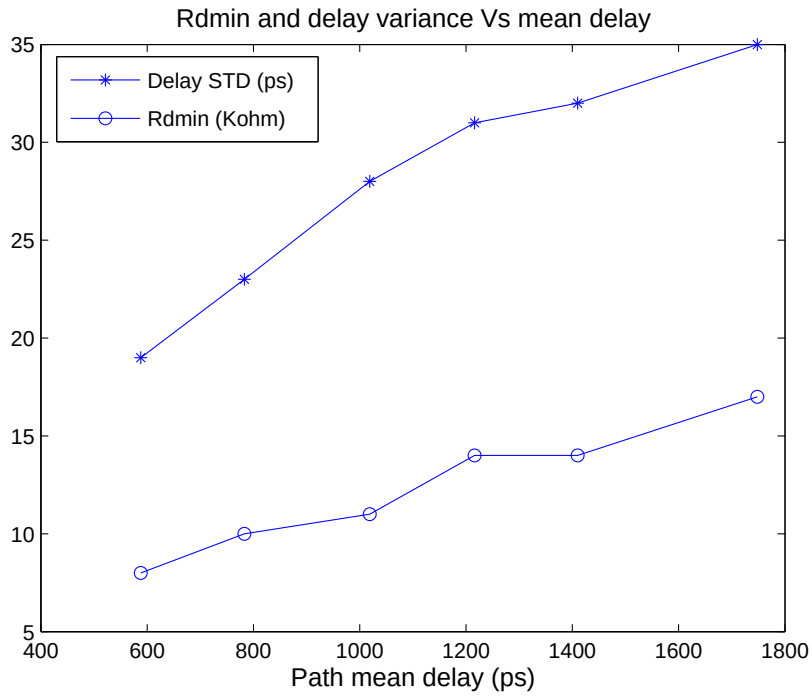


Figure 5.9: Path delay standard deviation and R_{dmin} variation with mean delay

Every net will have different R_{dmin} values for different paths that are used for testing it. In order to maximize the set of detectable defects for any given net, it would be required to select the path that has the smallest R_{dmin}. We suggest a heuristic algorithm to select paths that can minimize R_{dmin} for a given net. It was proved in the previous section that using the minimum-variance path through a net maximizes the probability of detection for a defect on that net. Also, from Figure 5.4, it can be seen that paths that have high wire-delay to device delay ratio show better delay response to small defects. From the above two observations and also using the effect of defect location, the proposed procedure to select

the path with minimum-R_{dmin} through a net is as follows.

1. For a given defect site (net), select all paths that have the smallest number of logic stages to minimize the path delay variance. Since smaller paths will require higher test frequencies, the selection process can be limited by the maximum allowed test frequency.

2. From the selected paths, select the path with the highest wire-delay to device-delay ratio. Since the net under consideration is common to all the selected paths, this would be the minimum-delay path through that net.

3. If multiple paths with minimum delay are obtained, then select the path for which the defect site is closer to the sink node.

Using the above procedure, the minimum delay path through a net will be selected as the minimum R_{dmin} path for that net. Path delay variation is caused by various factors including parameter variation, off-path input delays, coupling effects of neighboring line switching, etc. Intuitively, shorter paths should be less susceptible to both parameter variations and dynamic variations. The delay distributions of longest and shortest paths through a net in the $s1488$ benchmark circuit with and without resistive interconnect defects is shown in Figure 5.10. It can be seen that a resistive open defect of 10000Ω has much higher probability of detection using the proposed minimum R_{dmin} path compared with the longest path through the net.

One of the important challenges in delay testing is finding the true critical paths for test. Finding the structural longest or shortest path in a circuit can be done using graph search algorithms, but the complexity is in finding robustly testable paths. The most simple but time consuming method is to enumerate the structurally longest paths and then try to

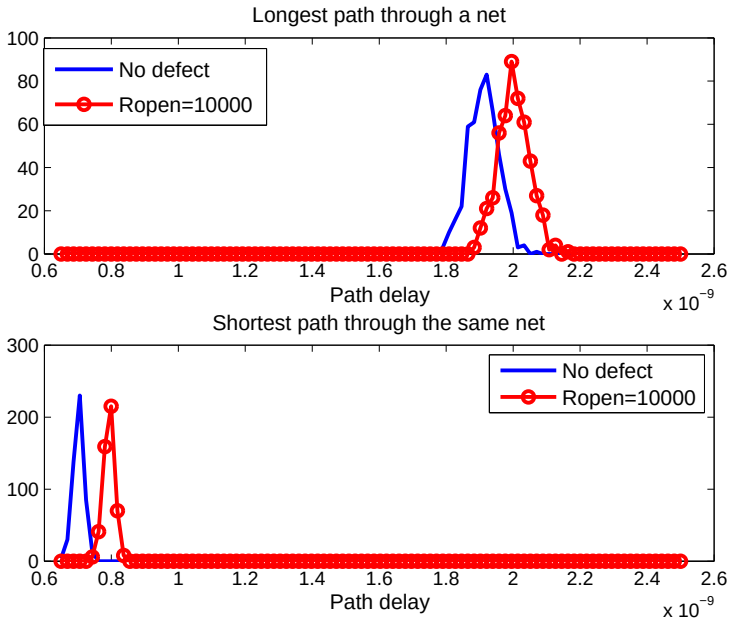


Figure 5.10: The delay distributions of long and short paths in the s1488 circuit

sensitize them. More intelligent methods, which perform directed search while pruning false paths simultaneously have also been proposed in the past [13]. Any of the currently known methods for longest path selection can be used for shortest path selection also, by simply changing the search criteria from min-delay to max-delay. In [55] a method to find a subset of paths covering all lines in the circuit at least once with the longest and shortest path through each line was proposed and it was shown that the coverage for the shortest paths is much higher than the longest paths. We use an algorithm similar to [56] to generate the shortest path through each net. It is an iterative algorithm in which it lists the K^{th} structurally longest (shortest) paths through each net in the K^{th} iteration. The algorithm was built on top of a commercial Static Timing Analysis (STA) tool and the flowchart of the algorithm is shown in Figure 5.11. In each iteration, starting with the longest(shortest) path, if the path is true then nets on that path are marked as covered. The net coverage at the end of K

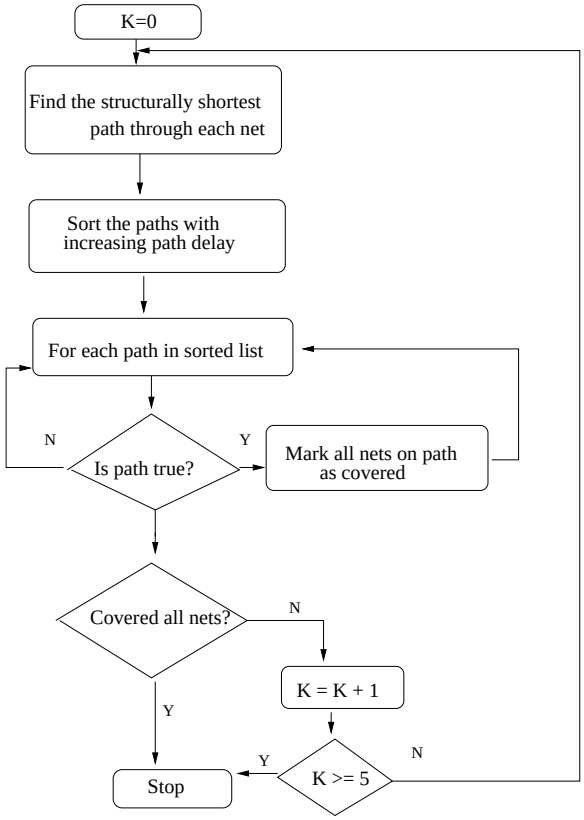


Figure 5.11: Algorithm for shortest path selection

iterations represents the percentage of total nets for which the longest(shortest) testable path through that net was found. The net coverage (NC) for a few ISCAS benchmark circuits when the algorithm was run for 5 iterations is shown in Table 5.1. It can be seen that the net coverage for shortest paths is much better than longest paths. Also the run time was observed to be much lower for shortest path selection. This difference can be explained by the fact that when a search is done for the K^{th} longest path, it needs to compare if the path is already covered. The longer the path, the longer the comparison effort, and justifying long paths also requires more backtracks than small paths.

Another observation is that a lot of structural long paths are false, which is not the

Table 5.1: Net coverage using longest and shortest paths (5 iterations)

ISCAS85 circuit	Longest path coverage	Shortest path coverage
c432	25.3	94.2
c2670	83.7	91.8
c1908	63.5	84.8
c1355	0	75.4
c3540	38.0	70.29

Table 5.2: Percentage True Paths (PTP) and Net Coverage(NC) using 5000 longest and shortest paths

ISCAS85	Longest Paths		Shortest Paths	
circuit	PTP	NC	PTP	NC
c432	0.1	12.4	29.1	94.7
c2670	10.5	13.4	63.3	75.2
c1908	0	0	55.1	78.2
c1355	0	0	9.8	64.1
c3540	0	0	32.0	56.2

case with short paths. To confirm this, the K structurally longest and shortest paths for the same benchmark circuits were generated. The result of this experiment is shown in Table 5.2. The ratio of the number of true paths found to the total number of paths generated, denoted by Percentage True Paths (PTP), was computed along with the net coverage. It can be seen that the number of true paths found is very low for long paths. For a few circuits it can be seen that the top 5000 paths were all false and hence the coverage was 0. From the above experiments, it is clear that shortest path selection requires less computational effort than longest path selection.

5.5 Simulation Results

Simulation results on some of the larger ISCAS89 benchmark circuits are shown in Table 5.3. The results compare the smallest defect-size on a net, that can be detected using the longest path versus the proposed minimum R_{dmin} path through the net. For each circuit, the longest true path was extracted and a subset of nets were selected as defect sites. For each of the selected nets, the minimum R_{dmin} testable path through it was obtained using the heuristic method proposed above. Monte-Carlo simulations were done on each of these extracted paths to determine the variance of the path delay. The variance could also be estimated using a statistical timing analysis tool. To introduce process variations, the L_{eff} was varied using a Gaussian distribution with $3\sigma/mean$ of 10%. Each of the selected nets has a different R_{dmin} value and the table lists the average value over all the selected nets for a particular circuit. For each path, the R_{dmin} value was computed by sweeping the interconnect defect resistance and the resistance value at which the path delay crosses the 3σ deviation is taken as the R_{dmin} for that path. The results in Table 5.3 show that much smaller resistive interconnect defects can be detected when the minimum R_{dmin} path through a net is used instead of the longest path.

Table 5.4 compares the smallest delay defect that can be detected using the minimum R_{dmin} path with that using the longest path, when the delay measurement resolution is $50ps$. The values have been averaged over all the selected nets. If small-delay defect detecion is to be used as a pre-burn-in reliability screen, then it is very important to detect as small defects as possible. Since using the minimum R_{dmin} path detects much smaller defects, test escapes are reduced and there is increased confidence in bining a die as reliable.

Table 5.3: Average R_{dmin} for min-delay path vs max-delay path

ISCAS89 circuit	Longest path	Min-R_{dmin} Path
S1488	8000 ohm	3000 ohm
S38584	10000 ohm	4000 ohm
S35932	6500 ohm	3500 ohm
S38417	7500 ohm	3000 ohm
S15850	8000 ohm	3500 ohm

Table 5.4: Minimum size delay defect detectable using $T_{res} = 50ps$

ISCAS89 circuit	Longest path	Min-R_{dmin} Path
S1488	139ps	71ps
S38584	172ps	66ps
S35932	102ps	56ps
S38417	142ps	50ps
S15850	114ps	57ps

5.6 Conclusion

We have proposed a test approach that increases the set of detectable resistive interconnect defects in the presence of process variations. Resistive interconnect defects can be seen as small-delay or latent defects that can easily escape conventional test methods and hence are reliability hazards. This research addresses the problem of uncertainty in determining if the delay increment in a path is due to process variations or a resistive interconnect defect. The primary contribution of this research is to prove that in the presence of process variations, the probability of detecting a delay defect on a net is higher using a path with lower delay variance. It is shown that if a technique to measure the actual path delay is used during test, then a much larger set of defects is detected using the shortest testable path through a net as compared to the longest testable path. The set of detectable delay defects

through a net can be maximized when the smallest detectable (R_{dmin}) resistive open value through the net is minimized. A path selection procedure to find the minimum-R_{dmin} path through a net has been proposed. Advantages of selecting short paths in terms of higher net coverage and lower computational complexity are also discussed. The proposed approach increases the defect detection probability of a large percentage of defects that would otherwise escape detection due the uncertainty caused by process variations.

Chapter 6

Path Selection for Small Delay Defects

6.1 Introduction

Selecting the optimal set of paths is one of the most important and challenging problems in delay testing [22], [51]. In the worst case, the total number of paths in a circuit can be a exponential function of the total number nodes in the circuit. A large number of paths, however, are either non-critical or false. Significant amount of research has been focused towards efficient path selection techniques. In [14] and [72], efficient techniques were described to select globally critical paths with delay above a predetermined threshold. In [22], a criticality model considering delay variability due to uncertainty in process parameters was developed and a corresponding path selection algorithm was discussed. In [65], an efficient algorithm to select the longest testable path through each node was described.

Most of the above techniques, however, are targeted towards at-speed testing. For defect based testing, especially for small delay defects, faster than at-speed capture speeds are required [79]. In [4], a test generation technique for small delay defects using commercial ATPG and static timing tools was described. The basic idea was to groups paths between any end points (scan flops) based on their delays. The authors further suggest that for nodes which are not covered by long paths, higher frequency clocks can be used to improve defect coverage. Thus, even though several research papers have emphasized the need for faster than at-speed testing for small delay defect detection, little work has been done in optimizing

the path selection algorithms for the same.

In Chapter 4, we discussed the PCG scheme that allows accurate control over the capture clock. This facilitates faster than at-speed capture during test, and can be used to measure the delay of any path during test by performing multiple captures as shown in Figure 6.1. In Chapter 5, we analyzed the effect of process variations on defect detection probability.

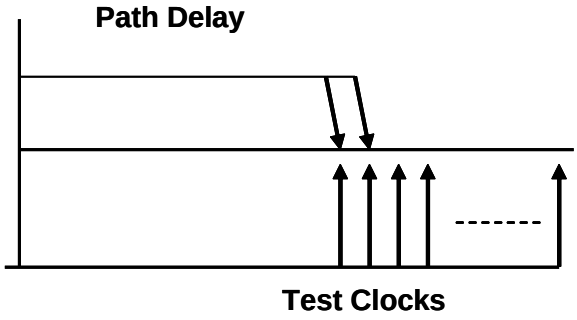


Figure 6.1: Multiple capture in slack interval

We have shown that if the delay of a path can be measured during test, the defect detection probability at any node can be maximized by selecting the minimum variance path through it. We further show that since path delay variance (absolute) increases with path length, shorter paths have higher defect detection probability than a longer path for a given defect size.

If the PCG scheme is to be employed for small-delay defect detection in high volume manufacturing, then the test time needs to be taken into consideration. Each test path could have a different delay and hence will require a different PCG setting to be scanned in as a part of the test vector. This can drastically increase the test time. A better solution would be to fix a set of test clock frequencies and then select paths that can be tested using one of the fixed test frequencies while maximizing the defect detection probability. The test clock

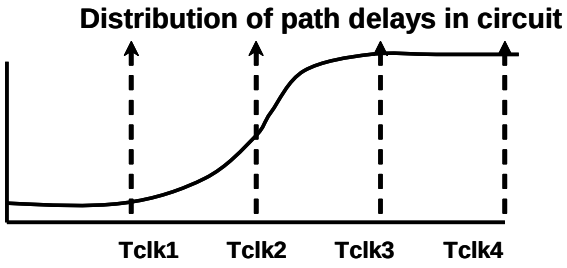


Figure 6.2: Multiple test clock frequencies

selection should be based on the path delay distribution of the circuit being tested. If all the paths in the test circuit are sorted by their delays, then the test frequencies can be selected such that paths are evenly distributed in each range as shown in Figure 6.2. Let the system clock period be T_{sclk} and the set of test clocks available be $[T_{clk_1}, T_{clk_2} \cdots T_{clk_n}]$, and where $\forall i$, $T_{clk_i} \leq T_{sclk}$. From Chapter 5, we know that R_{dmin} for a net n and a path P through n, is the smallest detectable-defect on n that can be detected using path P. In the following, we discuss a new path selection algorithm that selects the path that gives the smallest R_{dmin} for every node in the circuit, given a set of test clock frequencies. The algorithm can be described in three parts 1) preprocessing, 2) the main loop which iterates over each target test clock, and 3) the inner loop to find the best R_{dmin} path for a given node for a given test clock.

6.2 Preprocessing

The circuit is abstracted to a *Directed Acyclic Graph* (DAG) where each gate pin is a node and pin-to-pin connection is an edge. Two global nodes called *Source* and *Sink* are added, where the *Source* has edges to all the circuit inputs while the *Sink* has edges from all the outputs. Initially, a breadth-first traversal of the graph is done, to determine

the minimum and maximum signal Arrival Times (AT) at each node. This is similar to the block-based static timing analysis process. We specify the Timing Window (TW) within which a signal can arrive at any node n as

$$TW_{src,n} = [LB_{src,n}, UB_{src,n}] \tag{6.1}$$

where $LB_{src,n}$ is the lower bound on delay from *Source* node to node n and $UB_{src,n}$ is the upper bound on the same, Similarly, breadth-first traversal of the graph starting from the *Sink* node is done to estimate the delay bounds on the paths from any node to the *Sink* node. The *Sink* delay Timing Window at any node n will be given by

$$TW_{n,snk} = [LB_{n,snk}, UB_{n,snk}] \tag{6.2}$$

Thus for any node n, the delay bounds of any path that contains n, can be obtained by summing the source and sink timing windows.

$$TW_n = [LB_n, UB_n] \text{ where } LB_n = (LB_{src,n}+LB_{n,snk}) \text{ and } UB_n = (UB_{src,n}+UB_{n,snk}) \tag{6.3}$$

Next, the static implications at each node are derived using the algorithm given in [82], which generates the forward, backward and extended backward implications. Static implications are then used to eliminate false paths during path selection.

6.3 Main Loop

The objective is to find the best R_{dmin} path for each node in the circuit. Each node is assigned a R_{dmin} value, which denotes the smallest delay defect size that can be detected at that node using the currently known best path. Initially since no paths are known, the R_{dmin} value for all nodes is set to ∞. The R_{dmin} value is then updated every time a better

path is found. Any given node n could have testable paths with a range of delay values. If delay testing is done at a single test clock frequency, then the path that has the minimum slack at this frequency is the best path, as it minimizes the R_{dmin} value at n. Thus for traditional at-speed testing, the longest path through each node is the best test path. In our case, however, we have multiple test clocks and for each test frequency, a different path could be the best R_{dmin} path. Thus, to find the best R_{dmin} path at any node, we need to iterate over each of these test frequencies. From the analysis in Chapter 5, we know that shorter paths have smaller delay variance and hence smaller R_{dmin}. To give preference to short paths, we start with the highest test frequency (shortest clock period) and successively iterate over the available set of test clocks in decreasing order of frequency.

Since each test clock is being processed successively, when searching for the best paths for test clock T_{clk_i}, the paths with delays below $T_{clk_{i-1}}$ need not be considered since those paths would have been processed earlier. Thus, given a set of target test clock periods $\{T_{clk_i}; i = 1, \cdots n\}$, we define a sequence of Timing Intervals (TI), such that

$$TI_i = \begin{cases} [0, T_{clk_i}] & i = 1, \\ [T_{clk_{i-1}}, T_{clk_i}] & i > 1 \end{cases} \tag{6.4}$$

The timing intervals, help divide the entire system clock period into different bins, and circuit paths can be grouped into respective bins based on their delays. The main loop iterates over each target timing interval $TI_i = [LB_{TI_i}, UB_{TI_i}]$, and in iteration i, processes only the paths that belong to TI_i. For each TI, a new global path store is created that contains all the partial and complete paths that are discovered during the path search process.

In each iteration of the main loop, only the nodes that are not previously covered need to be considered. However, the criteria for marking a node as *Covered* is different in our case. Ideally, a node is *Covered* only when the search has found the best R_{dmin} path through

the node. In the example shown in Figure 6.3, node n has six testable paths through it. The example also shows four different clock frequencies that are being used for delay testing. It is clear that path $P3$ is the best path, since it has the smallest possible slack (10ps) with respect to one of the test clocks (700ps). Since there are multiple test clocks, it is possible

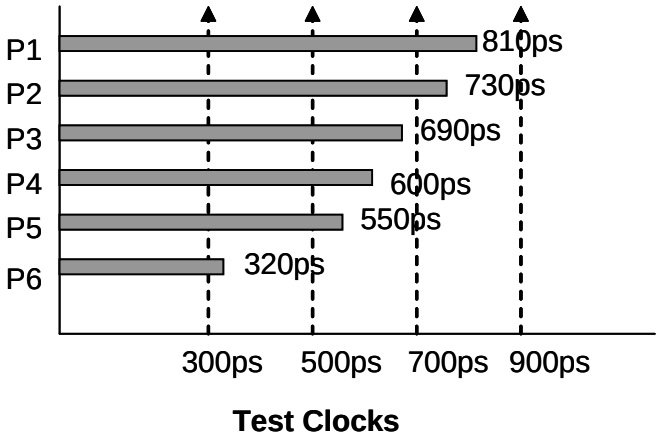


Figure 6.3: Path with minimum R_{dmin} (best test path)

that even though a testable path is found through a node in timing interval TI_i, a path with lower R_{dmin} is present in timing interval TI_{i+1}. Thus to identify the best path, each node should be processed in each TI. It is, however, inefficient to keep searching for better paths for each node in each timing interval. We therefore use a threshold R_{dmin} value to determine the stopping criteria. For each node, we first identify the best test path in the current target TI. Consider a path with delay distribution (μ, σ), and let the test clock period for which this path is selected be T_{clk}. If the slack on the path when tested with T_{clk} is less that the total delay variation on the path, then it is possible that the path would fail test even in the absence of a real defect. Thus, to minimize the probability that the path will fail the delay test due to process variations, instead of a defect, the path is considered *acceptable* only if $(\mu + 3 \times \sigma) < T_{clk}$. A node is considered *Covered* if an *Acceptable* path has already been

found for it and the R_{dmin} value is less than the predefined threshold. The following section describes the inner loop that selects the best path for any node in a given target TI.

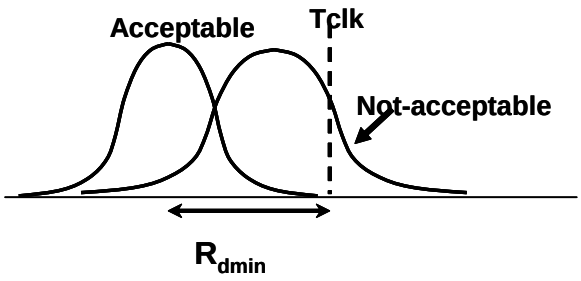


Figure 6.4: Acceptable test path

6.4 Finding the Best Path for a Node in a TI

Finding the least slack path through a target node requires traversing the entire graph in breadth-first order from the *Source* to the *Sink* node. At each node visited, new paths are created by extending the incoming partial paths with the outgoing edges. This needs to be repeated for each uncovered node. Since many partial paths will be common between various nodes, creating and propagating the same paths for each node is very inefficient. Instead we maintain a path store which contains all the partial and complete paths that are created during graph traversal. A new path store is created for each target TI and only paths that are relevant to the TI are added to store. Initially, the path store will contain a single partial path made of only the *Source* node.

We define the timing window of a partial path as the possible delays it can have to reach the *Sink*. For a partial path P_k from *Source* to m and nominal delay PD_k, the TW

is given by

$$TW_{P_k} = [LB_{P_k}, UB_{P_k}] \text{ where } LB_{P_k} = (PD_k + LB_{m,snk}) \text{ and } UB_{P_k} = (PD_k + UB_{m,snk}) \tag{6.5}$$

A path is a *Candidate Path* for node n, if

- The TW of the path overlaps with the TW of node n. This requires

$$(LB_{P_k} > UB_n) \text{ or } (UB_{P_k} < LB_n) \tag{6.6}$$

- The path goes through n, or is in the fan-in cone of n.

A *Candidate Path* P_k with delay PD_k is the best path if it gives the smallest R_{dmin} for the current target test clock T_{clk_i} where $R_{dmin} = (T_{clk_i} - PD_k)$. A similar path store structure was used in [65], in which the best path is always the path with the largest UB on delay since the test clock period will always be larger than any path delay. In our case, three possible categories of path can exist as shown in Figure 6.5. In case *a*, the entire TW of the path is smaller than the target test clock period T_{clk_i}, in case *b* T_{clk_i} is between the Upper Bound (UB) and Lower Bound (LB) of the path TW, while in the third case *c*, path TW entirely exceeds T_{clk_i}. Clearly, if a path belongs to the last category, it cannot be used for testing using the current target test clock and hence is never added to the path store. The paths belonging to the other two categories belong to the path store. To be able to access good candidates first, the path store can be sorted. There are two options here, either the paths can be sorted by their lower bounds or by their upper bounds. As new paths are created by extending current partial paths, the paths become longer, thereby increasing the lower bounds of the paths TW. When the lower bound becomes higher than the target test clock period, the path need not be added to the store. If the TW of a partial path has a higher

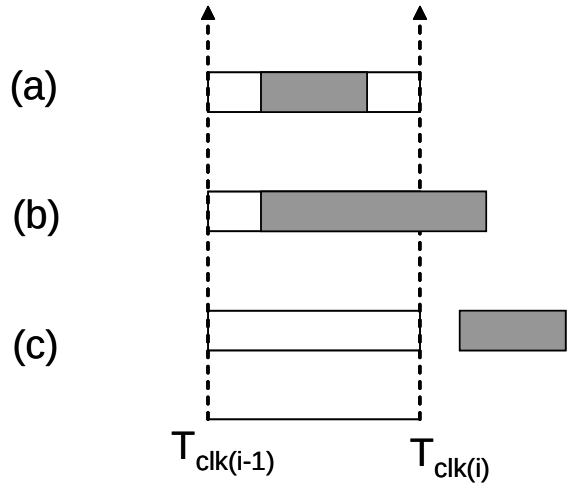


Figure 6.5: Path TW overlap with the current TI

LB, it means that there is lower overlap with the target TI and hence it is not the best candidate for extending. Thus, in order to select the best path first from the path store, the paths are sorted by the UBs of their TWs.

For any target node n, the path store is searched and the best *Candidate Path* is picked and a new paths are created by extending to the outgoing edges. A newly created path P_k is added to the store only if

- The node assignments required for Robust sensitization have no static implication conflicts
- The path TW overlaps with TI_i. This requires

$$(LB_{P_k} > UB_{TI_i}) \text{ or } (UB_{P_k} < LB_{TI_i}). \tag{6.7}$$

Thus overall, there are two timing constraints, the first is specific to the current TI and the second corresponds to the current target node. A path is propagated (extended), only if

it meets the node specific timing constraints, and a new path is added to the store only if it meets the constraints for the target TI. The search continues till either a complete and *Acceptable* R_{dmin} path is found for the target node, or there is no potential path for the target node in the current target TI.

If a complete path is found, then a SAT based ATPG procedure is called to obtain a test vector for robustly testing the path. Since any path selected for a target node using the above method has to be an *Acceptable* path and the best *Candidate Path*, it is guaranteed to give the smallest R_{dmin} value for the target node in the current TI. For the selected path, the R_{dmin} values for all the other nodes on the path are also updated. Thus, every time a path through a node m is found, its TW is updated, such that

$$TW_m = [Max(LB_{TI_i}, LB_m), Min(UB_{TI_i}, UB_m, (UB_{TI_i} - R_{dmin}(m)))] \tag{6.8}$$

Thus the node timing window is continually shrinking, which narrows down the search space, since the number of available *Candidate Paths* will reduce due to constraints from 6.6. Thus for any node which is not marked as *Covered* but has an $R_{dmin} < \infty$, we only select paths that will give an R_{dmin} better than its current known R_{dmin}. The selected candidate path is then removed from the path store before starting the search for the next target node. The complete algorithm is shown in Figure 6.6.

6.5 Computational Complexity

The overall algorithm is linear in the number of nodes in the DAG. In the worst case, all nodes need to processed for each target T_{clk}. The path store is recreated for each target test clock, but is common to all nodes within a target TI. By having a common path store, we need not recreate the same paths for each node, as many paths are shared between different

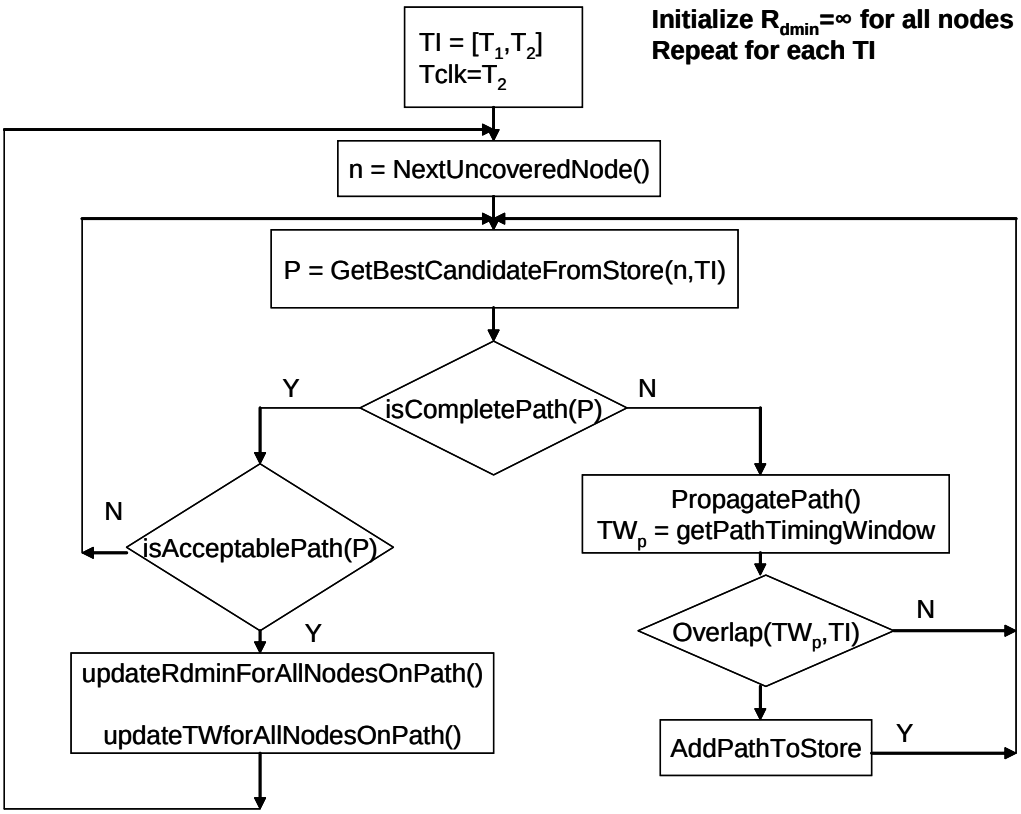


Figure 6.6: Path selection algorithm)

nodes. The candidate path selection is specific to any given target node and is based on the TW of the target node, while the decision whether a new path is to be added to the store is based solely on the target TI since the path might be a candidate for some other node. One problem with having a common path store, however, is that the size of the path store can grow significantly. A large number of paths in the store also increases the search time required to find the node specific candidate paths. In our current implementation, paths are eliminated based on only the static implications and the TWs of the paths. Advanced path pruning techniques such as forward trimming [65], or false path elimination based on SAT based learning [13] can be used to improve the run time of the algorithm. Since for each

target TI, the path store only contains paths whose TWs overlap with the current target TI, a potential method of controlling the path store size would be to decide the target test clock frequencies based on the distribution of path delays in the circuit. For instance, the test clock frequencies could be selected so that paths are distributed evenly over the different target TIs. In this work, we select the test clock frequencies arbitrarily and have not used the path distribution information to make the decision.

6.6 Simulation Results

The R_{dmin} path selection algorithm discussed in this chapter was implemented in C++. A standard cell library built using 130nm technology was characterized for timing at various load and slope points. This timing information was then used to estimate the pin-to-pin delays of the circuit. Tables 6.1 and 6.6 shows the results observed for some of the ISCAS89 benchmark circuits. For each test circuit, first the maximum circuit delay is estimated during the initial static timing analysis. Four different test clock frequencies are assumed for each circuit, $(T_{clk_1} = 0.4 * T_{sclk}, T_{clk_2} = 0.6 * T_{sclk}, T_{clk_3} = 0.8 * T_{sclk}, T_{sclk})$, where T_{sclk} is the maximum circuit delay. Table 6.1 shows the the total number of nodes (N_{nodes}), the total number of paths in the circuit and the selected R_{dmin} paths. As mentioned in the previous section, one of the criteria for a path P_k with delay distribution (μ_k, σ_k) to be an acceptable path for test clock T_{clk_i} is that $(\mu_k + 3\sigma_k) < T_{clk_i}$. For these test circuits, our experience was that this particular constraint was rather hard to satisfy. There were several nodes which had testable paths with delays less than at least one of the target test clocks but could not be classified as an *acceptable* R_{dmin} path since this variance constraint was not satisfied. Column 5 of Table 6.1 has the total net coverage (NC) and column 6 shows the percentage of nodes for which this variance constraint had to be relaxed (NC_R).

Table 6.1: Experimental results on ISCAS benchmark circuits

Circuit	N_{nodes}	Total paths	R_{dmin} Paths	NC (%)	NC_R (%)	Time (s)
s510	238	738	184	97.8	3.7	1.4
s641	435	3488	326	97.4	0	11.0
s713	449	43624	325	94.2	0	24.3
s838	514	3428	349	99.2	11.6	14.3
s953	442	2312	363	99.7	0.1	5.7
s1196	563	6196	465	99.8	5	15.7
s1238	542	7118	454	100	4.2	17.6
s1423	750	89452	526	100	0.5	207.4
s1488	669	1924	599	98.5	2.8	12.9
s3271	1729	38362	1380	100	4.3	46.9
s3330	1964	9458	1305	98.7	11.9	36.6
s3384	1922	39520	3357	99.7	5	607.0
s5378	2995	27084	2389	100	5.9	116.6
s9234	5846	489708	4217	99.8	12.9	2968.2

Finally column 7 shows the run time of the algorithm for each circuit. Table 6.6 gives the distribution of R_{dmin} paths found for each of the target test clocks. The median of the R_{dmin} values (in ps), and the number of paths found for each target test clock are shown. An empty cell in one of these columns implies that for the corresponding target timing interval no paths with a better R_{dmin} value were found.

The same results have been plotted in a the form of a bar chart in Figure 6.7, which shows the percentage of paths selected in each TI and Figure 6.8 which shows average R_{dmin} values selected in each TI. The results show that the almost all circuits, the number of test paths found for the fastest test clocks are the highest. A node for which a test path with $R_{dmin} \leq 50ps$ is found, is considered covered, and we do not process this node for the next test clock. This implies that for most of the nodes, either an acceptable R_{dmin} path was

Table 6.2: Experimental results on ISCAS benchmark circuits

Circuit	R_{dmin}(ps)				Num Paths			
	T_{clk_1}	T_{clk_2}	T_{clk_3}	T_{clk_4}	T_{clk_1}	T_{clk_2}	T_{clk_3}	T_{clk_4}
s510	22	34	33	34	74	81	23	6
s641	53	56	-	-	296	30	-	-
s713	61	64	80	-	311	13	1	-
s838	37	45	59	-	179	99	71	-
s953	22	27	31	-	187	159	17	-
s1196	42	46	48	-	357	93	15	-
s1238	46	44	46	-	362	85	7	-
s1423	81	78	-	-	497	29	-	-
s1488	45	57	82	72	224	270	94	11
s3271	39	37	52	-	1143	222	15	-
s3330	46	52	53	135	669	329	126	181
s3384	666	83	366	-	1308	2018	31	-
s5378	35	41	65	-	1453	861	75	-
s9234	100	53	69	-	3592	617	8	-

found for the faster test clock or no better path could be found for the slower clocks.

The R_{dmin} values also tend to be smaller for the faster test clocks. This is because under the constraint of $(\mu_k + 3\sigma_k) < T_{clk_i}$, the paths with smaller delay deviations will be closer to the test clock and hence give better R_{dmin}. There were some exceptions to this observation, especially circuits $s9234$ and $s3384$ where the median R_{dmin} for the fastest test clock was much higher than the other test clocks. It was found that for these circuits a significant number of nodes have the upper bound of their TWs much lower than the shortest test clock period. This means that these nodes can be covered only by the fastest test clock available but the paths covering the nodes still have a large slack. The only way to improve the R_{dmin} values for such cases is to have much faster test clocks (e.g. $T_{clk} = 0.2 * T_{sclk}$). The actual R_{dmin} value will also be affected by the clock skew, but has not been accounted

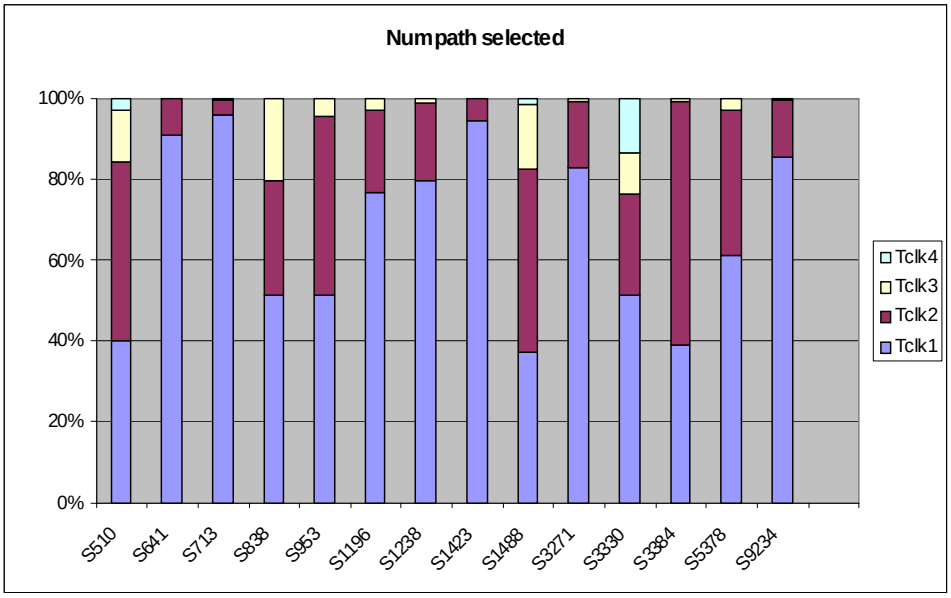


Figure 6.7: Number of paths selected

for here.

To show that the paths selected using our algorithm gives better R_{dmin} values as compared to selecting the longest path through each node as done in [65], we compare the R_{dmin} values obtained using the two methods for the s3330 and s5378 ISCAS benchmark circuits. The results are plotted in Figure 6.9(a) and 6.9(b) respectively.

6.7 Conclusion

In traditional path based delay testing, where the objective is to detect any timing failures with respect to a specified clock, good fault coverage can be obtained by selecting the longest path through each node. However, if delay testing is being used for detecting small reliability defects that may not cause timing failures, the test effectiveness should be measured by the size of the smallest detectable defects. While previous research has

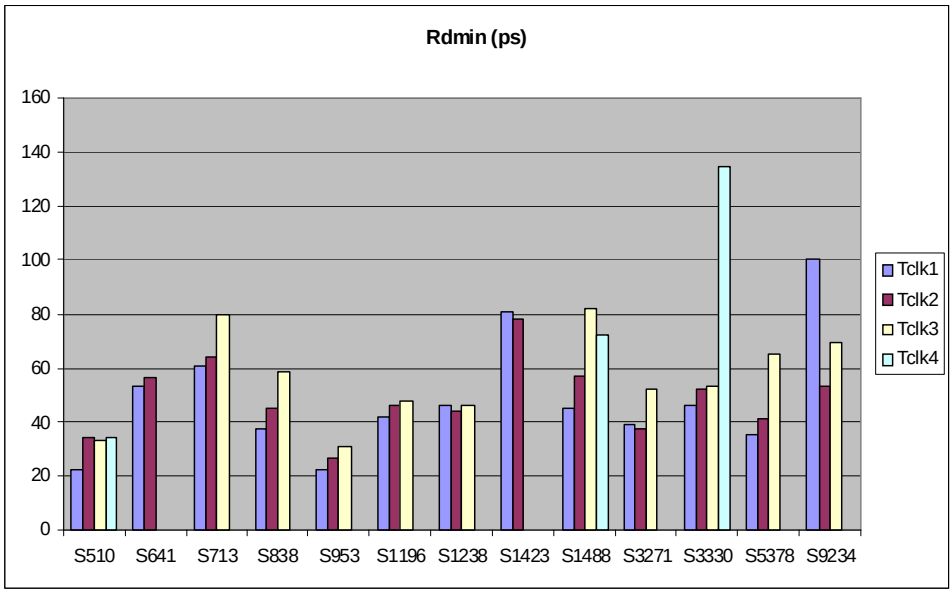
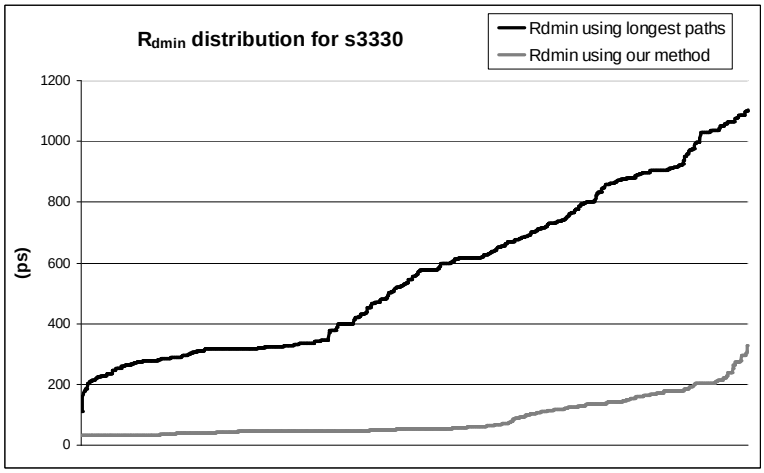
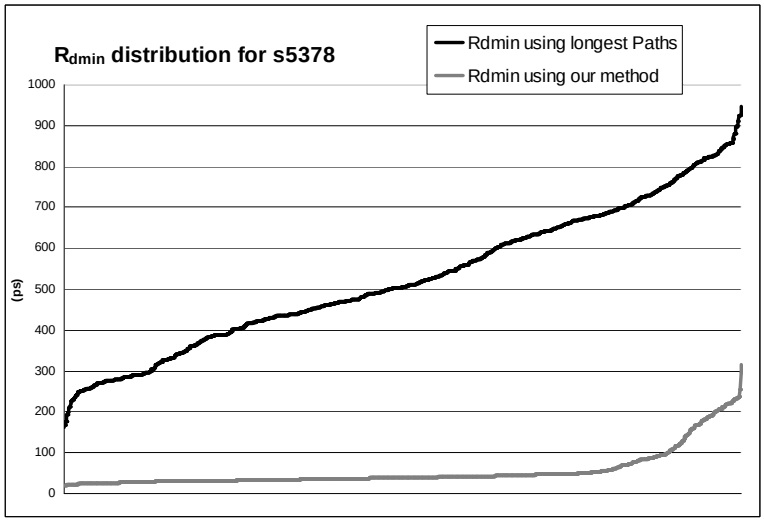


Figure 6.8: Average R_{dmin} for selected paths

emphasized on the need for faster than at-speed capture for detecting these reliability defects, there is no path selection technique that is targeted for faster clocks. In the previous chapter we showed that if faster clocks are available, then the longest paths need not be the best paths. This is because process variations affect the size of defects that can be detected and we showed that the detectable defect size is reduced by using shorter paths assuming that faster than at-speed capture is feasible. We apply this analysis in a path selection algorithm that focuses on maximizing the defect coverage at any node, given a set of test clock frequencies. Experimental results show that the size of resistive interconnect defects that can be detected using our path selection method are much smaller than the traditional method of selecting the longest paths.



(a) s3330



(b) s5378

Figure 6.9: R_{dmin} using our method Vs [65]

Chapter 7

Future Work and Conclusion

7.1 Supply Noise Effect on Delay Test

The supply voltage seen at any node in a circuit is different from the ideal value at which it is designed. The power supply is distributed to the circuit nodes using metal grid structure, which should be robust enough to carry all the required current. The current provided from the package bumps needs to travel through the various metal layers and via stack before it reaches the transistor nodes. Thus the power supply network is typically modeled as a resistive grid or mesh structure. During circuit switching, current flows from the supply network and to the ground network. Since the supply networks are resistive, as current is drawn during switching, it creates a voltage drop on the current path, and thus the actual supply voltage seen by a gate could be lower than the designed value. This drop in supply voltage is referred to as the IR drop. The IR drop seen at any node can be reduced if there are multiple current paths, since the resistances will be in parallel, thus reducing the effective resistance. IR drop is an instantaneous phenomenon which occurs only when switching activity happens. The larger the number of gate switching simultaneously in a circuit, the larger is the current drawn from the power supply and hence the larger is the IR drop. Thus the IR drop seen in any circuit depends on the power grid structure and the amount of switching activity that can happen at any given time. Given that modern circuits have millions of gates, there are too many possible switching configurations and it is

difficult to model the switching behavior of a circuit over time. Thus even though IR drop is a deterministic phenomenon, it is also referred to as supply noise.

The propagation delay of any gate increases as the supply voltage degrades and therefore supply noise plays an important part in circuit timing. Traditional industry practice is to decide on an acceptable lower bound on the supply voltage (VCC_{min}) and the power supply grid is then designed such that the IR drop assuming worst case switching is always lower than the allowed threshold. Static timing verification is then done at VCC_{min} and circuit is considered timing converged only if it meets timing constraints at the VCC_{min} corner. This approach is clearly pessimistic since it assumes that all nodes in the circuit will see the worst case IR drop. Such pessimism is acceptable for static timing verification where the objective is to estimate the upper bound on circuit delay and ensure that this upper bound is less than the target clock period, since it leaves extra margin for un-modeled variations. However, excessive pessimism should be avoided during critical path selection for delay testing as it can result in incorrect path ranking and hence missing some real critical paths as was demonstrated in Chapter 3.

The problem of estimating the dynamic variation on a path due to the supply noise effect is very different as compared to the coupling or MIS effects. Unlike coupling or MIS noise, where the delay change is due to local interactions related to the path, supply noise is a more global effect. If the path criticality is to be computed based on the worst case path delay considering supply noise effect, then we need to find the maximum switching activity that can happen in the circuit when the target path is active. Thus, this problem can be formulated as a constrained optimization problem where the objective is to maximize the simultaneous switching activity in the circuit while applying appropriate logic constraints. In addition to that, the sensitivity of any path to supply noise will depend on the power grid

structure, and the number of current paths available for the nodes on the path. This will require accurate modeling of the power supply network and the gate locations.

The supply noise effect also needs to be considered during test generation for delay testing. While path selection step only requires estimating the maximum path delay (without being too pessimistic), test generation requires identifying the test vector that will introduce the worst case delay. In [42], genetic algorithm based search techniques were used to obtain the vectors that maximize supply noise. Again, supply noise effect is different that other dynamic effects, and has unique issues for scan based testing. It has been shown before that scan operation causes excessive switching on the circuit nodes which is much higher than that seen during mission mode. This is because as test vectors are being scanned in one after another, the values at the output of the scannable flops keep toggling, which in turn triggers random switching on the internal circuit nodes. This excessive supply noise can result in exaggerated performance degradation during test mode which can then lead to yield loss. Thus during test vector generation for maximizing supply noise effect, it is important to model the mission mode behavior to prevent unnecessary yield loss.

7.2 Technology Trends in the MIS Effect

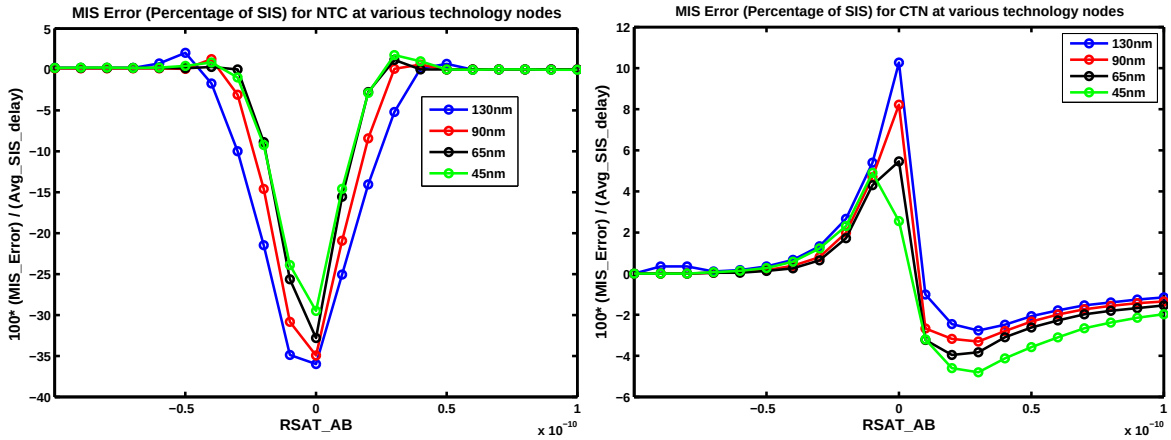
In Chapter 2, we discussed the dynamic delay variation introduced due to MIS effect and why it is important to model MIS effect during path selection and test generation for delay testing. In this research we have developed simple analytical models that can be used to estimate the delay change due to MIS, and hence estimate path criticality more accurately. However, further work is required for developing efficient techniques for incorporating the MIS effect in path selection and test generation methodology. For each node on a target path, the RSAT distribution needs to be computed by first determining the signal arrival

time distributions of the off-path inputs.

Another important factor that requires further investigation is the trend of MIS effect with technology scaling. While it is well known that coupling effects and supply noise effects become worse with scaling, the trend in MIS effect is not obvious. We did some simple experiments to study the effect of scaling on MIS using the BPTM models presented in [83]. In order to normalize the MIS dependency on output load and input slope, we performed the following simulations.

- For each of the technology node, the optimal W_p/W_n ratio that gives equal rising and falling delays were first obtained.
- If all logic gates are sized for FO4 delays, then the transition times or slopes generally do not vary too much from stage to stage. Such a typical slope value was then estimated for each technology node by simulating a large network of inverters, each having a fanout of 4 inverters.
- For each technology node, a 2-input NAND gate was simulated with input slope equal to the typical slope characterized for that technology and FO4 output load.

Figure 7.1(a) shows the percentage delay error induced due to MIS ($100 \times \Delta T_Z^M/\Delta T_Z^S$) for NTC transitions, and Figure 7.1(b) shows the same for CTN transitions. The MIS delay error is obtained by computing the distance of T_Z^M from both its asymptotes, which are nothing but ATs at Z for SIS transitions at A and B and ΔT_Z^S is the average SIS delay. For NTC transitions at the inputs, the MIS delay is much lower than the SIS delay and hence MIS effect needs to be accounted for during min-delay analysis only. From Figure 7.1(a), it can be seen that the MIS effect reduces the SIS delay by approximately -38% at the 130nm



(a) Case: NTC (b) Case: CTN

Figure 7.1: Percentage MIS delay change for different technology nodes

node while the change is reduced to -30% for the 45nm node. Even though a 30% error is still significant, it is interesting to observe that MIS delays are becoming closer to SIS delays as technology is scaling. This means that in future technologies, the optimism (for min-delay analysis) using SIS delay estimates will be lower. For CTN transitions, the MIS delay could be either lower than the SIS case, or higher, depending on the RSAT value and input location in the series stack. Thus the MIS effect needs to be considered for both min-delay and max-delay analysis. Figure 7.1(b) shows that for positive errors, the peak MIS delay is 10% higher than the SIS delay for the 130nm node, while the difference reduces to 5% for 45nm node. The negative error, however, is increasing in magnitude, but the peak negative error is still around 5%. Since the effect of MIS in the CTN case tends to be much lower, it can be ignored as a tradeoff to characterization effort. Next, we compute MIS error over the entire RSAT range (area under the error curve) given by $\int_{RSAT}(\Delta T_Z^M - \Delta T_Z^S)/\Delta T_Z^S dt$ for each technology node. If we consider the error area for the NTC case at 130nm to be 1, then the trend in the MIS error can be seen clearly as shown in Figure 7.2, where all the

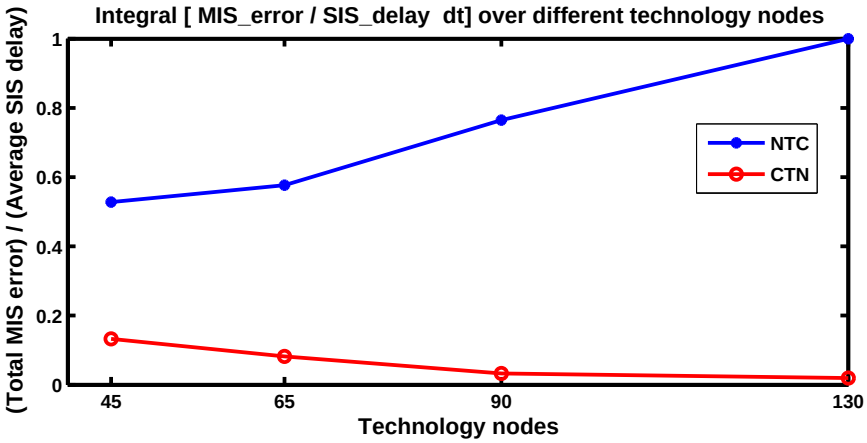


Figure 7.2: Total MIS error for different technology nodes

data points have been normalized by the value for the 130nm node.

The switching delay of a gate is directly dependent on the current drawn by the gates. A possible explanation for this trend in MIS behavior with technology scaling is the change in I_{on}/I_{off} ratio with scaling. If we consider the first order delay approximation using average current method, then for a NTC transition at the input of a 2-input NAND gate, the ouput will be rising (R). Therefore, we have

$$\tau_r = \frac{C_{load} \times \Delta V_r}{I_{avg,r}} \tag{7.1}$$

In the SIS case, if only input A has a NTC transition, then the total current is $I_{on,P_A} + I_{off,P_B}$, while in the MIS case, when both the inputs have NTC transitions, the total current would be $I_{on,P_A} + I_{on,P_B}$. Thus the ratio of MIS delay to SIS delay will be

$$\frac{\tau_{r,MIS}}{\tau_{LH,SIS}} = \frac{I_{on,P_A} + I_{on,P_B}}{I_{on,P_A} + I_{off,P_B}} \tag{7.2}$$

Notice that the I_{off} current we are refering to is the average transient current that will be conducted through the PMOS that is supposed to be OFF. This current value will be much

higher (peak of approximately 10μ) as compared to the dc I_{off} values that are much lower. The delay change due to MIS as a percentage of SIS delay is given by

$$MIS_{error,LH} = (\tau_{LH,SIS} - \tau_{LH,MIS})/\tau_{LH,SIS} \tag{7.3}$$

Combining the two expressions, we have

$$MIS_{error,LH} = (0.5 - 0.5\frac{I_{off,LH}}{I_{on,LH}}) \tag{7.4}$$

Here we assumed that the ON currents of both the PMOS transistors is the same. Thus as the I_{off}/I_{on} increases, the MIS delay becomes closer to the SIS delay and hence the percentage delay difference introduced due to the MIS effect reduces. Thus in general, more leaky gates will have less delay change due to MIS effect, since MIS delays will become closer to SIS delays. The experiments are done here use the BPTM models which might not be very accurate. The above analysis is very preliminary and this problem requires further theoretical analysis, supported by simulation results using more accurate process models.

7.3 Reducing the Gap between Functional and Structural Testing

One of the main motivations of this research was to investigate the gap between the test results obtained using structural delay testing and traditional functional testing. The advantage of scan based structural testing is that tests can be targeted towards a given fault model, or a set of paths and test effectiveness can be measured using the fault or defect coverage metric. The challenge with scan based delay testing for checking the timing performance of a circuit is that the tests might not represent real functional modes and hence there is always a chance of increasing yield loss (by being pessimistic) or missing some real faults (by being optimistic).

We believe there are two main reasons for the gap between scan based delay testing and at-speed functional tests; one, the paths being tested are not the real critical paths, and two, the vectors being used to test are not the optimal vectors. The dynamic delay effects are grossly overestimated using pre-silicon static timing analysis and hence STA tools typically fail to identify the real critical paths in a circuit. At the same time, performing dynamic system level simulations with accurate delay models could be infeasible considering the size and complexity of most modern circuits. In this research we primarily target the problem of path selection. The techniques developed in this research will also significantly reduce the search space for selecting optimal test vectors. However, generating the desired dynamic noise using scan based testing has additional challenges. Since scan based testing is typically constrained by test time, test compression is an important step of the process. The compressed test set will have the same fault coverage as the uncompressed one, but will have increased switching activity since multiple faults are targeted using a single vector. Thus even though test vectors that generate the desired dynamic noise are found, the effect could be lost after test compression. In addition to that, as explained earlier, delay testing either requires special scan structures or the test vectors need to be designed for *Launch on Shift* or *Launch on Capture* schemes which incurs additional limitations on the tests that can be selected.

Functional testing, on the other hand, involves running real applications on the test chip and hence represent mission mode operation. The challenge here is that the effectiveness (or coverage) of the tests is hard to quantify. For instance, it is difficult to claim with certainty that all the possible worst case combinations have been activated during test. In addition to that, it is difficult to perform tests that are targeted to activate a given path or a segment.

Recently, there has been work on mapping test vectors for path delay faults to real

instruction sequences of a processor [31]. These instructions sequences can then be run from cache in native mode, and hence can be targeted for any selected paths or segments without requiring any scan. This technique overcomes the shortcomings of scan based testing and at the same time can mimic the functional tests more effectively. Since the test time is no longer an issue, test compression is not required. Also, test vectors need not be designed for *LOS* or *LOC* schemes since the tests will be run directly from the cache and the circuit will be running at-speed. A good extension of our work would be to generate tests that can be mapped to instructions while maximizing the dynamic noise on the path. Our current work is targeted for selecting the real critical paths, and the future work can be towards finding the optimal instruction sequences to test these paths in native mode.

7.4 Conclusion

In this research we have studied the effects of dynamic and process variability, and provided solutions to deal with the delay uncertainty introduced due to variability. We have addressed two different applications of delay testing, each of which have unique challenges. The first application uses delay testing for ensuring a circuit meets timing at a specified frequency and the second application uses delay testing as a reliability screen for detecting small delay defects.

For the first case, we use path based delay fault models and emphasize on the effect of dynamic delay variations. Path based delay testing has two important steps, path selection and test generation. Selecting the optimal set of paths for delay testing has always been a challenge due to the exorbitant number of paths in modern circuits. The delay variation caused by uncertainty in process parameters makes critical path selection even more challenging. While a significant amount of the previous research tries to model the effect

of process variability during path selection, the dynamic delay effects have been considered only during the test generation phase. In this research we show that it is important to consider the dynamic delay variability during the path selection phase itself, since it will affect the critical paths being selection. Since dynamic effects such as coupling, multiple-input switching and supply noise are vector dependent, estimating the total delay variability is very difficult. Our solution is to estimate the maximum vector delay of any path without being too pessimistic. During path selection, the path criticality is then computed based on the estimated worst case path delay. Traditional static timing analysis tools are excessively pessimistic when modeling the dynamic delay effects, which is appropriate for timing verification but not for critical path selection. We show that if the worst case path delay is computed using techniques used by STA tools, then it can affect path ranking and hence result in incorrect path selection. Our theory is supported by the observations that critical path reported by STA often do not match the real critical paths on silicon [47]. Since path delay is vector dependent and vectors are known only during test generation, path selection and test generation cannot be independent steps. In this research, we have analyzed two dynamic delay effects, namely coupling and MIS. For coupling noise, we have developed a simple model for estimating MCF as a function of victim and aggressor signal arrival times and developed a efficient path selection algorithm that uses path criticality based on maximum path delay estimate considering coupling noise. For MIS, we have developed simple analytical models that can be used for estimating worst case path delay distribution during path selection.

If delay testing is being used for small delay defect detection, then the main challenge is being able to perform faster than at-speed testing. In this research we have developed a circuit scheme called the PCG (Programmable Capture Generator) which facilitates faster

than at-speed testing, by controlling the capture edge arrival time. The capture period can be accurately controlled by programming the PCG using scan. Thus the test vector itself can have the code for the test frequency. We further analyze how process variations can affect the defect coverage if the PCG is being used for detecting small delay defects. We show that if the effect of process variability on path delay is considered, then the shorter paths can actually give better defect coverage than the longer paths. This is contrary to the well established method of selecting the longest paths in the circuit for detecting the smallest defects. The difference here is that longest paths are best only for at-speed testing, but if faster than at-speed is enabled, then the optimal paths are different. We have developed an efficient path selection algorithm that is targeted towards multiple fixed capture frequencies, so that each path need not be tested separately. The algorithm then selects the best path, that is, the path that minimizes the detectable defect size for each node in the circuit.

Thus, the path selection strategy should be different for different delay test applications. For defect based test, where the objective is to detect the smallest possible defects, faster than at-speed testing is required and shorter paths can be better. On the other hand when testing for circuit timing failure, at-speed testing is required and the optimal path set should consider the most critical paths in the circuit, where criticality is computed considering both dynamic and process variations.

Bibliography

[1] Aseem Agarwal, Florentin Dartu, and David Blaauw. Statistical Gate Delay Model Considering Multiple Input Switching. In *DAC '04: Proceedings of the 41st annual conference on Design automation*, pages 658–663, New York, NY, USA, 2004. ACM.

[2] Kanak Agarwal, Yu Cao, Takashi Sato, Dennis Sylvester, and Chenming Hu. Efficient Generation of Delay Change Curves for Noise-Aware Static Timing Analysis. In *ASP-DAC '02: Proceedings of the 2002 conference on Asia South Pacific design automation/VLSI Design*, page 77, Washington, DC, USA, 2002. IEEE Computer Society.

[3] Nisar Ahmed, C. P. Ravikumar, Mohammad Tehranipoor, and Jim Plusquellic. At-Speed Transition Fault Testing With Low Speed Scan Enable. In *VTS '05: Proceedings of the 23rd IEEE VLSI Test Symposium*, pages 42–47, Washington, DC, USA, 2005. IEEE Computer Society.

[4] Nisar Ahmed, Mohammad Tehranipoor, and Vinay Jayaram. Timing-Based Delay Test For Screening Small Delay Defects. In *DAC '06: Proceedings of the IEEE Design Automation Conference*, pages 320–325, 2006.

[5] Charles J. Alpert, Anirudh Devgan, and Chandramouli Kashyap. A Two Moment RC Delay Metric For Performance Optimization. In *ISPD '00: Proceedings of the 2000 international symposium on Physical design*, pages 69–74, New York, NY, USA, 2000. ACM.

[6] Chirayu Amin, Chandramouli Kashyap, Noel Menezes, Kip Killpack, and Eli Chiprout. A Multi-Port Current Source Model For Multiple-Input Switching Effects In Cmos Library Cells. In *DAC '06: Proceedings of the 43rd annual conference on Design automation*, pages 247–252, New York, NY, USA, 2006. ACM.

[7] Martin Amodeo and Bruce Cory. Beyond at-speed. In *Test and Measurement World*, Nov. 2005.

[8] Xiaoliang Bai, Sujit Dey, and Angela Krstic. HyAC: A Hybrid Structural SAT Based ATPG for Crosstalk. In *ITC '03: Proceedings of the IEEE International Test Conference*, volume 1, pages 112–121, Washington, DC, USA, Oct. 2003. IEEE Computer Society.

[9] Keith Baker, Guido Gronthoud, Maurice Lousberg, Ivo Schanstra, and Charles Hawkins. Defect-Based Delay Testing of Resistive Vias-Contacts A Critical Evaluation. In *ITC '99: Proceedings of the 1999 IEEE International Test Conference*, page 467, Washington, DC, USA, 1999. IEEE Computer Society.

[10] Matthias Beck, Olivier Barondeau, Martin Kaibel, Frank Poehl, Xijiang Lin, and Ron Press. Logic Design for On-Chip Test Clock Generation - Implementation Details and Impact on Delay Test Quality. In *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, pages 56–61, Washington, DC, USA, 2005. IEEE Computer Society.

[11] O. Bula, J. Moser, J. Trinko, M. Weissman, and F. Woytowich. Gross delay defect evaluation for a CMOS logic design system product. In *IBM Journal of Research and Developmen*, volume 34, pages 325–338, Riverton, NJ, USA, 1990. IBM Corp.

[12] V. Chandramouli and Karem A. Sakallah. Modeling The Effects Of Temporal Proximity Of Input Transitions On Gate Propagation Delay And Transition Time. In *DAC '96: Proceedings of the 33rd annual conference on Design automation*, pages 617–622, New York, NY, USA, 1996. ACM.

[13] Kameshwar Chandrasekar and Michael S. Hsiao. Integration of Learning Techniques into Incremental Satisfiability for Efficient Path-Delay Fault Test Generation. In *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, volume 2, pages 1002–1007, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[14] Hoon Chang and Jacob A. Abraham. VIPER: An Efficient Vigorously Sensitizable Path Extractor. In *DAC '93: Proceedings of the 30th international conference on Design automation*, pages 112–117, New York, NY, USA, 1993. ACM.

[15] Hoon Chang and Jacob A. Abraham. An Efficient Critial Path Tracing Algorithm for High Performance VLSI Systems. In *Journal of Electronic Testing*, volume 11, pages 119–129, Norwell, MA, USA, 1997. Kluwer Academic Publishers.

[16] Jonathan Chang and Edward McCluskey. Detecting Delay Flaws By Very-Low Voltage Testing. In *ITC '96: Proceedings of the International Test Conference*, pages 367–376, Washington, DC, USA, Oct. 1996. IEEE Computer Society.

[17] Chih-Ang Chen and Sandeep K. Gupta. A Satisfiability-Based Test Generator For Path Delay Faults In Combinational Circuits. In *DAC '96: Proceedings of the 33rd annual conference on Design automation*, pages 209–214, New York, NY, USA, 1996. ACM.

[18] Liang-Chi Chen, Sandeep K. Gupta, and Melvin A. Breuer. A New Gate Delay Model For Simultaneous Switching And Its Applications. In *DAC '01: Proceedings of the 38th*

conference on Design automation, pages 289–294, New York, NY, USA, 2001. ACM.

[19] Pinhong Chen, Desmond Kirkpatrick, and Kurt Keutzer. Miller Factor For Gate-Level Coupling Delay Calculation. In *ICCAD '00: Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design*, pages 68–75, Piscataway, NJ, USA, 2000. IEEE Press.

[20] Weiyu Chen, Sandeep K. Gupta, and Melvin A. Breuer. Test Generation In Vlsi Circuits For Crosstalk Noise. In *ITC '98: Proceedings of the 1998 IEEE International Test Conference*, page 641, Washington, DC, USA, 1998. IEEE Computer Society.

[21] Kwang-Ting Cheng. Transition Fault Simulation for Sequential Circuits. In *ITC '92: Proceedings of the 1992 IEEE International Test Conference*, pages 723–731, Washington, DC, USA, 1992. IEEE Computer Society.

[22] Kwang-Ting Cheng, Li-C. Wang, and Jing-Jia Liou. On Theoretical And Practical Considerations Of Path Selection For Delay Fault Testing. In *ICCAD '02: Proceedings of the International Conference on Computer-Aided Design*, volume 0, pages 94–100, Los Alamitos, CA, USA, 2002. IEEE Computer Society.

[23] Bruce D. Cory, Rohit Kapur, and Bill Underwood. Speed Binning with Path Delay Test in 150-nm Technology. In *IEEE Design and Test of Computers*, volume 20, pages 41–45, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[24] Ramyanshu Datta, Antony Sebastine, and Jacob A. Abraham. Delay Fault Testing and Silicon Debug Using Scan Chains. In *ETS '04: Proceedings of the European Test Symposium, Ninth IEEE*, pages 46–51, Washington, DC, USA, 2004. IEEE Computer Society.

[25] C. Fang. *Probabilistic Interval-Value Computation: Representing And Reasoning About Uncertainity In DSP And VLSI Design*. PhD thesis, Carnegie Mellon University, 2005.

[26] Piero Franco, Siyad C. Ma, Jonathan Chang, Yi-Chin Chu, Sanjay Wattal, Edward J. McCluskey, Robert L. Stokes, and William D. Farwell. Analysis and Detection of Timing Failures in an Experimental Test Chip. In *ITC '96:Proceedings of the IEEE International Test Conference*, pages 691–700, Washington, DC, USA, 1996. IEEE Computer Society.

[27] Jerome Friedman. Multivariate Adaptive Regression Splines. In *Annals of Statistics*, pages 1–14, 1991.

[28] Zhaohui Fu and Sharad Malik. Solving The Minimum-Cost Satisfiability Problem Using Sat Based Branch-And-Bound Search. In *ICCAD '06: Proceedings of the 2006 IEEE/ACM International Conference on Computer-Aided Design*, pages 852–859, New York, NY, USA, 2006. ACM.

[29] Anne Gattiker, Sani Nassif, Rashmi Dinakar, and Chris Long. Timing Yield Estimation from Static Timing Analysis. In *ISQED '01: Proceedings of the 2nd International Symposium on Quality Electronic Design*, page 437, Washington, DC, USA, 2001. IEEE Computer Society.

[30] X. Gefu and A. D. Singh. Low Cost Launch-On-Shift Delay Tst With Slow Scan Enable. In *ETS '06: Proceedings of the IEEE European Test Symposium*, pages 9–14, Washington, DC, USA, May 2006. IEEE Computer Society.

[31] Sankar Gurumurthy, Ramtilak Vemu, Jacob A. Abraham, and Daniel G. Saab. Automatic Generation of Instructions to Robustly Test Delay Defects in Processors. In *ETS*

'07: Proceedings of the IEEE European Test Symposium, volume 0, pages 173–178, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[32] Hong Hao and Edward J. McCluskey. Very-Low-Voltage Testing for Weak CMOS Logic ICs. In *ITC '93: Proceedings of the IEEE International Test Conference*, pages 275–284, Washington, DC, USA, 1993. IEEE Computer Society.

[33] Hamidreza Hashempour, Yong-Bin Kim, and Naphill Park. A Test-Vector Generation Methodology for Crosstalk Noise Faults. In *DFT '02: Proceedings of the 17th IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems*, pages 40–50, Washington, DC, USA, 2002. IEEE Computer Society.

[34] Keerthi Heragu, Janak H. Patel, and Vishwani D. Agrawal. Fast Identification Of Untestable Delay Faults Using Implications. In *ICCAD '97: Proceedings of the 1997 IEEE/ACM international conference on Computer-aided design*, pages 642–647, Washington, DC, USA, 1997. IEEE Computer Society.

[35] Shahdad Irajpour, Sandeep K. Gupta, and Melvin A. Breuer. Timing-Independent Testing Of Crosstalk In The Presence Of Delay Producing Defects Using Surrogate Fault Models. In *ITC '04: Proceedings of the International Test Conference on International Test Conference*, pages 1024–1033, Washington, DC, USA, 2004. IEEE Computer Society.

[36] Vikram Iyengar and et. al. At-Speed Structural Test for High-Performance ASICs. In *ITC '06: Proceedings of the IEEE International Test Conference*, pages 1–9, Washington, DC, USA, Oct 2006. IEEE Computer Society.

[37] H-S. Jun, S-S. Chung, and H. Kim. Programmable In-situ Delay Fauly Test Clock Generator. In *U.S. Patent No. 20060242474*, 2006.

[38] Janak Patel Keerthi Heragu and Vishwani Agrawal. Transition Fault Simulation for Sequential Circuits. In *VTS '96: Proceedings of 14th VLSI Test Symposium*, pages 32–39, Washington, DC, USA, 1996. IEEE Computer Society.

[39] Kee Sup Kim, S. Mitra, and P. Ryan. Delay Defect Characteristics And Testing Strategies. In *IEEE Design and Test of Computers*, volume 20, pages 8–16, Oct. 2003.

[40] Angela Krstic, Kwan-Ting Cheng, and Li-C. Wang. New Challenges in Delay Testing of Nanometer, Multigigahertz Designs. In *IEEE Design and Test*, volume 21, pages 241–247, Los Alamitos, CA, USA, 2004. IEEE Computer Society Press.

[41] Angela Krstic and Kwang-Ting Cheng. *Delay Fault Testing for VLSI Circuits*. Springer, 1998.

[42] Angela Krstic, Yi-Min Jiang, and Kwang-Ting (Tim) Cheng. Delay Testing Considering Power Supply Noise Effects. In *ITC '99: Proceedings of the 1999 IEEE International Test Conference*, page 181, Washington, DC, USA, 1999. IEEE Computer Society.

[43] Angela Krstic, Jing-Jia Liou, Kwang-Ting (Tim) Cheng, and Li-C. Wang. On Structural vs. Functional Testing for Delay Faults. In *ISQED '03: Proceedings of the 4th International Symposium on Quality Electronic Design*, page 438, Washington, DC, USA, 2003. IEEE Computer Society.

[44] Angela Krstic, Jing-Jia Liou, Yi-Min Jiang, and Kwang-Ting (Tim) Cheng. Delay Testing Considering Crosstalk-Induced Effects. In *ITC '01: Proceedings of the 2001*

IEEE International Test Conference, page 558, Washington, DC, USA, 2001. IEEE Computer Society.

[45] Medha Kulkarni and Tom Chen. A Sensitivity Based Approach to Analyzing Signal Delay Uncertainty of Coupled Interconnects. In *ISQED '04: Proceedings of the 5th International Symposium on Quality Electronic Design*, pages 331–336, Washington, DC, USA, 2004. IEEE Computer Society.

[46] Y. Satish Kumar, Jun Li, Claudio Talarico, and Janet Wang. A Probabilistic Collocation Method Based Statistical Gate Delay Model Considering Process Variations and Multiple Input Switching. In *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, pages 770–775, Washington, DC, USA, 2005. IEEE Computer Society.

[47] Leonard Lee, Li-C. Wang, Praveen Parvathala, and T. M. Mak. On Silicon-Based Speed Path Identification. In *VTS '05: Proceedings of the IEEE VLSI Test Symposium*, volume 0, pages 35–41, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[48] Huawei Li, Peifu Shen, and Xiaowei Li. Robust Test Generation for Precise Crosstalk-induced Path Delay Faults. In *VTS '06: Proceedings of the 24th IEEE VLSI Test Symposium*, pages 300–305, Washington, DC, USA, 2006. IEEE Computer Society.

[49] James C.-M. Li, Chao-Wen Tseng, and E.J. McCluskey. Testing for Resistive Opens and Stuck Opens. In *ITC '01: Proceedings of the 2001 IEEE International Test Conference*, volume 0, page 1049, Los Alamitos, CA, USA, 2001. IEEE Computer Society.

[50] Jing-Jia Liou, Angela Krstic, Li-C. Wang, and Kwang-Ting Cheng. False-Path-Aware Statistical Timing Analysis And Efficient Path Selection For Delay Testing And Timing

Validation. In *DAC '02: Proceedings of the 39th conference on Design automation*, pages 566–569, New York, NY, USA, 2002. ACM.

[51] Jing-Jia Liou, Li-C. Wang, Angela Krstic, and Kwang-Ting Cheng. Experience In Critical Path Selection For Deep Sub-Micron Delay Test And Timing Validation. In *ASPDAC '03: Proceedings of the 2003 conference on Asia South Pacific design automation*, pages 751–756, New York, NY, USA, 2003. ACM.

[52] Xiang Lu, Zhuo Li, Wangqi Qiu, D. Walker, and Weiping Shi. Longest-Path Selection For Delay Test Under Process Variations. In *IEEE Tran. On Computer-Aided Design of Integrated Ckts. and Systems*, volume 24, pages 1924–1929, 2005.

[53] Xiang Lu, Zhuo Li, Wangqi Qiu, D. M. H. Walker, and Weiping Shi. Longest path selection for delay test under process variation. In *ASP-DAC '04: Proceedings of the 2004 conference on Asia South Pacific design automation*, pages 98–103, Piscataway, NJ, USA, 2004. IEEE Press.

[54] Y.S. Mahajan, Z. Fu, and S. Malik. Zchaff2004: An Efficient SAT Solver. In *Theory and Applications of Satisfiability Testing*, volume 3542/2005, pages 360–375, Berlin / Heidelberg, 2005. Springer.

[55] A. Majhi, J. Jacob, L. Patnaik, and V. Agrawal. On Test Coverage of Path Delay Faults. In *VLSID '96: Proceedings of the 9th International Conference on VLSI Design*, pages 418–421, Washington, DC, USA, 1996. IEEE Computer Society.

[56] A.K. Majhi, J. Jacob, L.M. Patnaik, and V.D. Agrawal. An Efficient Automatic Test Generation System For Path Delay Faults In Combinational Circuits. In *VLSID '95:*

Proceedings of the 8th International Conference on VLSI Design, volume 0, page 161, Los Alamitos, CA, USA, 1995. IEEE Computer Society.

[57] Teresa L. McLaurin and Frank Frederick. The Testability Features Of The MCF5407 Containing The 4Th Generation Coldfire Microprocessor Core. In *ITC '00: Proceedings of the 2000 IEEE International Test Conference*, page 151, Washington, DC, USA, 2000. IEEE Computer Society.

[58] M. Milir, V. Kumar, and S. Tragoudas. High Quality Transition Fault Atpg For Small Delay Defects. In *IEEE Tran. on Computer Aided Design of Circuits and Systems*, pages 983–989, May 2007.

[59] A. Nabavi-Lishi and N. C. Rumin. Inverter Models Of Cmos Gates For Supply Current And Delay Evaluation. In *IEEE Tran. on Comp. Aided Design of Integrated Circuits*, pages 1271–1279, Oct 1994.

[60] Hiroyuki Nakamura, Akio Shirokane, Yoshihito Nishizaki, Anis Uzzaman, Vivek Chickermane, Brion Keller, Tsutomu Ube, and Yoshihiko Terauchi. Low Cost Delay Testing of Nanometer SoCs Using On-Chip Clocking and Test Compression. In *ATS '05: Proceedings of the 14th Asian Test Symposium on Asian Test Symposium*, pages 156–161, Washington, DC, USA, 2005. IEEE Computer Society.

[61] W. Needham, C. Prunty, and E. Yeoh. High Volume Microprocessor Test Escapes, An Analysis Of Defects Our Tests Are Missing. In *ITC '98: Proceedings of the International Test Conference*, pages 25–34, Washington, DC, USA, Oct. 1998. IEEE Computer Society.

[62] P. Nigh and A. Gattiker. Test Method Evaluation Experiments And Data. In *ITC '00: Proceedings of the Internatinal Test Conference*, pages 454–463, Washington, DC, USA, 2000. IEEE Computer Society.

[63] Sanjay Pant, David Blaauw, Vladimir Zolotov, Savithri Sundareswaran, and Rajendran Panda. Vectorless Analysis of Supply Noise Induced Delay Variation. In *ICCAD '03: Proceedings of the 2003 IEEE/ACM International Conference on Computer-aided Design*, page 184, Washington, DC, USA, 2003. IEEE Computer Society.

[64] R. Press and J. Boyer. Easily Implement PLL Clock Switching For At-Speed Test. In *Chip Design*, March 2006.

[65] Wangqi Qiu and D. M. H. Walker. An Efficient Algorithm for Finding the K Longest Testable Paths Through Each Gate in a Combinational Circuit. In *ITC '03: Proceedings of the 2003 IEEE International Test Conference*, volume 0, page 592, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[66] Karthik Rajagopal, Lawrence T. Pileggi, and Ravishankar Arunachalam. TACO: Timing Analysis with Coupling. In *DAC '00: Proceedings of the 37th conference on Design automation Conference*, volume 0, pages 266–269, Los Alamitos, CA, USA, 2000. IEEE Computer Society.

[67] R. Rajsuman. Iddq Testing for CMOS VLSI. In *Proceedings of the IEEE*, volume 88, pages 544–568, April 2000.

[68] Manish Sharma and Janak H. Patel. Finding a Small Set of Longest Testable Paths that Cover Every Gate. In *ITC '02: Proceedings of the 2002 IEEE International Test Conference*, page 974, Washington, DC, USA, 2002. IEEE Computer Society.

[69] Pei-Fu Shen, Hua-Wei Li, Yong-Jun Xu, and Xiao-Wei Li. Non-robust Test Generation for Crosstalk-Induced Delay Faults. In *ATS '05: Proceedings of the 14th Asian Test Symposium on Asian Test Symposium*, pages 120–125, Washington, DC, USA, 2005. IEEE Computer Society.

[70] Gordon Smith. Model for Delay Faults Based Upon Paths. In *ITC '85: Proceedings of the IEEE International Test Conference*, pages 342–349, Washington, DC, USA, 1985. IEEE Computer Society.

[71] Jayashree Sridharan and Tom Chen. Gate Delay Modeling with Multiple Input Switching for Static (Statistical) Timing Analysis. In *VLSID '06: Proceedings of the 19th International Conference on VLSI Design held jointly with 5th International Conference on Embedded Systems Design*, pages 323–328, Washington, DC, USA, 2006. IEEE Computer Society.

[72] S. Tani, M. Teramoto, T. Fukazawa, and K. Matsuhiro. 9.1 Efficient Path Selection for Delay Testing Based on Partial Path Evaluation. In *VTS '98: Proceedings of the 16th IEEE VLSI Test Symposium*, page 188, Washington, DC, USA, 1998. IEEE Computer Society.

[73] Chao-Wen Tseng, Ray Chen, Edward J. McCluskey, and Phil Nigh. MINVDD Testing for Weak CMOS ICs. In *VTS '01: Proceedings of the IEEE VLSI Test Symposium*, page 339, Washington, DC, USA, 2001. IEEE Computer Society.

[74] Chao-Wen Tseng, Edward J. McCluskey, Xiaoping Shao, and David M. Wu. Cold Delay Defect Screening. In *VTS '00: Proceedings of the 18th IEEE VLSI Test Symposium (VTS'00)*, page 183, Washington, DC, USA, 2000. IEEE Computer Society.

[75] Pramodchandran N. Variyam and Abhijit Chatterjee. Specification-Driven Test Design for Analog Circuits. In *DFT '98: Proceedings of the 13th International Symposium on Defect and Fault-Tolerance in VLSI Systems*, pages 335–340, Washington, DC, USA, 1998. IEEE Computer Society.

[76] R. Vollersten. Burn-in. In *Integrated Reliability Workshop*, pages 167–173, Oct. 1999.

[77] L-C. Wang, J-J. Liou, and K-T. Cheng. Critical Path Selection for Delay Fault Testing Based Upon a Statistical Timing Model. In *IEEE Tran. On Computer Aided Design*, pages 1550–1564, Nov. 2004.

[78] T. W. Williams, Bill Underwood, and M. R. Mercer. The Interdependence Between Delay-Optimization Of Synthesized Networks And Testing. In *DAC '91: Proceedings of the 28th conference on ACM/IEEE design automation*, pages 87–92, New York, NY, USA, 1991. ACM.

[79] Haihua Yan and Adit D. Singh. Evaluating the Effectiveness of Detecting Delay Defects in the Slack Interval: A Simulation Study. In *ITC '04: Proceedings of the International Test Conference on International Test Conference*, pages 242–251, Washington, DC, USA, 2004. IEEE Computer Society.

[80] S. Yanamanamanda, J. Li, and J. Wang. Uncertainty Modeling Of Gate Delay Considering Multiple Input Switching. In *ISCAS 2005. IEEE International Symposium on Circuits and Systems*, pages 2457–2460, Washington, DC, USA, May 2005. IEEE Computer Society.

[81] Jing Zeng, Magdy Abadir, G. Vandling, L.-C. Wang, S. Karako, and Jacob A. Abraham. On Correlating Structural Tests with Functional Tests for Speed Binning of High

Performance Design. In *MTV '04: Proceedings of the Fifth International Workshop on Microprocessor Test and Verification*, pages 103–109, Washington, DC, USA, 2004. IEEE Computer Society.

[82] J.-K. Zhao, E. M. Rudnick, and J. H. Patel. Static Logic Implication With Application To Redundancy Identification. In *VTS '97: Proceedings of the 15th IEEE VLSI Test Symposium (VTS'97)*, page 288, Washington, DC, USA, 1997. IEEE Computer Society.

[83] Wei Zhao and Yu Cao. New Generation of Predictive Technology Model for Sub-45nm Design Exploration. In *ISQED '06: Proceedings of the 7th International Symposium on Quality Electronic Design*, pages 585–590, Washington, DC, USA, 2006. IEEE Computer Society.

Vita

Rajeshwary G. Tayade was born in Amravati, Maharashtra on July 25 1978 to Shobha Tayade and Gunwantrao Tayade. Her initial schooling was done in Our Lady of Nazereth in Bhayander, Bombay in India. She completed her high school from S.P. College in Pune, India, and got her Bachelors in Engineering from Govt. College of Engineering, Pune, India, in the field of Electronics and Telecommunication. After working with Tata Infotech Ltd for one year, she decided to pursue higher education. She worked with Dr.Halverson at Texas A&M University (TAMU) in College Station to obtain a Masters degree in Electrical Engineering. After completing her masters, she worked briefly with Dr. Gwan Choi at TAMU on developing efficient decoder implementations for LDPC codes. During that period she also interned with the IBM Austin Research Lab. She joined the PhD program in the department of Electrical and Computer Engineering at the University of Texas at Austin in spring 2005. Since then, she has been working with Dr. Jacob Abraham in the field of VLSI testing and debug. She is currently working as a Component Design Engineer in the Ultra Low-power Mobile Devices group at Intel Corp., Austin.

Permanent address: 9404 Billingham Trail
Austin, Texas 78717

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.