Signal Integrity - Aware Pattern Generation for Delay Testing

Anu Asokan

Collège Doctoral du Languedoc-Roussillon

THÈSE

Pour obtenir le grade de

Docteur

Délivré par **l'Université Montpellier**

Préparée au sein de l'école doctorale **I2S**
Et de l'unité de recherche **LIRMM UMR 5506**

Spécialité : **Microélectronique**

Présentée par **Anu ASOKAN**

Signal Integrity – Aware Pattern Generation for Delay Testing

Soutenue le 9 Décembre 2015 devant le jury composé de

M. Emmanuel SIMEU	MCF, INP Grenoble	Rapporteur
M. Matteo Sonza REORDA	Professeur, Politecnico di Torino	Rapporteur
M. Philippe DEBAUD	Manager DFT, ST-Microelectronics	Examinateur
M. Olivier SENTIEYS	Professeur, Université Rennes 1	Examinateur
M. Alberto BOSIO	MCF, Université Montpellier	Co-encadrant
M. Arnaud VIRAZEL	MCF, Université Montpellier	Co-encadrant
M. Patrick GIRARD	Directeur de Recherche CNRS – LIRMM	Co-Directeur
M. Serge PRAVOSSOUDOVITCH	Professeur, Université Montpellier	Directeur

Université Montpellier

I dedicate this thesis to

my parents, my sister,

my husband,

my little cutie

&

to all the loving people in my life

Abstract

Signal Integrity-Aware Pattern Generation for Delay Testing

Advancing nanometer technology scaling enables higher integration on a single chip with minimal feature size. As a consequence, the effects of signal and power integrity issues such as crosstalk noise between interconnects, power supply noise and ground bounce in the supply networks significantly increases. Also, reliability issues are eventually introduced by variations in the manufacturing process. These issues will negatively impact the timing characteristics in an integrated circuit (IC), as they give rise to delay defects. Delay-related parametric failures increase the defect escape rate, yield loss and diminish reliability rate. Hence, design-for-test techniques are employed to have a better controllability and observability on the internal nodes to easily detect and locate the faults. However, they are not always detected by the traditional fault models.

In our work, we target these challenges and propose novel physical design-aware path delay test methods to deal with delay faults coming from manufacturing defects or physical design issues. They include the investigation of path delay variations in the presence of crosstalk noise, power supply noise, ground bounce and process variations. Based on this, we develop technology independent test methods for identifying the test patterns that may cause a worst-case delay on a target path. Then, we develop a dedicated test pattern generation method for path delay testing in the presence of crosstalk noise, power supply noise and ground noise. The proposed methods can be used to characterize the path speed and it helps to address the speed binning problem. Also, they can be employed in improving the classical ATPG approach of pattern generation. The application of these contributions can bring tremendous improvements to the IC test quality by ensuring better defect coverage and for an increased manufacturing yield during speed binning of IC chips.

Acknowledgements

This PhD thesis is a part of the three-year research work that has been carried out in SYSMIC (TRAFIC) research group of the Laboratory of Informatics, Robotics and Microelectronics of Montpellier (LIRMM). During this time, I have been supported by many people who actively contributed to this present work. With immense pleasure, I take this opportunity to convey my gratitude to each one of them.

I am very grateful to Pr. Patrick Girard and Pr. Serge Pravossoudovitch for believing and giving me this opportunity to become part of this PhD work. My deep gratitude and appreciation also goes to Dr. Aida Todri-Sanial for her consistent guidance in the initial part of this thesis. I would also like to thank my thesis advisors, Dr. Arnaud Virazel and Dr. Alberto Bosio for supervising, continuously supporting and guiding me in the rest part of my research. My sincere thanks to their vibrant efforts and skillful leaderships for attaining this thesis in its present form. Without them, I wouldn't have made so far. Thank you very much for the trust and confidence that you have given to me.

I would like to acknowledge the University of Montpellier and French National Centre for Scientific Research (CNRS) for providing the financial support of this PhD research.

I would like to acknowledge my thesis jury members Dr. Emmanuel Simeu, Pr. Matteo Sonza Reorda, Dr. Philippe Debaud and Pr. Olivier Sentieys for their valuable advice and suggestions to this work. I appreciate them for giving their time to review and comment on this thesis.

Moving towards more personal acknowledgments, I would like to execute a big thanks towards all my family members, my husband and to all my friends[1]. I am proud of the warm friendships from my colleagues and friends Alejandro, Syhem, Mohammad, Mariam, Saif, Khalid, Amit, Ioana, Virginie, Martine, Seza, Imran, Sophiane, Rida and Aymen. Also, from the Indian friends in Montpellier Sreedhar, Sourav, Jija, Rechan, Vishnu and

[1]You know who you are.

Vidhya throughout this period has been invaluable.

I am, of course, particularly indebted to my mother, my father and my sister for their monumental, unwavering support and encouragement on all fronts. They were always been there for me and without them none of this would have been even remotely possible. Finally, I would like to express my gratitude to my loving husband Suri for his patience, understanding and encouraging me throughout this thesis. The past 20 months being far from each other after our marriage has been a long adventurous happy journey. Thank you for your love, sacrifice and for our little cutie you have given me.

Montpellier *Anu ASOKAN*

December 2015

Contents

List of Figures

List of Tables

List of Abbreviations

ATPG Automatic Test Pattern Generation

CMOS Complementary Metal Oxide Semiconductor

DFM Design For Manufacturing

DFT Design-For-Test

DFM Design For Yield

FF Flip-Flop

GB Ground Bounce

IC Integrated Circuit

ITRS International Technology Roadmap for Semiconductors

PDAPG Physical Design Aware Pattern Generation

PDF Path Delay Fault

PI Power Integrity

PI's Primary Inputs

PO's Primary Outputs

PSN Power Supply Noise

PV Process Variation

RTL Register Transfer Logic

SE Scan Enable

SI Signal Integrity

SI's Scan-in Inputs

SO's Scan-out Outputs

SPICE Simulation Program with Integrated Circuit Emphasis

Xtalk Crosstalk

Introduction

As predicted by "Moore's law" [2], the integrated circuit (IC) products became denser with increased functionality and aggressive CMOS device scaling. Subsequently, with the arrival of "More Moore" scaling, geometrical shrinking of physical feature size were attained. Currently, "beyond CMOS" technologies are under development to satisfy the competitive consumer market demands. The entire device scaling evolutionary process has driven huge changes in our everyday life. Obvious applications are in medical field, aerospace, consumer products etc., and researches are still ongoing to make it more and more compact. International Technology Roadmap for Semiconductors (ITRS) has predicted that the growth of CMOS device scaling would slowdown and the progress would reach saturation. In the coming years, a functional diversification approach "More-than-Moore" is expected to flourish [3] with higher efficiency, additional functionality and increased complexity. Well, whatever the technology advancements, the plethora of physical design issues in IC's and manufacturing process imperfectness associated with the current and upcoming technologies is still not resolved. Accordingly, it is important to develop test techniques that can increase the manufacturing yield, along with any progressing technology scaling.

The denser, complex and fast switching circuits impose significant challenges to IC's, even though the remarkable scaling has assisted technology demands very well. They pose challenges to the signal, power and thermal integrity of circuits, as well as, reliability issues. Examples are crosstalk–induced delay, logic errors and substrate coupling. Excessive and varying voltage drop in the power supply and ground networks considerably affect the power integrity of a design. Also, variations in the IC manufacturing process, commonly called as process variations affect the reliability of the circuits. All of their impacts can give rise to distributed delay variations that may affect the functioning and performance of IC's. A new class of "delay" faults caused by the above mentioned delay variations needs to be properly addressed, modelled and used during fault simulation, test

generation, DFT, DFM and DFY.

"What is not testable is not fixable" [4].

An IC must be testable, not only manufacturable. Also, must reach a good yield figure, which makes it cost-effective. Along with the IC designs, DFT techniques were implemented to apply manufacturing tests. Different tests has been widely adopted in industry to detect delay-related defects. Performing at-speed delay test using path delay fault (PDF) model validates that all the delay defects are captured during IC testing phase. But some of the path delay faults escape the test due to the ineffectiveness in the applied test vectors. Path delay estimations must involve the gate models and the interconnect models for an accurate estimation. The PDF models used are based on the basic simplistic models at the gate level and they do not consider circuit physical design data (package, power/ground network parasitics, pad/pin location and cell placements). Therefore, PDF models generate ineffective vector pairs. They can only be fixed by testing the IC with the PDF models generated at physical design level rather than at gate level. The problem of testing IC in the presence of various issues at physical design level is also of major concern, as their occurence is not consistent. Path delay fault testing in the presence of crosstalk noise, power supply noise, ground bounce and process variations is examined in this work in accordance with their impact to a circuit path. Other research works in this field were dedicated towards their individual impacts and the ways to mitigate these effects; their combined physical design impacts were not considered. This work focusses on modifying the selection of patterns generated by the testing tools to accommodate the exact patterns that are identified for causing path delay variations in the circuit.

The overall outline of the thesis "Signal Integrity-Aware Pattern Generation for Delay Testing" is shown in Figure 1. This research work includes an introduction, four chapters and finally conclusion and perspective are discussed. The brief description of the thesis chapters are as follows.

Chapter 1 ellaborates the state-of-the-art related to this thesis work. The first part in this chapter discusses, the problems assoiciated with device scaling, i.e., signal integrity issues such as crosstalk noise, power integrity issues such as supply noise and reliability issues such as process variations. In the second part, delay testing principle, types of delay testing, ATPG and DFT are detailed. And, the final part provides an overview of the prior work to the existing solutions for physical design issues and pattern generation.

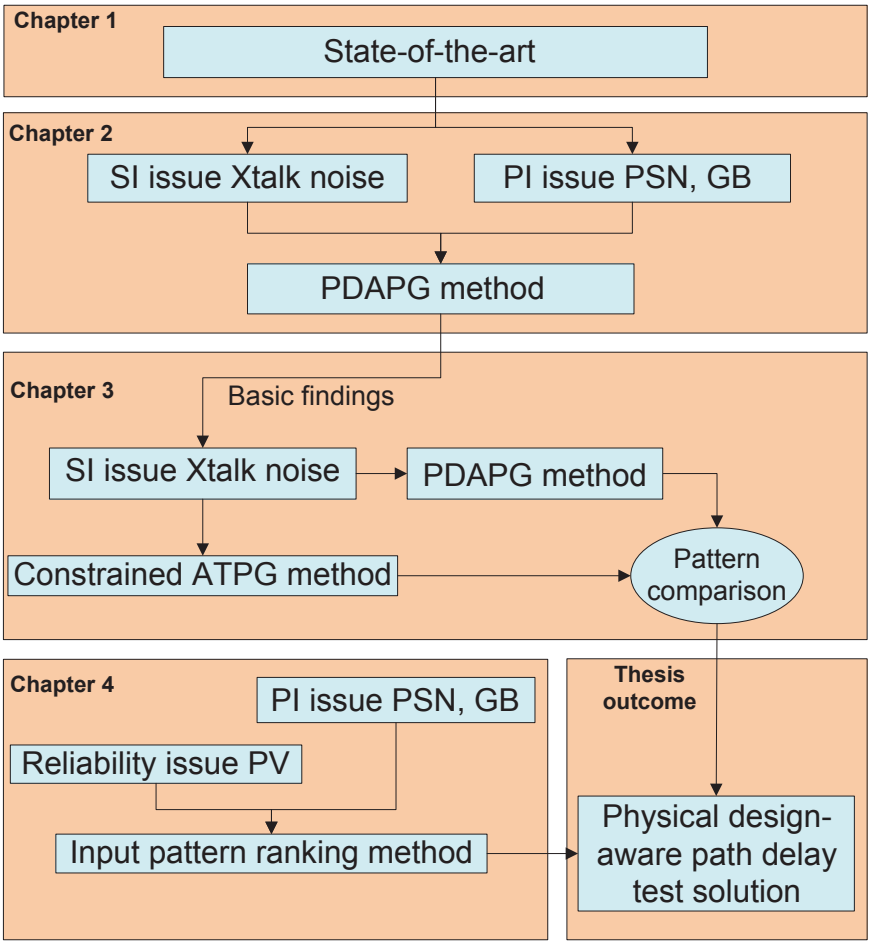


Figure 1: Thesis outline

Thereby, we show the context and motivation of our research work.

Chapter 2 presents our study on path delay variations in the presence of physical design issues such as crosstalk noise, power supply noise and ground bounce. The aim is to identify a worst-case path delay pattern that can predetermine delay defects. Then, we show that a path delay fault pattern generated by an ATPG tool does not match the pattern identified, hence indicating a discrepancy in the existing PDF model to detect delay defects. A physical design aware pattern generation method based on selective SPICE simulations is proposed to automate the different steps in pattern generation. Our major contributions in this method are the identification of aggressor nets and X-filling by backtrace approach. Our method is verified and results on ITC'99 benchmark circuits are provided.

Chapter 3 focuses on generating a worst-case path delay pattern in the presence of

crosstalk noise. Our objective in this chapter is to eliminate the selective SPICE simulation, which is exhaustive for bigger circuits. The basic findings from chapter 2 is utilized in the constrained ATPG method. This method is capable of modifying the selection of patterns generated by testing tools to accommodate the exact patterns that are identified as causing path delay variations in the physical design of a circuit layout. The proposed method is able to generate a worst-case path delay pattern in lesser computational time. The results are verified on ITC'99 benchmark circuits.

Chapter 4 deals with path delay variations in the combined presence of process variations and supply noise in circuits. By using our proposed metric for probabilistic pattern ranking method, we are able to identify the patterns that can capture the worst-case path delay in their combined presence. This metric aims at detecting the most-effective pattern for path delay testing from the subset of all input patterns. An input pattern ranking method is described based on the mean delay difference and the area of the delay probability distribution for all input patterns.

Chapter 5 includes concluding remarks of this thesis work and future research directions.

Chapter 1

State-of-the-art

Contents

With continually shrinking nanoscale technologies, the increase in manufacturing defects causes difficulties in developing a reliable semiconductor device. Usually, these devices are expected to meet the target specification for a wide range of operating conditions for different on-chip activity levels, voltage distributions in the power supply and ground networks, parametric variations etc. But with critical transistor sizes approaching only tens of atoms and the process engineers can no longer control the manufacturing processes as accurate as needed, the conventional large design margin allocation method to mitigate the impact of faults and variations hit the diminishing rate of returns. Furthermore, the rapid shrinking of the area-performance-power budget along with the increase

in process induced variations and time-dependent transistor parameters shift makes this design approach not anymore applicable [5].

In sub-45nm technology nodes, there is always a change in the nature of different physical design effects and reliability effects causing abrupt functional problems to a progressive degradation of the performance characteristics of devices and system components. There is also an increase in the occurrence of both permanent and transient faults, as well as, timing errors due to spatial, temporal and dynamic variations. New failure mechanisms that are not covered by current fault models are observed in designs fabricated in new technologies and new materials. At the same time, the power and signal integrity issues that come with scaled supply voltages and higher operating frequencies increase the number of faults that violate the pre-defined timing margin. Therefore, testing has become more and more important and challenging to verify the correctness of design and manufacturing processes [6]. The effective way to deal with these problems is by devising better Design-for-test (DFT) methods and capturing defects in the integrated circuits during the testing phase. This approach helps to minimize the defective parts per million (DPPM) of integrated circuit's being manufactured.

1.1 Background

Scaling of CMOS structures into the nanometer regime has posed new challenges to the physical design and reliability of circuits. The reasons for this are manifold [7], [8], [9]: (1) Manufacturing structures much smaller than the wavelength of the light used in modern lithography is difficult and can be practically done only in certain coarse limits. (2) As going closer to the dimensions of atoms, the actual location of doping atoms has an effect on the properties of transistors. (3) the structures are closer to each other, resulting in even smaller impurities or metal silvers to create shorts or other defects. Furthermore, the smaller proximity of a circuit structure to another makes the noise environment worse. (4) As the number of transistors, wires, contacts and vias on a single chip increase, the probability of one or more of being faulty increases.

It is evident that many of these new problems are not visible at the design phase, and thus, cannot be handled by discarding faulty chips after manufacturing and testing them. Since the faults can become visible only at runtime, so they have to be tested with patterns

that can detect the faults at runtime. Signal integrity issues such as crosstalk noise, power supply noise and ground bounce, as well as manufacturing process defects due to process variations cause variations in the path delay estimation. These variations are pattern dependent. Early stage prognosis of input patterns can give a better estimate of path delay in a circuit that may affect the performance. Different delay fault models that are currently employed by Automatic Test Pattern Generation (ATPG) tools for capturing the delay defects are path delay faults and transition delay fault models. Among these, the path delay fault model can be utilized for testing failures due to crosstalk noise. The impact of the physical design issue, such as crosstalk noise can cause a worst-case delay on any victim path. They give rise to delay faults. Therefore, it is essential to predetermine patterns that can capture crosstalk-related delay fault.

ATPG tools are not well versed to capture the faults or defects as they are based on simple logic models and also supply noise or manufacturing variations may take more time to capture a fault. Therefore, testing with patterns generated by the existing and commercially available ATPG tools [10] may create low-quality products. This motivates us to examine the issues such as crosstalk noise, supply noise and process variations and include their impact during pattern generation in the existing ATPG methods. These methods can be also used for increasing the manufacturing yield and reduce the rate of returns. Chips that have faults can still be used; at least in some low-reliability applications.

1.2 Challenges in Nanoscale Technologies

1.2.1 Crosstalk

Crosstalk is a disturbance in a signal interconnect (i.e., victim net, a signal propagation net of interest) induced by a sudden change in voltage by an aggressor interconnect (i.e., the neighboring nets). It has a direct impact on the signal traveling through the interconnects in the chip. Crosstalk continues to be a major design issue due to interconnects becoming taller, narrower, and more closely spaced with each new technology node. As a result, the interconnect sidewalls become prime locations for coupling while the parasitic load capacitance of the interconnect itself becomes smaller, allowing the coupling capacitance to become more dominant. This has also been a major concern in design verification and timing analysis for many years [11] [12] [13].

During a test, the ATPG tool does not consider which nets are causing it to switch simultaneously. As a result, it is possible to fortuitously exercise some aggressors to increase the amount of crosstalk and push the victim path closer to the timing closure limits calculated during verification. At the same time, it is also possible that the tool will not generate any aggressor switching and the victim path will not be anywhere close to those timing limits. As a result, if a timing related fault were to fall on that path, it is possible the latter case could occur and the fault would go undetected, potentially resulting in higher DPPM.

Several techniques have been proposed to deal with crosstalk issues during verification and test. Crosstalk verification with interconnect process variation is discussed in [14]. The authors in [15] present fault modeling called maximum aggressor (MA) and simulation for crosstalk on SOC interconnects. Other techniques have focused on similar approaches to maximize crosstalk [16] [17] [18]. Further investigations have shown that the MA model may not always ensure highest crosstalk effects [19] especially when mutual inductance effects are considered in addition to the timing of transitions. Additionally, many of these approaches require new ATPG algorithms, hindering easy adoption in the industry. Some researchers have proposed using on-chip sensors or glitch/delay detectors [20] to detect noise and delay violations. The drawback of such techniques is that the sensors must be tuned and very accurate and adding one sensor per interconnect is prohibitively expensive. Due to their high sensitivity to voltage and timing, process variation can also negatively impact their operation. The authors in [21] utilize the boundary scan cells to generate test patterns to detect noise and delay violations on a system chip and observe the responses which are then scanned out using boundary scan shift procedure. Most of the proposed techniques, mentioned above, target only buses or interconnects between cores in a SOC rather than internal paths. Authors in [22] propose validation and test generation for crosstalk-induced delay and noise for SOC interconnects. An analytical model for crosstalk was developed in [23] and used as a basis for pattern generation to induce delay due to crosstalk in [24]. However, this approach only generates patterns for a single aggressor affecting a target path. The procedure proposed in [25] considers a genetic algorithm based approach when inducing crosstalk into delay test patterns. There have also been proposed academic ATPGs that considers crosstalk and transition arrival times during pattern generation [24] [22] but it lacks the immediate use in practice since they

are computationally intensive and would require a significant change to modern ATPG algorithms and models.

We have also summarized below the existing techniques and approaches that were proposed to deal with the different aspects of crosstalk noise in the recent years.

- Accurate crosstalk noise models [26] [11] [15] [27] [12] [13] [28];
- Timing analysis in presence of crosstalk noise [29] [15] ;
- Test generation to maximize crosstalk noise [30] [24] [31] [32] [25] [22] [33] [34] [17];
- Timing defect diagnosis in presence of crosstalk [35].

1.2.2 Power Supply Noise

While Crosstalk directly impacts the signal paths, Power supply noise (PSN) indirectly impacts the data traveling through the signal paths by affecting the transistor drive currents. PSN is a disturbance in the power distribution network that can momentarily change the voltage difference between the power and ground rails of the functional logic. Power distribution network design has become a significant obstacle as process nodes become smaller and smaller [36]. Noise on the power supply can come from four potential sources: IR-drop, L di/dt noise, LC resonance, and electromigration [37]. Just from a first-order perspective (IR), as each new process node pushes wire widths smaller, the resistance of these wires becomes greater and as functional density increases, current demand increases. Incorrect design of the network can cause supply starvation to a portion of the chip and it will never work on actual silicon.

The power distribution network is usually very customized for the design based on functional power needs [38] [39] and the type of chip packaging [40]. The functional power is typically estimated early in the design process based on both static and dynamic IR-drop needs. The static power is now dominated by transistor leakage due to the low voltage thresholds used by modern process nodes. The dynamic power is based on the expected average transient current over a period of time (usually the clock period). While this can range from 10%-20% for many designs, the switching activity can be much higher when the design is dominated by buses, but this can often be the exception rather than the rule.

Test power, in the meantime, can cause much greater switching activity than the 10%-20% range [41]. As a result, the IR-drop experienced during the test can be much greater since the power distribution network was not designed to handle such current demands. This is due to the ability to scan in any state into the design using the scan chains.

There is an extensive list of literature on the topic power supply noise related to test [42]. In [43], the authors propose a low-capture-power (LCP) X-filling method for assigning 0's and l's to the X-bits in a test cube so that the number of transitions at the outputs of scan flip-flops in capture mode for the resulting fully-specified test vector is reduced. The authors in [44] propose another method, called capture-aware (CA) test cube generation, for deterministically generating test cubes not only for fault detection but also for capture power reduction. The preferred fill technique proposed in [45] attempts to reduce the Hamming distance between the initialized, launched, and captured patterns during TDF testing. In [46], a pattern generation technique was proposed to create maximum supply noise to increase the delay along targeted paths. Neural network and genetic algorithm based solutions were proposed in [47]. Also, a method of measuring average power called switching cycle average power (SCAP) was used to produce supply noise tolerant patterns [48]. SCAP considers both simultaneous switching and which of the long paths in the design are affected. The method requires pattern delay information which may make it computationally intensive. There have also been more precise techniques that perform RLC analysis for pattern generation [49]. However, the extensive analysis required can become time-consuming as these networks become more complicated in modern designs. Vector-based compaction solutions to reduce overkill and power supply noise induced delay have been proposed in [50]. The authors developed a vector-dependent power supply noise analysis solution that models the voltage drop based on the layout of the chip. However, simulation of each compacted pattern to estimate IR-drop can result in significant run times. Also, their proposed approach does not consider areas of the chip that may be underutilized. There has also been a pattern post-processing technique to verify whether patterns generated will cause an excessive IR-drop [51]. Again, these approaches attempt to mitigate overkill rather than focus on underkill. Methods to modify conventional ATPG algorithms have also been proposed [52].

While most approaches to reduce power during test involve manipulation the test pat-

terns, there have also been proposed some approaches that involve integrating additional circuitry or logic to control which scan chains can be enabled or portions of functional logic can capture [53]. While effective since it disables whole sections of the design, the amount of additional logic is not considered trivial.

Power supply noise during the test is still an open research area, and many researchers are currently working on various aspects of the power supply noise problem [54] [55]. The EDA and semiconductor industry has a major interest in dealing with different PSN issues. Below is the summary of different methods and approaches that were proposed to deal with varying aspects of PSN-related issues:

- Accurate model for power and power supply noise [49] [56] [57] [58] [59] [60] ;
- Power distribution network analysis [61] [62] [63] [36] [37] ;
- Timing analysis in the presence of supply noise [64] [65] [66] [58] [67] [46] ;
- Low-power scan-based test generation and application [68] [69] [70] ;
- Supply-noise aware delay test pattern generation [71] [58] [57] ;
- Test compaction considering IR drop [71] [60] ;
- Worst-case PSN analysis at core and system levels [72] [73] .

1.2.3 Process Variations

Besides crosstalk and supply noise, this thesis is also concerned regarding the impact of process variations on path delay of a circuit. Practically, it is difficult to achieve the fabricated transistor parameters similar to the design specification. In reality, the parameters are different from die-to-die, wafer-to-wafer, and lot-to-lot, and also between transistors on the same die. Parameter variations are caused by differences in impurity concentration densities, oxide thicknesses, diffusion depths, and similar factors. These nonuniform conditions result in deviations in transistor parameters, such as threshold voltage, W/L ratio, as well as variation in the widths of interconnect wires [74]. The process variations are expected to impact design performance (increase or decrease gate and interconnect delays) to a large extent in newer technologies. Thus, it is necessary to take these effects into consideration along with the crosstalk noise and PSN issues during

pattern generation procedures. Summary of the previous works in this area are discussed in Chapter 4.

In the next section, we will show the importance of testing a circuit design and the various existing test methods that can ensure better quality of IC's being manufactured.

1.3 Testing

The goal of manufacturing test is to detect any defect that occurs in the fabricated circuits. Ideally, we can differentiate the faulty circuits and fault-free circuits after the manufacturing test. Figure 1.1 illustrates the basic principle of chip testing. Test vectors are applied to the inputs of the circuit-under-test (CUT), and the responses are collected and compared with the expected values. If the responses match, the circuit is considered good. Otherwise, it is considered bad. The automatic test equipment (ATE) is used to test chips. It is obvious that the test quality depends upon the thoroughness of the test vectors. However, the test quality and test cost are interdependent. A large number of test vectors/patterns may result in a good test quality, but it will increase the test time and test cost at the same time.

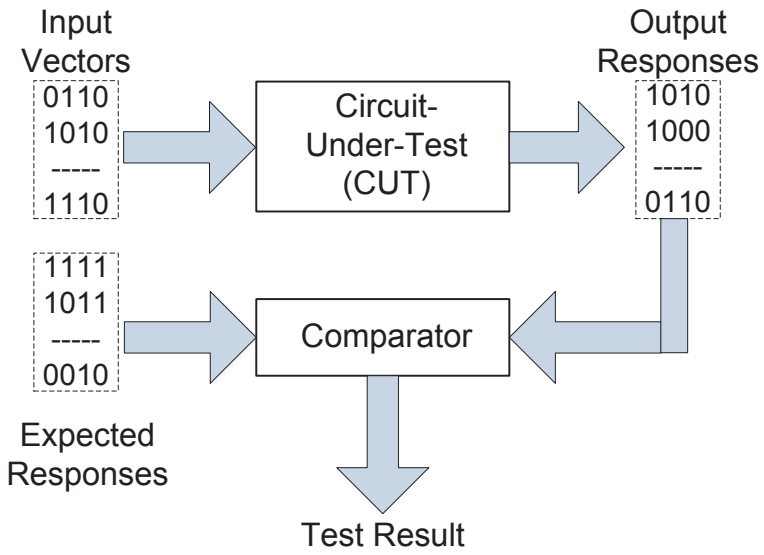


Figure 1.1: Principle behind testing chips

1.3.1 Different Types of Manufacturing tests

Chip vendors have a variety of different test methods at their disposal, which are often used in some combination to balance costs. It is usually left to the vendor's discretion which tests to apply and the amount of coverage they need to achieve to satisfy the customers' needs while also containing their own test costs. While there are not a set of standardized tests each vendor must follow there are still common practices amongst most vendors; allowing customers to switch to a different vendor without risking a need to make a dramatic change in the way quality of the shipped parts is measured.

- **Parametric Testing**

Parametric testing is typically used to check the electrical properties of the device and can be used to characterize any potential systematic issues with the process node. These tests may not check any functionality of the device but can find gross shorts, opens, leakage issues, or current drive problems.

Often specialized circuitry is added for the purpose of parametric testing. Adding ring oscillators to both the wafer scribe lines and to the die itself is a common practice to perform propagation delay tests, setup and hold tests, and speed testing. The most common technique is to employ an on-die chip measurement circuitry to monitor supply droop and adaptively scale it depending on its usage. It is possible to leverage the use of these measurement circuits during parametric testing to get a more granular reading than what can be achieved with the ATE alone.

- **Functional Testing**

While most often used for design verification, functional testing can also be used for manufacturing test. It is considered very time-consuming and very expensive due to the exhaustive nature of applying test patterns to cover each known reachable functional state. Expert design knowledge is usually required and little in the way of design automation tools can be used to assist in the effort. Usually, an ad-hoc approach is required for each design, so unless the next design architecture is based very heavily on the previous design, it will not be possible to reuse the assets from the previous design.

• **Structural Testing**

Structural testing, which will be the backbone of this presented thesis work, allows state observation of circuit behavior from relatively few observation points [7]. Unlike functional testing, it does not require enumeration of all functional states to test the design, so test volumes are not as large. Additionally, structural tests do not require the same expert design-specific knowledge as functional testing and algorithms can be utilized to create automatic test pattern generation (ATPG) tools that can be leveraged to create the patterns. So, even if there is a little design re-use from one design to the next, there can still be a re-use of the same automation tools, which significantly reduces the cost of test. Due to the lower test volumes and tool reuse, structural testing has seen widespread adoption across the semiconductor industry. A closer look at how structural testing is used to find defective chips is detailed in the next section.

1.3.2 Structural Fault models

There are three terms that are usually used to describe the incorrectness of an electronic system.

• **Defect**

A defect in an electronic system is the unintended difference between the implemented hardware and its intended design. Typical defects in VLSI chips are: process defects, material defects, aging defects and package defects.

• **Error**

A wrong output signal produced by a defective system is called an error. An error is an effect whose cause is some "defect".

• **Fault**

A representation of a "defect" at the abstract functional level is called a fault.

A fault model is a mathematical description of how a defect alters the design behavior. A fault is said to be detected by a test pattern if, when applying the pattern to the design, any logic value observed in one or more of the circuit's primary output's differs between

the original design and the design with the fault. There are a lot of fault models developed to describe different kinds of physical defects. The most common fault modes for modern VLSI test includes: stuck-at fault, bridging fault, delay faults (transition delay fault and path delay fault), stuck-open and stuck-short faults, etc.

• **Stuck-at faults**

A signal, which is an input or an output of a logic gate or flip-flop is stuck at a 0 or 1 value, independent of the inputs to the circuit. The single stuck-at fault is widely used, i.e. two faults per line, stuck-at-1 (sa1) and stuck-at-0 (sa0).

• **Bridging faults**

Two signals are connected together when they should not be. Depending on the logic circuitry employed, this may result in a wired-OR or wired-AND logic function. Since there are O(n2) potential bridging faults, they are normally restricted to signals that are physically adjacent in the design.

• **Delay faults**

These faults make the signals to propagate slower than normal, and cause the combinational delay of a circuit to exceed clock period. Specific delay faults are: transition delay faults(TDF), path delay faults (PDF), gate delay faults, line delay faults, segment delay faults. Among them, slow-to-rise and slow-to-fall PDF and TDF are the most commonly used ones. Path delay fault model targets the cumulative delay through the entire list of gates in a path while the transition fault model targets each gate output in the design.

• **Stuck-open and Stuck-short faults**

A CMOS transistor is considered as an ideal switch. Stuck-open and stuck-short faults model the switch being permanently in either the open or the shorted state. And they assume just one transistor to be stuck-open or stuck short.The effect of a stuck-open fault is a floating state at the output of the faulty logic gate. It can be detected in a similar way as detecting a stuck-at fault at the output fault on the gate's output pin. The effect of stuck-short fault is that the short connects power line and the ground line. So quiescent current (IDDQ) measurement can be used to detect such fault.

1.3.3 Fault Categories

An ATPG tool maintains a list of potential faults in the design and assigns each such fault to a fault class according to its detectability status. Faults classes are organized into categories.

There are 5 higher-level fault categories containing a total of 17 lower-level fault classes:

1.3.3.1 Detected Faults

The DT (detected) category of faults includes four classes:

- **DR (detected robustly)**

The faults that are determined during Path Delay ATPG or fault simulation. During ATPG, at least one pattern that caused the fault to be placed in this class is retained.

- **DS (detected by simulation)**

The faults that are determined by generating patterns and simulating to verify that the patterns result in the faults being detected.

- **DI (detected by implication)**

The faults that do not have to be detected by specific patterns, because these faults result from shifting scan chains. The faults in the DI class usually occur along the scan chain paths and include clock pins and scan-data inputs and outputs of the scan cells.

- **D2**

The faults that are clock faults detected when the loadable non-scan cell faulty value is set to both 0 and 1. Note that the loadable non-scan cells feature must be active.

1.3.3.2 Possibly Detected Faults

The PT (possibly detected) category of faults contains the following four classes:

- **AP (ATPG possibly detected)**

This class contains faults for which the difference between the good machine and the faulty machine results in a simulated output of X rather than 1 or 0. Analysis proved that the fault cannot be definitely detected under current ATPG conditions, only possibly detected.

- **NP (not analyzed-possibly detected)**

This class also contains faults for which the difference between the good machine and the faulty machine results in a simulated output of X rather than 1 or 0. However, the analysis to prove that the fault cannot be definitely detected using current ATPG conditions was not conclusive. Like the AP class, the simulation cannot tell the expected output of the faulty machine.

- **P0**

This class contains clock faults when the loadable non-scan cell faulty value is set to 0. Note that the loadable non-scan cells feature must be active.

- **P1**

This class contains clock faults when the loadable non-scan cell faulty value is set to 1. Note that the loadable non-scan cells feature must be active.

1.3.3.3 Undetectable Faults

The UD (undetectable)] category of faults contains faults that cannot be tested by any means: ATPG, functional, parametric, or otherwise. Usually, when an ATPG tool calculates test coverage, these faults are subtracted from the total faults of the design.

The UD category includes four classes:

- **UU (undetectable unused)**

This class contains faults located on unused outputs or, in general, outputs that have no electrical connection to any other logic. A fault located on one of these fault sites has no logic simulation effect on any other logic in the design.

- **UO (undetectable unobservable)**

This class is similar to the UU (Undetectable Unused) class, except that the UO fault class specifically includes faults on unused gates with fanout (i.e., gates connected to other unused gates). Faults on unused gates without fanout are identified as UU faults.

- **UT (undetectable tied)**

This class contains faults located on pins that are tied to a logic 0 or 1, which are usually unused inputs that have been tied off. A stuck-at-1 fault on a pin tied to a logic 1 cannot be detected and has no fault effect on the circuit. Similarly, a stuck-at-0 fault on a pin tied to a logic 0 has no effect.

- **UB (undetectable blocked)**

This class contains faults at locations for which controllability and observability are hindered by redundant faults. UB faults are companion faults to UR faults.

- **UR (undetectable redundant)**

This class contains faults for which there are redundant logic paths. A fault in one path cannot be detected because the other redundant logic path masks the fault effect. Because of the self-protecting nature of redundant logic design, a single fault cannot be detected and is indistinguishable from the good machine behavior as seen from the design outputs and scan cells.

1.3.3.4 ATPG Untestable Faults

The AU (ATPG untestable) category of faults contains faults that are not necessarily intrinsically untestable, but are untestable using ATPG methods. These faults cannot be proven to be undetectable and might be testable using other methods (for example, functional tests).

The AU category includes two classes:

- **AN (ATPG untestable-not detected)**

A fault cannot be controlled or observed because setting up the required patterns would violate a PI or ATPG constraint. Faults associated with non-scan sequential devices (latches and flip-flops) are not testable with simple ATPG methods.

• **AX (ATPG untestable-timing exceptions)**

This classification occurs under the following circumstances: For each fault affected by timing exceptions, if all the gates in both the backward and forward logic cones are part of the same timing exception simulation path, then the fault is marked AX. This analysis finds the effects of setup exceptions, so it does not impact exceptions that are applied only to hold time.

1.3.3.5 Not Detected Faults

An ATPG classifies a fault as ND (not detected) when the analysis for that fault was not completed or was aborted. Incomplete or aborted analysis could be caused by the default ATPG iteration limits or by designs that are too complex for the ATPG algorithm to solve.

The ND category has two classes:

• **NC (not controlled)**

This class contains faults that the ATPG algorithm could not control to achieve both a logic 0 and a logic 1 state. Nodes that are always at an X state are classified as NC because ATPG cannot achieve either a logic 0 or a logic 1.

• **NO (not observed)**

This class contains faults that could be controlled, but could not be propagated into a scan chain cell or to a design output for observation.

1.3.4 Automatic Test Pattern Generation

ATPG tools have been a great benefit to test engineers. Since they take advantage of structural fault models to generate patterns, the same tool can be used in many different designs with very different architectures. These tools are widely available to industry whether it is commercially [10] or from academia.

The goal of these tools is to deterministically generate patterns that will stimulate a fault site and propagate the potential fault effect to an observation point to identify as many faults as possible with as few patterns as possible. While not all faults will be

testable due to the topology of the circuit, the ATPG will be able to generate patterns for those faults that are testable. The ability to test these fault sites is measured by the fault coverage, which is a fraction of the number of faults detected after pattern generation over the total number of faults in the design, as shown in Equation 1.1, where FC is the fault coverage percentage, DT is the total number of detected faults, and TF is the total number of faults in the design. There is additionally another metric often used by these tools called test coverage. The test coverage, as calculated in Equation 1.2 considers those faults that are not testable either due to circuit redundancies or tied off signals that would make coverage of particular fault types impossible. TC is the test coverage percentage and UT are those faults that have been classified by the ATPG as untestable.

$$FC = \frac{DT}{TF} \times 100 \tag{1.1}$$

$$TC = \frac{DT}{TF - UT} \times 100 \tag{1.2}$$

While the fault coverage provides a metric of how much of the design has been covered with the generated pattern set, the test coverage essentially measures the effectiveness of the ATPG tool. These two values can also be used to help guide design issues since a large discrepancy between the two values could imply a potential problem in the design.

While each commercial vendor has their own proprietary algorithm for pattern and coverage optimizations, they all basically follow a similar flow as shown in Figure.1.2. First, a fault dictionary for the design has to be built. Next, the tools begin to deterministically target these faults one at a time. While the patterns are being generated, compaction of the patterns also occurs dynamically. This is possible since very few bits of the pattern are usually needed to stimulate and observe a specific fault site. After the pattern generation and compaction have filled in some of the pattern's with care-bits, the remaining don't care bits can be filled randomly or with some other fill scheme. The final fully specified pattern is then fault simulated for any fortuitous detection of additional faults.

The automatic test pattern generation (ATPG) is the process of automatically generating a set of test patterns for detecting a specific group of faults. The inputs of the ATPG procedure are design data (e.g., netlist), fault group (specifying what faults are

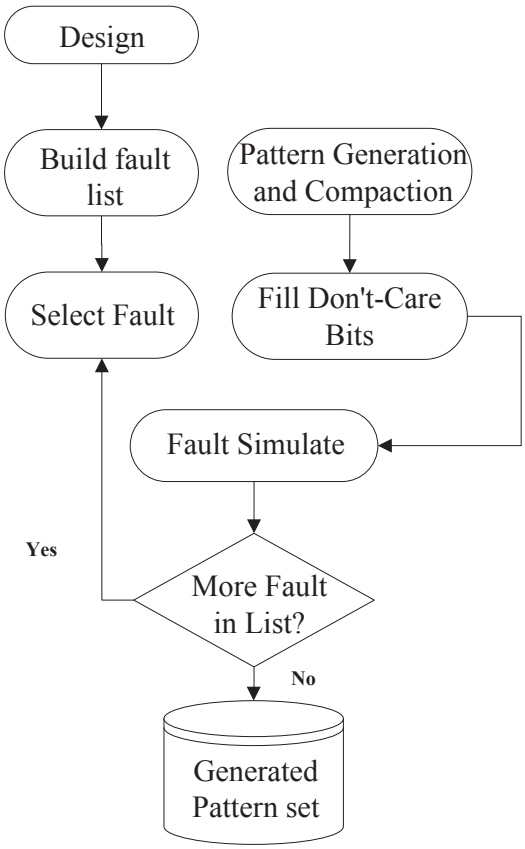


Figure 1.2: General ATPG Flow

targeted), test protocol and test constraints, and the output is a set of test patterns. The test patterns are then applied to the design for fault detection. If a fault can be detected by the input test patterns, it is called as a detected fault. Otherwise, it is an undetected fault. Note that there may be some faults in the design that cannot be detected by structural patterns, which are called undetectable faults.

ATPG algorithms inject a fault into the CUT, and then use a variety of mechanisms to activate the fault and propagate its effect to the circuit output. The output signal changes from the value expected for the fault-free circuit, and this causes the fault to be detected. There are various ATPG algorithms that can be found in the literature. Some of them are listed below:

- Roth's D-Algorithm (D-ALG) [75];
- Goel's PODEM algorithm [76];
- Fujiwara and Shimono's FAN algorithm [77];
- Kirkland and Mercer's dominator ATPG programs TOPS [78];

- Schulz's learning ATPG programs SOCRATES [79] [80][81];
- Giraldi and Bushnell´s EST methodology [82] [83][84];
- Kunz and Pradhan's Recursive Learning methodology [85] [86][87];
- Chakradhar's NNATPG algorithm family [88];
- BDD-Based ATPG Algorithms [89][90].

The following terms are important test generation definitions and are commonly used in ATPG literature:

- Controllability: A testability metric that measures how difficult it is to drive a node to a specific value.
- Observability: A testability metric that measures how difficult it is to propagate the value on a node to the primary output or scan flip-flop.
- Sensitization: The process of sensitizing the circuit and enable the fault to cause an actual erroneous value at the point of the fault.
- Propagation: The process of propagating error effects to the primary output or scan flip-flop.
- Justification: The process of finding the input combination required to drive an internal circuit node to a specified value.

All the above terms in this section are utilized in the coming chapters during the explanation of our methods to generate patterns.

1.4 DFT Techniques

Design-for-Testability (DFT) techniques are widely used in nowadays integrated circuits. "DFT" is a general term applied to design methods that lead to more thorough and less costly testing. In general, DFT is achieved by employing extra hardware circuits for test purpose. The extra test circuits provide improved access to internal circuit elements. Through these test circuits, the local internal state can be controlled and/or observed

more easily. It adds more controllability and observability to the internal circuits. DFT plays an important role in the development of test programs and as an interface for a test application and diagnostics. With appropriate DFT rules implemented, many benefits ensue from designing a system so that faults are easy to detect and locate. DFT can bring the many benefits. Generally, integrating DFT in the development cycle can help:

- Improve fault coverage
- Reduce test generation time
- Potentially shorten the test length and reduce test memory
- Reduce test application time
- Support hierarchical test
- Reduce life-cycle costs

These benefits come at the price of extra cost from pin overhead, more area and thus low yield, performance degradation and longer design time. However, since it reduces the overall costs of the chip, DFT is a cost-effective methodology and widely used in the semiconductor industry.

Nowadays DFT techniques have become even more critical in modern designs. Without it, it is only possible to apply test patterns to the primary inputs and observe the results at the primary outputs of the design. When sequential logic is included, covering all of the faults efficiently becomes much more difficult since it will likely take multiple clock pulses to activate the fault site from the primary input and propagate the result from the fault site to the primary output.

1.4.1 Scan Design

Scan design has been widely adopted across the industry due to the ability to achieve high fault coverage with relatively low overhead. A scan flip-flop is used in place of a conventional D flip-flops and is stitched together to form a shift register. As shown in Figure 1.3, a basic scan flip-flop is a combination of a D flip-flop with a multiplexer placed at the D input; the multiplexer is controlled by a new signal called scan-enable (SE).

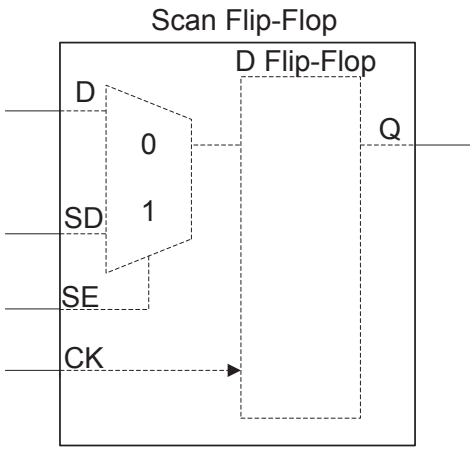


Figure 1.3: A scan flip-flop

The scan flip-flops are then stitched together, as shown in Figure 1.4, to form a shift register. The multiplexer selects between the functional input of the original D flip-flop and the output of the previous scan flip-flop in the newly formed scan chain. The functional input is selected when scan-enable is 0 and the scan path is selected when scan-enable is 1. While a new port will have to be added for the SE signal, the input and output ports can be shared as shown in Figure 1.4.

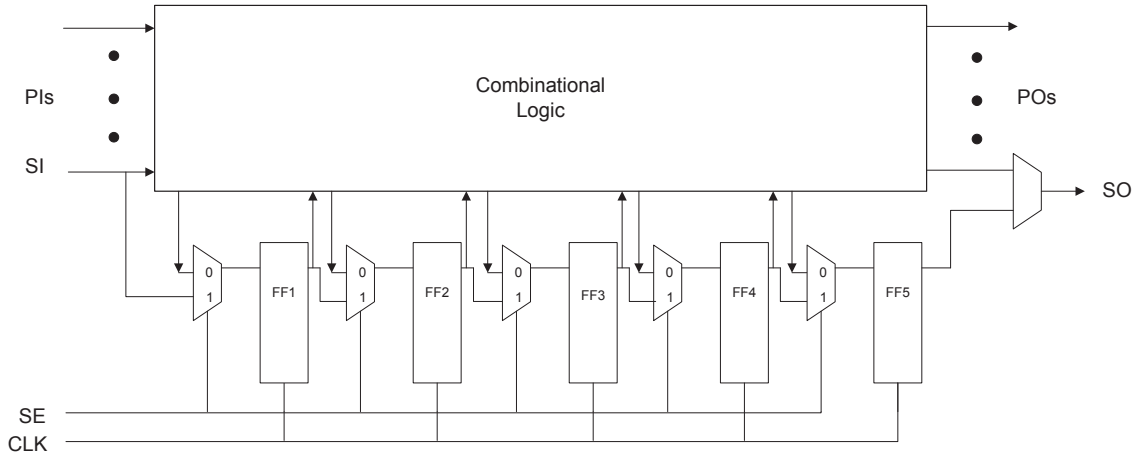


Figure 1.4: An example of scan-chains inserted design

The scan chain essentially grants full control and observability of all internal state logic through a single input and output port. This grants the ability to easily place the chip into any state perform a single clock, and scan out the result, turning a potentially complex design into a simple combinatorial circuit.

1.4.2 Different ATPG Modes

These are some of the different operational modes offered by an ATPG tool

• **Basic-Scan ATPG**

In Basic-Scan mode, ATPG can be operated in a full-scan, combinational only ATPG tool. To get high test coverage, the sequential elements need to be scan elements. Combinational ROM's can be used to gain coverage of circuitry in their shadows in this mode.

• **Fast-Sequential ATPG**

Fast-Sequential ATPG provides limited support for partial-scan designs. In this mode, multiple capture procedures are allowed between scan load and scan unload, allowing data to be propagated through non-scan sequential elements in the design such as functional latches, non-scan flops, and RAM's and ROM's. However, all clock and reset signals to these non-scan elements must still be directly controllable at the primary inputs of the device.

• **Full-Sequential ATPG**

Full-Sequential ATPG, like Fast-Sequential ATPG, supports multiple capture cycles between scan load and unload, thus increasing test coverage in partial-scan designs. Clock and reset signals to the non-scan elements do not need to be controllable at the primary inputs; and there is no specific limit on the number of capture cycles used between scan load and unload.

1.4.3 At-speed Test

In scan-based (as shown in Figure 1.4) at-speed test, input test patterns can be applied in two different manners: launch-off-capture (LOC), also called broadside testing [91] and launch-off-shift (LOS) [92]. While both techniques rely on the scan-based testing to initialize the circuit, the two techniques differ in the manner the data is launched during the first at-speed test cycle. While LOC relies on the functional path to stimulate transitions, LOS stimulates transitions through the shift path. This difference essentially means the scan-enable signal is 0 during the launch cycle for LOC, as shown in Figure

1.5, while it is 1 during the launch cycle for LOS, as shown in Figure 1.6. There have also been other DFT techniques that attempt to combine the two techniques that attempt to use the advantages of each technique while mitigating from the disadvantages [93].

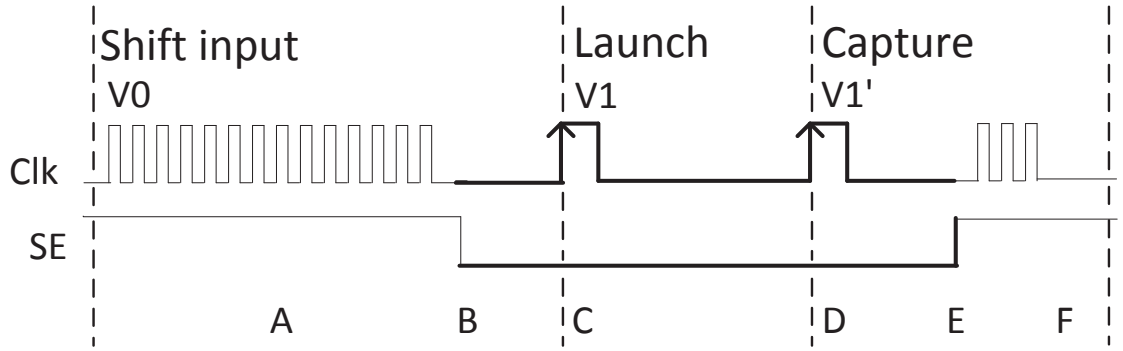


Figure 1.5: At-speed pattern generation using launch-off capture

In each figure, line A notes the last slow scan shift to set the pattern V0 into the scan chain. Line B marks the transition of the SE signal from 1 to 0. At line C, the first at-speed pulse launches the transition and changes the values of the flops to pattern V1. Line D denotes the capture of the functional response of pattern V1. The SE signal is restored to 1 at the same time for both LOC and LOS at line E. Finally, at F, shifting of the pattern begins again to observe V1' and start the application of a new pattern.

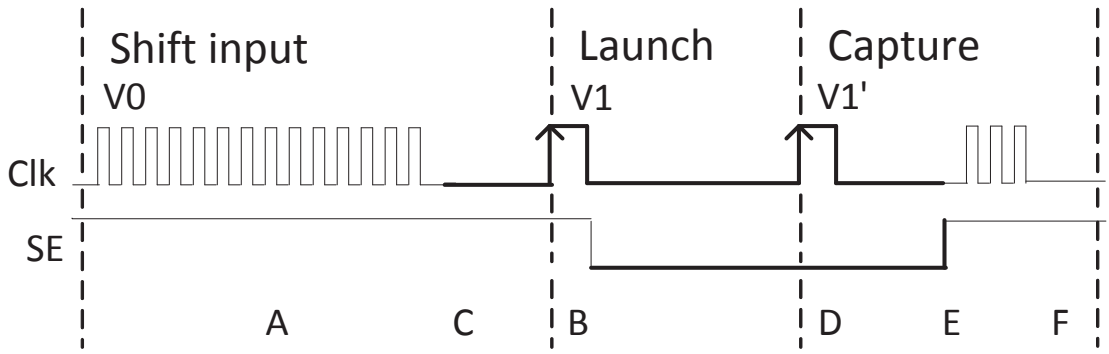


Figure 1.6: At-speed pattern generation using launch-off shift

Both LOC and LOS have advantages and disadvantages. While LOS provides higher fault coverage since there is a higher degree of control when stimulating the transition through the shift path, the scan-enable signal must be timing closed since it must be asserted during the launch cycle and deserted during the capture cycle. LOC does not have the same design constraints on scan-enable and is the easier of the two methods to

implement in physical design, but does not achieve the same fault coverage since transitions must generate through functional logic and become dependent on the functional behavior of the device.

1.4.4 At-Speed Delay Test Challenges

As circuit complexity and functional frequency increase, power integrity and timing integrity are becoming more and more important to circuit design and test. The test power consumption, supply voltage noise, crosstalk noise caused by signal coupling effect, and hot spots caused by nonuniform on-chip temperature will significantly impact yield and reliability. As shown in Figure 1.7 from [94], with shrinking technology node, the percentage of delay caused by coupling effect between signal lines (crosstalk noise) and IR-drop on power and ground lines (power supply noise) is taking a larger portion. Power supply noise and crosstalk noise are becoming two important noises that impact circuits' timing integrity. The lower supply rails in today's IC's mean much less immunity from signal integrity problems that tie directly into power integrity. Supply voltages on many high-end IC's are now down to 1V and below, leading to decreasing margins for voltage fluctuation. Simultaneously, switching noise can cause fluctuations in the ground voltage level, leading to difficult-to-isolate signal-integrity problems and timing issues. Power, timing, and signal integrity effects are all interdependent at 90-nanometers (nm) and below.

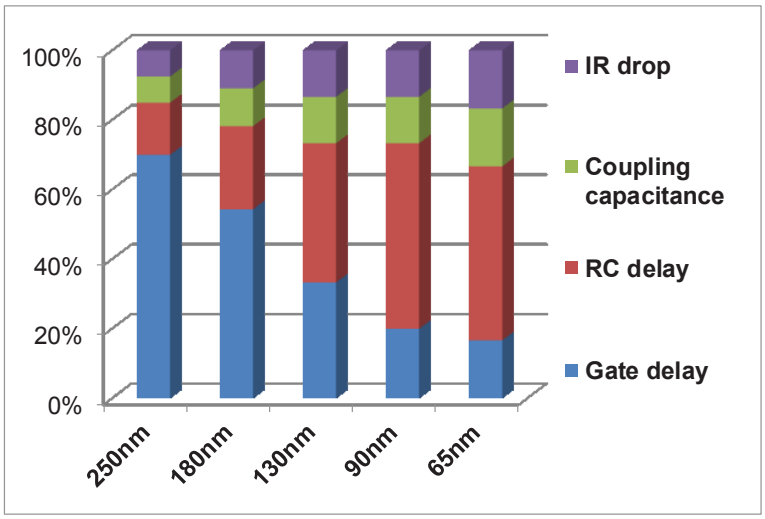


Figure 1.7: At nanometer process nodes, parasitic effects increase

Timing failures are often the result of a combination of weak points in a design and

silicon abnormalities, which reduce the noise immunity of the design and expose it to signal integrity issues. For example, a poor power planning or missing power via's can incur on-chip power droop for some test vectors. The power droop can impact a gate(s) on a critical path and it may cause timing failure. This failure may only be excited with certain test vectors as inputs. If the corresponding test vector is not included in the test pattern set, the failure becomes an escape and cannot be reproduced during diagnosis with the current test pattern set. Current automatic test pattern generation (ATPG) tools are not aware of the switching distribution on the layout and the pattern induced noises. There are escapes and "No Problem Found" (NPF) parts returned by customers, which have passed the tests using the layout-unaware test patterns generated by ATPG tools. Thus, high-quality test patterns are imperative which can be used to capture noise-induced delay problems during production test and identify noise-related failures during diagnosis. Test vector (pattern) generation considering the noise effect, mainly supply noise, crosstalk noise and process variations will be the focus of this thesis work.

Chapter 2

Path Delay Test in the Presence of Multi-Aggressor Crosstalk, Power Supply Noise and Ground Bounce

Contents

2.1 Introduction

With technology scaling, the varying impacts of physical design (PD) issues have become a major challenge in IC's. These PD may issues create undesired effects such as multi-aggressor crosstalk noise, power supply noise and ground bounce. PD issues are the processes that induce additional delays in any circuit path. Their effects cause considerable path delay variations, thereby degrading the circuit performance. Path delay test patterns generated by timing-aware Automatic Test Pattern Generation (ATPG) tools are commercially utilized to determine faults or errors caused by path delay variations. As these tools are not built to account for PD issues, identifying proper test patterns

to capture the worst-case path delay becomes difficult. Thus, by applying the existing path delay test techniques, might not necessarily identify the right set of test patterns. This may allow some faults to go undetected. Therefore, PD issues need to be considered during the path delay fault test to reduce delay fault escape and to ensure better path delay defect coverage.

Path delay variation usually occurs due to PD issues such as crosstalk (Xtalk) noise, Power Supply Noise (PSN), Ground Bounce (GB). Crosstalk is due to capacitive or inductive coupling between a victim interconnect (i.e., signal propagation net of interest) and an aggressor interconnect (i.e., the neighboring nets). Depending on the circuit's layout, one victim interconnect can have single or multiple aggressor interconnects. Their impact on a victim interconnects (or nets) delay varies depending on the behavior of multi-aggressors, such as their signal switching frequency, signal transitions and their arrival time. Also, their path delay estimation defers as it may expand to the entire victim path (consists of many victim nets) with distributed delay variations. Xtalk impacts may speedup or slowdown the worst-case path delay in a victim net [95]. Also, an opposite signal transition in the aggressor net maximizes the delay slowdown [96] of the respective victim net, when compared to the same signal transitions and stable inputs (i.e., stable 0 and stable 1). In our work, we determine all the aggressor nets from the physical design of the given circuit layout rather than from their gate level circuit netlist, as it would represent a real circuit structure that will be manufactured and tested.

Xtalk directly impacts the signal through the victim interconnect, while PSN and GB indirectly impact the signal by affecting the drive strength of gates on the signal path. Mainly, PSN and GB are due to the fast switching circuits that introduce voltage fluctuations at their supply nets. We consider the combined behavior of the PD issues, as they would occur similarly in a real circuit design. In this work, our focus is to identify a test pattern that can capture the path delay variations in the combined presence of PD issues such as multi-aggressor crosstalk, PSN and GB on a victim path. ATPG tools utilize static timing analysis (STA) inputs [97] that are dependent on the lengths of a path. However, they do not consider the physical design data of a circuit (package, power/ground network parasitics, pad/pin location, cell placements, etc.) during pattern generation. So they might not identify actual test patterns that can lead to worst-case path delay in a circuit. Due to these reasons, path delay test using commercial ATPG

tools needs to be reexamined for taking into account PD issues.

2.2 Prior Work

Several pattern generation techniques were proposed in the literature to consider PD issues during path delay testing. Authors in [98] present a genetic algorithm based approach to find patterns that create maximum supply noise. Layout-aware pattern generation techniques were proposed in [99], [100] for PSN on critical (victim) paths. These approaches are better in terms of adding the PSN effects during pattern generation, but they take longer simulation time. In [101], a pattern generation framework by inducing maximum crosstalk effects across targeted delay-sensitive paths are proposed. A test generation technique in the presence of worst-case coupling effects for critical delay paths is presented in [95]. In [102], a pattern generation method with PSN and GB impact is shown and [96] proposes pattern evaluation and selection procedure for the impact of crosstalk and process variations. The method proposed in [103] focuses on an ATPG solution for the multi-aggressor crosstalk noise and [104] on finding the patterns activating worst-case crosstalk effects. All these works were based on pattern generation in the presence of the individual or partial combinations of PD issues (crosstalk, PSN or GB). The motivation behind this research work is quite different from all the previous work. Our efforts are to investigate on the combined impact of multi-aggressor crosstalk, PSN and GB on path delay variations and identify the test patterns that lead to worst-case path delay on a victim path. To do this, we develop a method for test pattern generation with physical design awareness.

2.3 Contributions and Chapter Organization

In this chapter, we propose a physical design aware pattern generation (PDAPG) method for identifying a test pattern that can capture worst-case path delay in the combined presence of PD issues.

Main contributions of this chapter are:

- To the best of our knowledge, this is the first work to investigate on the combined impact of multi-aggressor crosstalk, PSN and GB on pattern generation.

- We demonstrate, that a pattern generated by timing-aware ATPG tool does not capture the worst-case path delay.

- We present our PDAPG method to identify patterns that can capture worst-case path delay in the combined presence of PD issues.

- To identify a high-quality test pattern during path delay testing.

The chapter is organized as follows. In Section 2.4, we describe our motivational experiments which indicate the significance of path delay testing in the combined presence of Xtalk, PSN and GB. Section 2.5 presents the detailed flow of PDAPG method and the simulation results obtained on ITC'99 benchmark circuits. In Section 2.6, we summarize this chapter.

2.4 Motivational Experiments

The main motivation behind this chapter is presented in this section. The combined impact of PD issues such as multi-aggressor Xtalk, PSN and GB may cause worst-case delay on a circuit path. We highlight their impact based on the SPICE level simulations performed on a buffer gate circuit, shown in Figure 2.1. Simultaneously, we utilize an ATPG tool that is industrially used for testing path delay faults. ATPG tools function on the klogical level description of the circuit (i.e., VHDL or Verilog description), so they lack the physical design information while generating patterns. The worst-case delay test patterns identified from SPICE simulations are then compared with the patterns generated by the ATPG tool and their discrepancies were reported.

2.4.1 Path Delay Analysis

Figure 2.1 shows the buffer gate circuit developed in SPICE for analyzing the individual and combined impact of crosstalk, PSN and GB on path delay variations. This circuit consists of four buffer gates with two inputs {Ip1 Ip2} and two outputs {Op1 Op2}. π-network interconnects are modeled between the two gates for demonstrating crosstalk effects. The interconnect parasitic values and CMOS models for the buffer gates are estimated from the Predictive Technology Model (PTM) [105] for 90nm technology node. Using SPICE simulations, the path delay from {Ip1} to {Op1} is measured for all possible input pattern transitions at {Ip1 Ip2}. We introduce Xtalk noise by adding coupling capacitances

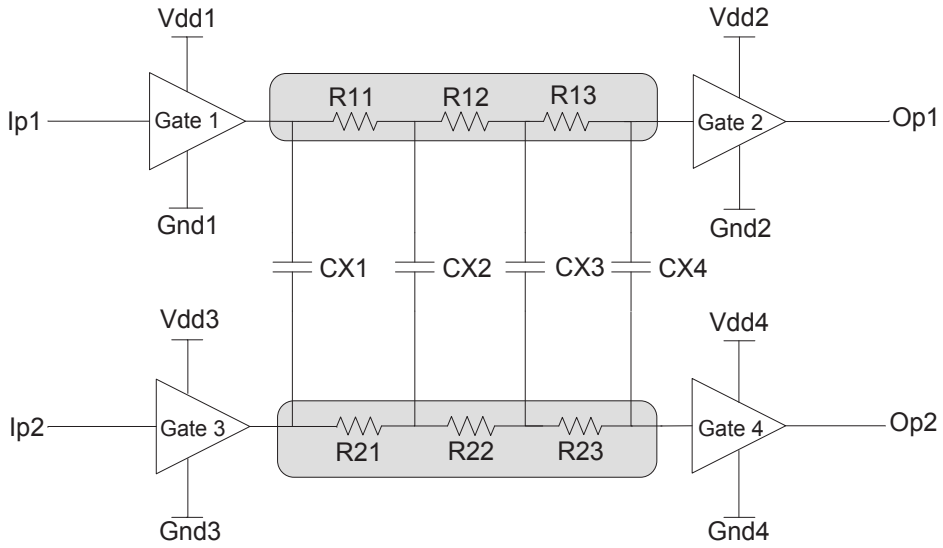


Figure 2.1: Buffer gate circuit for path delay analysis

{Cx1 Cx2 Cx3 Cx4} between the two interconnects. Also, parasitic resistances {R11 R12 R13 R21 R22 R23} are derived from the PTM intermediate interconnects models. For PSN and GB, the gate supply voltage and ground reference voltage at each gate {Gate1 Gate2 Gate3 Gate4} are modeled as {Vdd1 Vdd2 Vdd3 Vdd4} and {Gnd1 Gnd2 Gnd3 Gnd4}, respectively. The nominal power supply voltage, the ground reference voltage and switching frequency of 1V, 0V and 1GHz, respectively are used in this experiment.

Path delay variations are observed at the target output {Op1} with trigger input at {Ip1} for all the possible input signal transition patterns such as {10 10}, {10 01}, {01 10} and {01 01}} at {Ip1 Ip2}. The stable input patterns {00 00}, {00 01}, etc. were not considered here as there is no voltage level transition between the input and the output signals of this circuit. Please note that vector pair (01) represents a rising signal transition, (10) represents a falling signal transition, (00) represents a stable 0 condition and (11) represents a stable 1 condition. We not only varied the input patterns, our tests were also performed for varying input signal arrival time at their inputs. The different arrival times given at the primary input {Ip2} were {-200ps 0ps +200ps}. The +/- indicates the advance and the lag in arrival times at {Ip2} with respect to {Ip1}. We have assumed ±10% tolerance [106] in all these experiments as this can practically represent the supply voltage constraints in the distributed power and ground network grids. For analyzing PSN, gate supply voltages are varied by ±10% with respect to the nominal supply voltage of 1V i.e, {0.9V 1V 1.1V}. Similarly for GB, {0V 0.1V} are taken

for ground voltage levels. A total crosstalk capacitance of {2fF} is assumed for {Cx1 Cx2 Cx3 Cx4} with a load capacitance of {10fF} at the output of Gate2 and Gate4 (derived from PTM interconnect model).

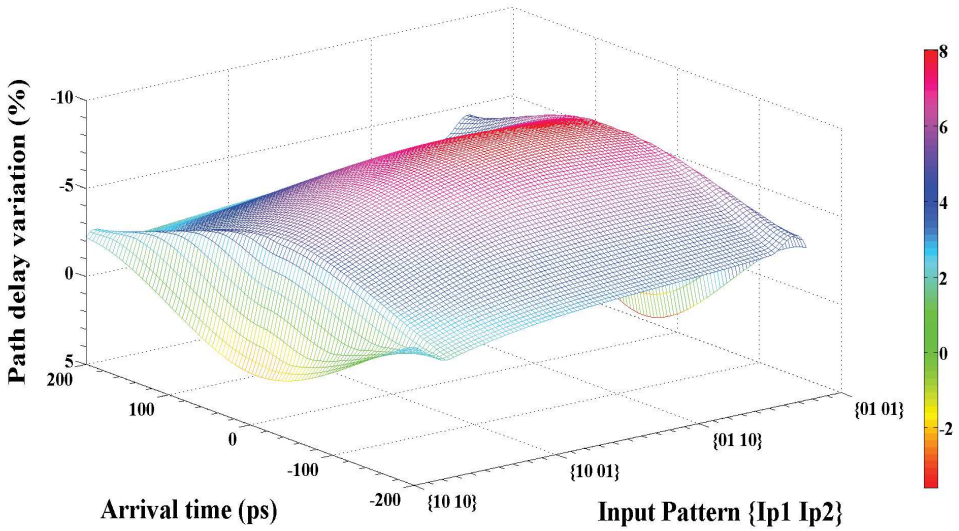


Figure 2.2: Path delay variations due to Xtalk noise

We consider three different cases to demonstrate the impact of each phenomena individually and collectively. In the first case, Xtalk effects are analyzed standalone by varying input patterns and their arrival time. PSN and GB were not considered here. Figure 2.2 refers to the variation in path delay due to the capacitive coupling between the aggressor and victim interconnects for an early and lag in arrival times at {Ip2}. The delay is plotted as a function of measured maximum Xtalk noise for the all the input patterns and varying arrival times.

Path delay variations were represented with respect to the nominal path delay with Xtalk noise on the circuit path. Positive and negative values on the z axis denote the speedup and slowdown impact on the victim path. We also note that for some of the input patterns, path delay is larger than others. This clearly indicates the input pattern to be used for obtaining a worst-case path delay. This delay plot becomes much more complex with many corners for bigger circuits.

In Table 2.1, we list the path delay variations for different input patterns with respect to their arrival times. Their impact on the victim path is also mentioned, i.e. slowdown or speedup impact. In this experiment, input pattern {01,10} leads to the worst-case path delay of up to -7.6%. We can make two major observations from this experiment. 1) Depending on the crosstalk noise conditions and arrival time, path delay on a victim path

Table 2.1: Path delay variations for different input patterns (Xtalk)

{Ip1 Ip2}	Arrival time	Delay variation	Delay impact
{10 10}	-200ps	-2.84%	slowdown
	0ps	+2.07%	speedup
	+200ps	-2.17%	slowdown
{10 01}	-200ps	-1.72%	slowdown
	0ps	-5.07%	slowdown
	+200ps	-1.69%	slowdown
{01 10}	-200ps	-3.64%	slowdown
	0ps	-7.60%	slowdown
	+200ps	-3.28%	slowdown
{01 01}	-200ps	-3.64%	slowdown
	0ps	+3.59%	speedup
	+200ps	-3.72%	slowdown

may increase or decrease. 2) Opposite direction input signal transition leads to worst-case path delay in a circuit. 3) Also, same direction signal transition may or may not lead to either slowdown or speedup depending on the value of the crosscoupling capacitances between the two interconnecting nets.

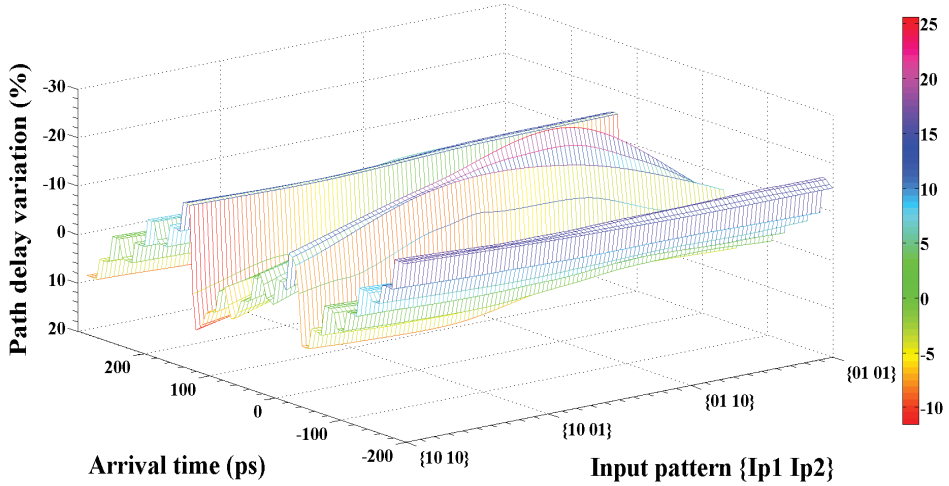


Figure 2.3: Path delay variations due to the combined impact of Xtalk noise and PSN

In the second case, the same circuit is utilized to analyze the combined impact of Xtalk and PSN. Figure 2.3 shows the variation in path delay due to the capacitive coupling between the aggressor and victim interconnects along with the voltage drop at each of the gates (maximum PSN). Path delay variations are plotted as a function of different input

pattern transitions and for varying input arrival time. GB is not considered here. The smooth curve in the earlier case gets replaced with many corners due to the addition of PSN. This is caused by the voltage overshoot and undershoot at the supply voltage Vdd of each gate.

Table 2.2: Path delay variations for different input patterns (Xtalk+PSN)

{Ip1 Ip2}	Arrival time	{Xtalk}{PSN}	Delay variation
{10 10}	{-200ps 0ps 200ps}	{0-2fF}{0.9V 1V 1.1V}	upto -17%
{10 01}			upto -20%
{01 10}			upto -26%
{01 01}			upto -15%

Path delay variations are listed in Table 2.2. Xtalk and PSN values were varied as shown in column 3 along with the varying input patterns in column 1 and input arrival times in column 2. As the combinations of all possible permutations of Xtalk noise, PSN distribution, input patterns, arrival times results in a quite large data set, so only the maximum values are shown in the table. From the simulation results, the worst-case delay is observed with the input pattern {01 10} for the same input arrival time {0ps} and for PSN of {0.9V 0.9V 1.1V 0.9V} at {Vdd1 Vdd2 Vdd3 Vdd4}. This is due to the voltage level difference at each gates affecting the drive strength of the gates in the succeeding stages [107]. We can conclude two more new observations from this experiment. 1) Path delay variation has increased to -26% compared to nominal delay (i.e., with no PSN and Xtalk) with the addition of a PD issue to the first case. 2) The combined minimal margin between the voltage drop in each gate does not lead to their worst-case path delay. They are due to the varying drive strength of the gates in the preceding stages before the path delay is measured.

In the third case, we perform path delay analysis in the combined presence of Xtalk, PSN and GB. The same circuit is utilized for this experiment also. Figure 2.4 refers to path delay variations with respect to 1) input pattern transitions, 2) arrival time, 3) capacitive coupling between the interconnects 4) supply voltage variation (maximum PSN) at the gate supply inputs and 5) variation in the ground network voltage (maximum GB) at each of the gates. The plot in the figure has become much more random with many corners due to the addition of GB.

Table 2.3 lists the path delay variations for different input patterns. Similarly, as in

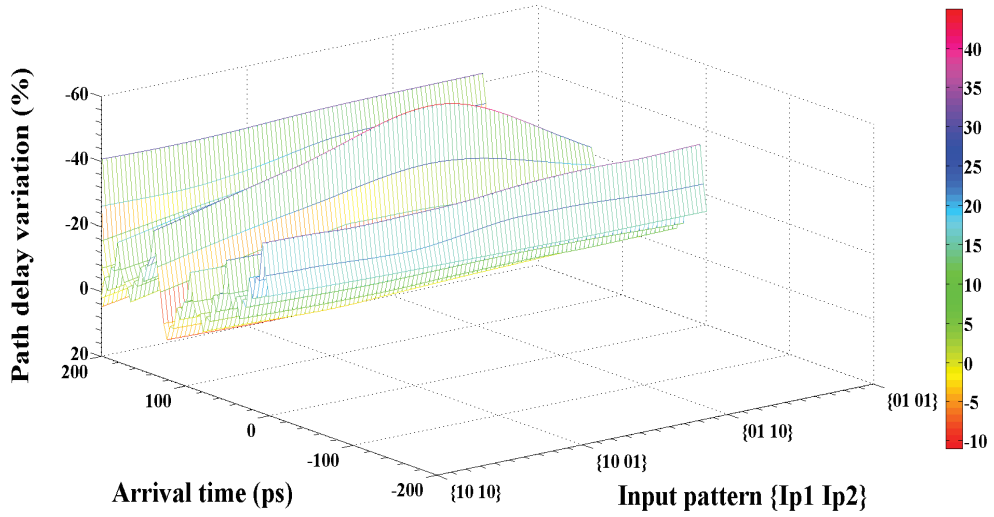


Figure 2.4: Path delay variations due to the combined impact of Xtalk noise, PSN and GB

Table 2.3: Path delay variations for different input patterns (Xtalk+PSN+GB)

{Ip1 Ip2}	Arrival time	{Xtalk}{PSN}{GB}	Delay variation
{10 10}	{-200ps 0ps 200ps}	{0-2fF}{0.9V 1V 1.1V} {0V 0.1V}	upto -32%
{10 01}			upto -34%
{01 10}			upto -47%
{01 01}			upto -38%

the previous case, the combinations of Xtalk, PSN, GB, input patterns and arrival time values results in a large data set and we have chosen only to show worst-case delays in the table.

Table 2.4: Worst-case path delays for the combined impact of Xtalk, PSN and GB

{Input pattern}{Xtalk}	{PSN}	{GB}
{01 10}{2fF}	{0.9V 0.9V 1.1V 0.9V}	{0.1V 0.1V 0V 0V}
	{0.9V 0.9V 1.1V 0.9V}	{0.1V 0.1V 0V 0.1V}
	{0.9V 0.9V 1.1V 1V}	{0.1V 0.1V 0V 0V}
	{0.9V 0.9V 1.1V 1V}	{0.1V 0.1V 0V 0.1V}
	{0.9V 0.9V 1.1V 1.1V}	{0.1V 0.1V 0V 0V}
	{0.9V 0.9V 1.1V 1.1V}	{0.1V 0.1V 0V 0.1V}

From the simulation results, the same amount of worst-case delay is observed for 6 different cases. In Table 2.4, worst-case delays for input pattern {01 10}, all arriving at the same time {0ps} and for different conditions of PSN and GB are shown. This is due to non-uniform voltage distribution among gates that impacts their drive strengths in the succeeding stages [102]. Additionally, we obtain up to -47% of path delay increase with respect to the nominal delay (i.e., with no Xtalk, PSN and GB) in the combined presence of Xtalk, PSN and GB.

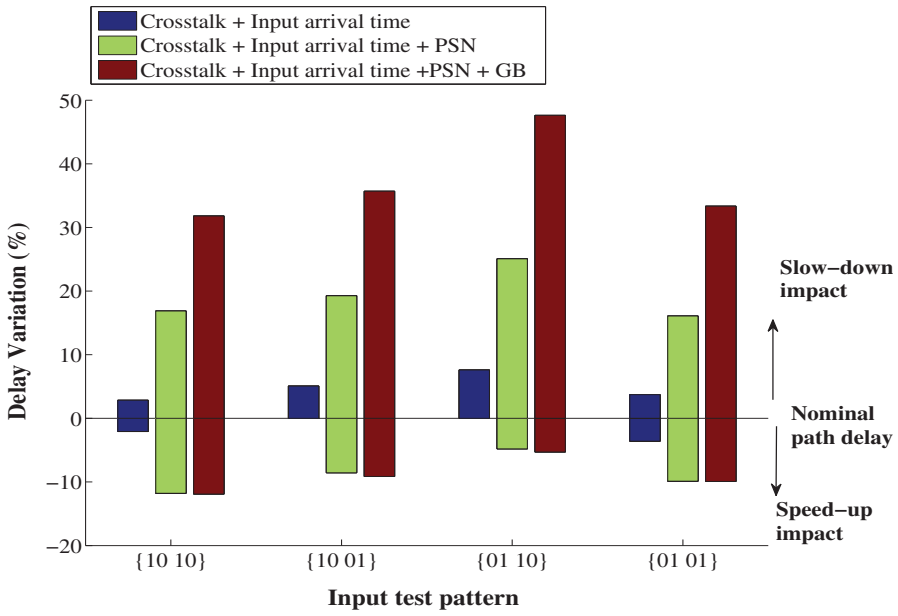


Figure 2.5: Path delay variation of buffer gate circuit

The individual and the collective impact of PD issues demonstrated earlier are collec-

tively shown in Figure 2.5. In the first case, the stand-alone impact of crosstalk effects is analyzed. In the second case, PSN is injected to the previous case. And in the final case, the GB is added along with the previous two cases. For each input pattern, the minimum to maximum path delay variations when compared with the nominal delay for the three different cases are indicated. The slowdown or speedup impact is also shown here. From the graph, we obtain a worst-case path delay of -47.62% for the input pattern {01 10}, at same input arrival time {0ps}, a Xtalk capacitance of {2fF}, for different values of PSN and GB. This is due to the Xtalk capacitance and the voltage level difference at each gate (i.e., PSN and GB) affecting the drive strength of the gates in their succeeding stages [107].

2.4.2 Path Delay Fault Testing

For path delay fault testing, we developed a combinational circuit module in Verilog similar to the buffer gate circuit in Figure 2.1 with two inputs {Ip1 Ip2} and two outputs {Op1 Op2}, refer to Figure 2.6. This circuit module does not have any interconnect parasitics, cross-coupling capacitances or supply voltages at their gate inputs to include the impacts of Xtalk noise, PSN and GB during pattern generation. We use the 90nm standard cell library for the path delay test flow utilizing an ATPG tool. Input patterns were generated for path delay fault test for the path from {Ip1} to {Op1}. The input pattern generated by the ATPG tool that can detect the path delay fault in a circuit is {01 01}.

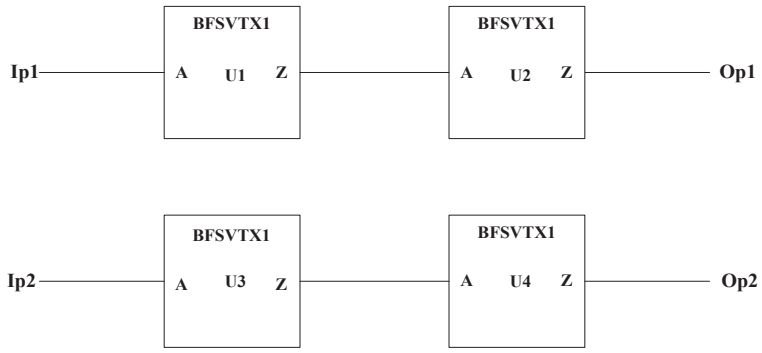


Figure 2.6: Verilog circuit for Path delay test in TetraMAX

Table 2.5: Input Pattern Comparison and Path Delay Variation

ATPG pattern	SPICE pattern	Delay variation	Result
{01 01}	{01 10}	-47.62%	Pattern mismatch

2.4.3 Comparison of input patterns

In Table 2.5, we show the ATPG tool generated input pattern (ATPG pattern) and the input pattern identified from our SPICE simulations (SPICE pattern). The ATPG pattern clearly does not match with the SPICE pattern selected from our detailed timing analysis with SPICE simulation. This is due to the addition of the PD issues such as crosstalk, PSN and GB. The worst-case path delay obtained using SPICE pattern is +47.62% larger than the one obtained using an ATPG pattern (with no Xtalk, PSN and GB). This shows that the path delay testing may miss the worst-case delay due to the low quality of the applied test set. The path delay variation in the presence of PD issues is quite high, thus indicating the need for refinement and the requirement of novel techniques to capture the worst-case path delay pattern.

In the next section, we propose a method to generate input test patterns to capture worst-case path delay under the combined impact of multi-aggressor crosstalk, PSN and GB.

2.5 Physical Design Aware Pattern Generation Method

2.5.1 PDAPG Flow for Pattern Generation

We propose a Physical Design Aware Pattern Generation (PDAPG) method to generate test patterns (vector pairs) that can capture worst-case path delay under the combined impact of multi-aggressor crosstalk noise, power supply noise and ground bounce. This method describes our complete flow from circuit netlist creation to a physical design aware pattern generation. Accordingly, this pattern can be utilized to test the presence of crosstalk noise, PSN and GB on a victim path. The PDAPG method is applied on ITC'99 benchmark circuits [1], to obtain a physical design aware pattern sequence. This method can be exhaustive for larger circuits in terms of computational time. Improvement and refinement of this method are explained in section 2.6. However, the validity and relevance

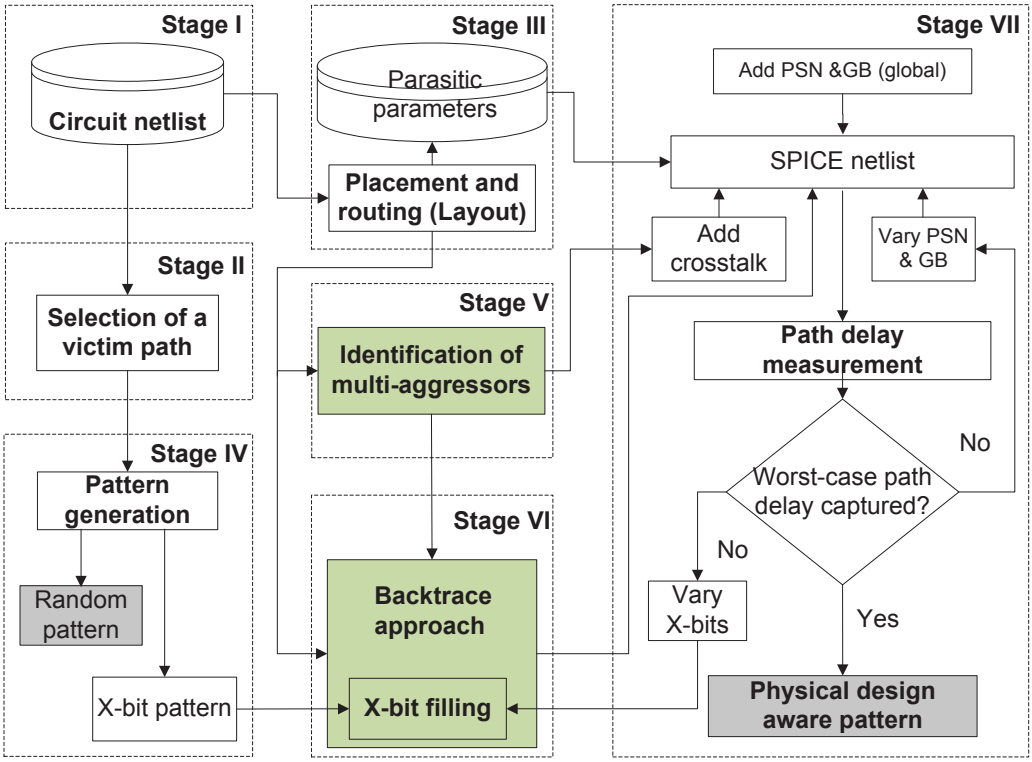


Figure 2.7: Physical design aware pattern generation method

of the proposed method show a great promise to close the gap between current path delay fault ATPG and the patterns identified under the impact of physical design issues.

PDAPG method is executed in seven different stages as shown in Figure 2.7. The detailed description of each stage is given in the following subsections. The seven stages are: (i) circuit netlist creation, (ii) selection of a victim path, (iii) placement and routing of circuit netlist (layout), (iv) pattern generation, (v) identification of multi-aggressors, (vi) X-bit filling by backtrace approach, and (vii) path delay measurement. Stage V and VI are our major contributions to the standard flow of physical design aware pattern generation method.

2.5.1.1 Stage I : Circuit Netlist Creation

We utilize the industrial RTL compiler tool, Cadence Encounter RC Compiler [53] for circuit netlist creation. The 90nm standard cell technology library is used during circuit netlist creation. The RTL codes are mentioned in Appendix A.

2.5.1.2 Stage II : Selection of a Victim Path

Victim paths are the most sensitive paths in path delay fault test, as they facilitate to predetermine path delay defects. Synopsys PrimeTime® Static Timing Analysis (STA) tool [97] is used for generating victim paths from the circuit netlist. This tool extracts a subset of all possible victim paths based on their signal propagation in the longest paths, as the faults are expected to be located in these paths. These paths are the combinatorial signal propagation path located between two scan flip-flops. Depending on the logic in the gates, some victim paths do not propagate a signal transition to the output of combinatorial path. Therefore, we select a single victim path from the subset of paths that can propagate a signal transition. This path is the robust path in the path delay fault test. Synopsys design constraints file (SDC) and CMOS 90nm technology database (db) files were given during STA. SDC file consists of design constraint information, timing assignments, power and area constraints of the circuit design. We use Synopsys ATPG tool, TetraMAX® [10] to test all the paths. Different steps in the selection of a victim path are shown in the algorithm below. The codes for generating the set of all possible victim paths are mentioned in Appendix A.

Algorithm 1: Victim path selection algorithm

```
01: Run ATPG (for all the victim paths)
02: Analyse for fault detectability status for a Path delay fault
if Detected faults (DT) then
    if Detected Robustly (DR) then
        if Strong robust then
            Select this victim path (strongly robust path) for pattern generation
        else
            This is a weak-robust path. Ignore this path
        end
    else
        Check another path
    end
else
    No paths can propagate signal transition
    Exit
end
```

Faults are assigned to classes indicating their current fault detection or detectability

status. Our objective from this algorithm is to select a strong robust, detected fault class, to ensure a signal transition in an at-speed test (as explained in section 1.3.3.1).

2.5.1.3 Stage III : Placement and Routing of Circuit Netlist

Developing layout of a circuit aids to infer the crosstalk induced noise issues from the perspective of a manufactured IC. We use the industrial CAD tool (Cadence SOC Encounter) for placement and routing of the circuit netlist [108] to integrate a layout-aware path delay test solution. The step by step procedure in this stage is as follows. Initially, design import of our initial verilog circuit netlist is performed, then included design rules, technology details and standard cell abstract information (Library exchange format); followed by floor planning, power planning (power supply network and ground network), standard cell placement (comprising of functional gates and scan flip-flops), power routing and finally signal routing. Parasitic parameters, i.e., resistances in the interconnecting nets and ground coupling capacitances between two standard cells were extracted in this stage. The codes for the placement and routing of circuit netlist are mentioned in Appendix A.

2.5.1.4 Stage IV : Pattern Generation

Synopsys ATPG tool, TetraMAX® [10] is employed to perform path delay fault test on the selected victim path from Stage II. This test ensures that a fault is detected in the path irrespective of other faults that may affect the circuit. As the outcome of this test, the tool generates a random pattern (without any X-bit) and an X-bit pattern that are capable of propagating a signal transition in the victim path. Random patterns are generated by the ATPG tool by randomly filling the X-bits in a pattern based on the built-in algorithms (extended D-Algorithm). X-bits are the don't care bits in the patterns. The care bits generated by the tool are not modified. X-bit patterns are also generated by the tool, left to be filled in any manner based on any heuristic or augmented algorithms. We take forward this X-bit pattern to stage VI, for our selective X-bit filling to identify a physical design aware pattern. The codes for pattern generation of the selected victim path can be referred to Appendix A.

We employ Full-sequential mode ATPG algorithm for executing the PDF tests (as explained in section 1.4.2.3). ATPG tool TetraMAX® uses only deterministic pattern generation. During deterministic pattern generation, the tool uses a pattern generation

process based on path-sensitivity concepts to generate a test vector that detects a specific fault in the design. After generating a vector, the tool fault-simulates the vector to determine the complete set of faults detected by the vector. Test pattern generation continues until all faults either have been detected or have been identified as undetectable by the process [10].

2.5.1.5 Stage V : Identification of Multi-aggressors

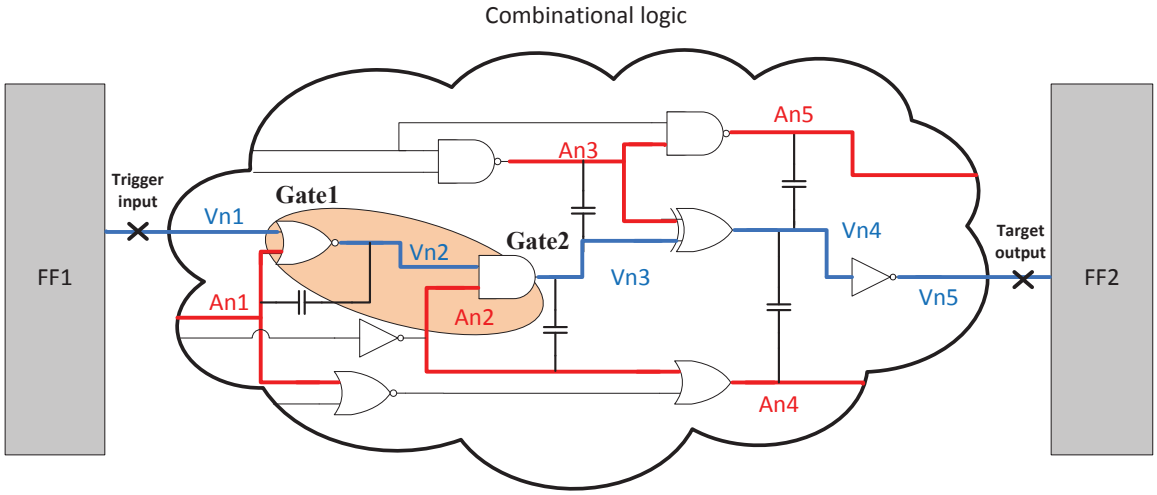


Figure 2.8: Identification of Multi-aggressors

At this stage, we identify all the multi-aggressor nets (i.e., the neighboring signal nets) causing crosstalk noise in the victim nets. The same identification process is repeated for the entire victim path. Their details are derived from the DEF file of the circuit layout. DEF (Design Exchange Format) file is produced after the placement and routing of the circuit netlist. It gives us, the information of the standard cell location and the placement of the interconnecting nets (X-Y plane) in the two-dimensional space. From this information, we estimated all the individual crosstalk capacitance's between the victim nets and the aggressor nets. Also, we have considered parts of an interconnecting net (of a victim net and an aggressor net) that were located in multiple metallic layers and connected by vias. For illustration, consider a victim path between the two scan flip-flops FF1 and FF2 as shown in Figure 2.8 with the victim nets (i.e, Vn1, Vn2, Vn3, Vn4 and Vn5) and the identified multi-aggressor nets (i.e, An1, An2, An3, An4 and An5). This sketch is derived from the actual physical design representation of a circuit layout, shown in Figure 2.9. Path delay is measured between the two points, i.e., the trigger input and

the target output of the victim path. Trigger input and the target output are respectively the start and the end of a victim path, where the propagated signal transition is launched and then captured.

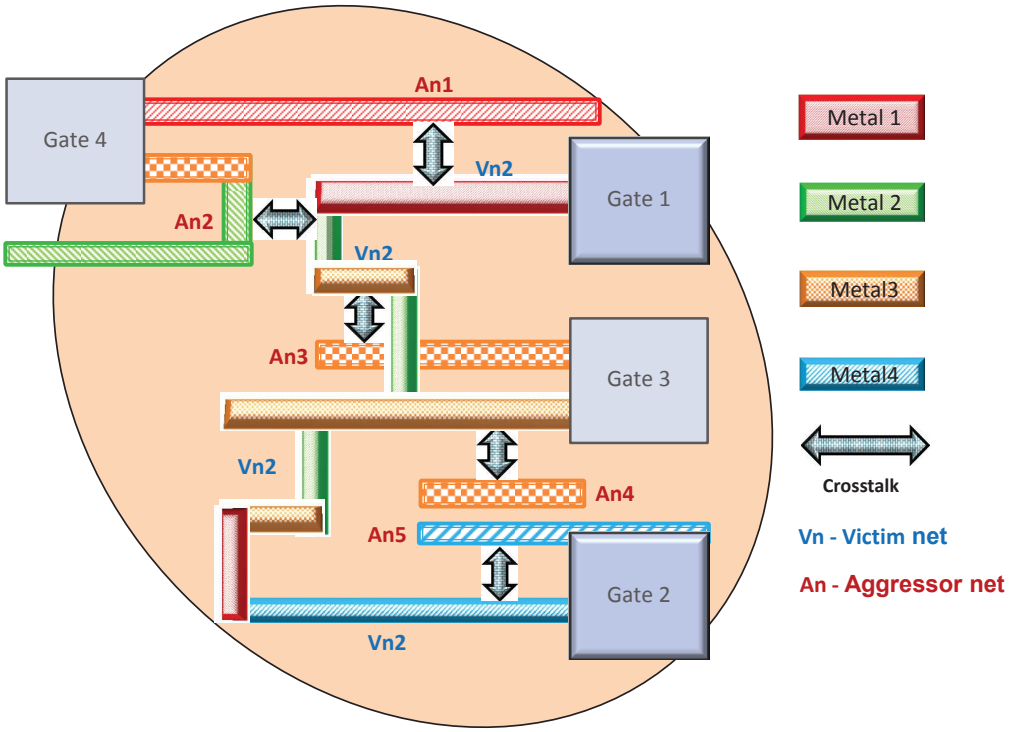


Figure 2.9: Victim-Aggressor net sketch from the layout

Table 2.6: Victim nets and aggressor nets

Victim nets	Aggressor nets
Vn1, Vn2, Vn3, Vn4, Vn5	An1, An2, An3, An4, An5

The different steps in the identification procedure for selecting a victim net is briefly shown in Algorithm2, below. Initially, a single victim net is selected. Then, checked whether the net is horizontally or vertically located in the X-Y plane of the circuit's layout. After knowing its location, it is checked for the placement in the metal layer, i.e., metal1 (M1), metal1 (M2), metal1 (M3), metal1 (M4), metal1 (M5), metal1 (M6). Once metallic placement is verified, different conditions are analysed for the orientation of the aggressor net with respect to the victim net. The horizontal orientation conditions are represented in Figure 2.10. And, the vertical orientation conditions are represented in Figure 2.11. Subsequently, cross-coupling capacitance is measured, the steps are repeated

for all the orientation conditions and for all the metallic layer placements. Then the same is repeated for the vertically located nets. Finally, for rest of the victim nets in the victim path. In this way, we were able to find the aggressor nets for the entire victim path.

Algorithm 2: Algorithm for identifying multi-aggressors

01: DEF file as input

02: Select a single victim net from the victim path

03: Check whether this victim net is horizontally/vertically located in the X-Y plane of the circuit layout

if *(Horizontally located)* **then**

- **if** *(M1,M2,M3,M4,M5,M6)* **then**
  - **if** *(H1,H2,H3,H4,H5,H6)* **then**
    - C = (ϵA)/d; measure the Xtalk capacitance/fringe capacitance
    - repeat the same for all H1,H2,H3,H4,H5,H6
  - **else**
    - no cross-coupling capacitance, Ignore this horizontally placed net
  - **end**
- **else**
  - check for vertical nets
- **end**

else

- **if** *(Vertically located)* **then**
  - **if** *(M1,M2,M3,M4,M5,M6)* **then**
    - **if** *(V1,V2,V3,V4,V5,V6)* **then**
      - C = (ϵA)/d; measure the Xtalk capacitance/fringe capacitance
      - repeat the same for all V1,V2,V3,V4,V5,V6
    - **else**
      - no cross-coupling capacitance
      - Ignore this vertically placed net
    - **end**
  - **else**
    - no aggressor net in this metal layer, exit
  - **end**
- **else**
  - no aggressor net, exit
- **end**

end

04: Repeat the same for all the victim nets in the victim path

We have considered all multi-aggressor interconnects from a minimum to maximum

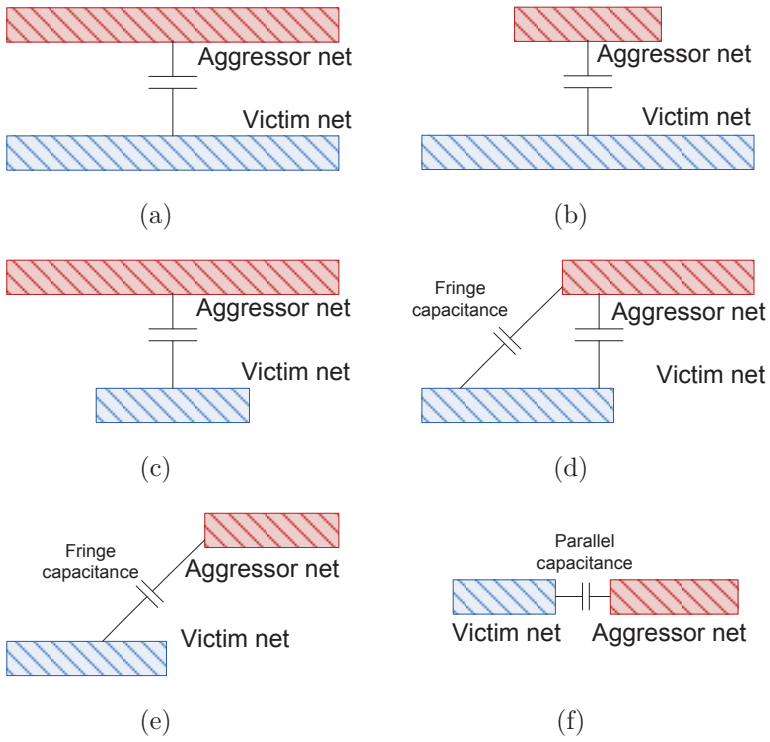


Figure 2.10: Aggressor net (An) and victim net (Vn) is horizontally located (a) both with equal length (H1) (b) Vn > An (H2) (c) Vn < An (H3) (d) Vn, An in parallel, but with fringe capacitance (H4) (e) distant apart between the nets, but with fringe capacitance (H5) (f) in the same X plane (H6)

spacing based on the Design Rule Check (DRC) of the 90nm technology. The individual crosstalk capacitance's were calculated using the equation C = (ϵA)/d; where A is the area of the aggressor net, d is spacing between the aggressor and victim net and ϵ is the permittivity of dielectric material between two nets. These multi-aggressor crosstalk capacitances were added to the SPICE netlist in the last stage of path delay measurement.

2.5.1.6 Stage VI : X-bit Filling by Backtrace Approach

We utilize the X-bit pattern generated in stage IV for our selective X-bit filling based on backtrace approach. After identifying all the multi-aggressors from the circuit layout, we then employ to trace back to the origin of their X-bit inputs that control each of the aggressor nets, as depicted in Figure 2.12. Some of the backtraced aggressor net gate input can be an aggressor net itself (for example An2 and An3), the subset of all these inputs are considered. There might be a huge number of unfilled X-bits and filled bits in the primary input (PI) and the scan flip-flop input (SI) of the patterns generated by the ATPG. Our approach is to identify and fill the relevant X-bits that is getting

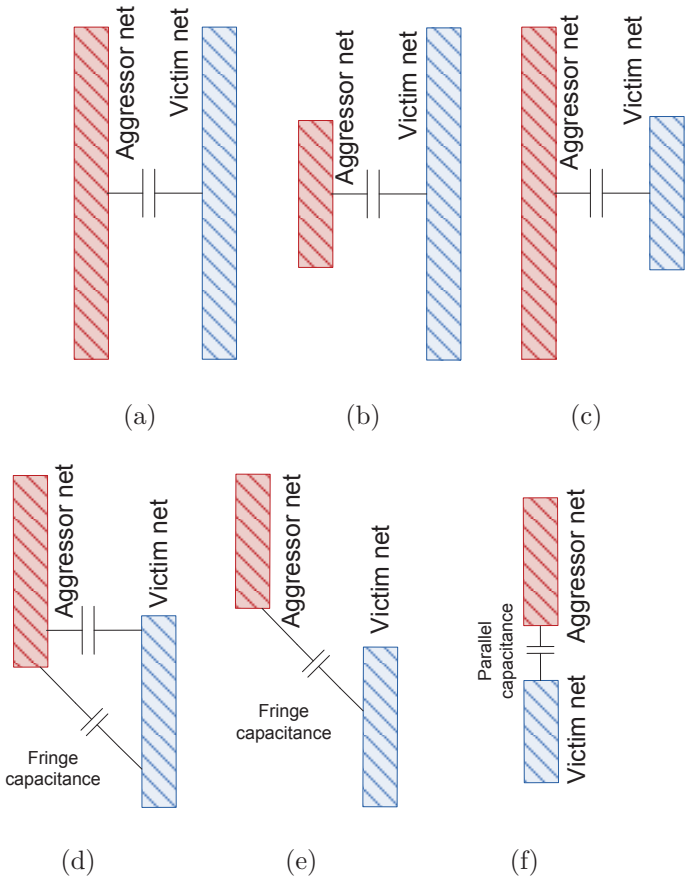


Figure 2.11: Aggressor net (An) and victim net (Vn) is vertically located (a) both with equal length (V1) (b) Vn > An (V2) (c) Vn < An (V3) (d) Vn, An in parallel, but with fringe capacitance (V4) (e) distant apart between the nets, but with fringe capacitance (V5) (f) in the same Y plane (V6)

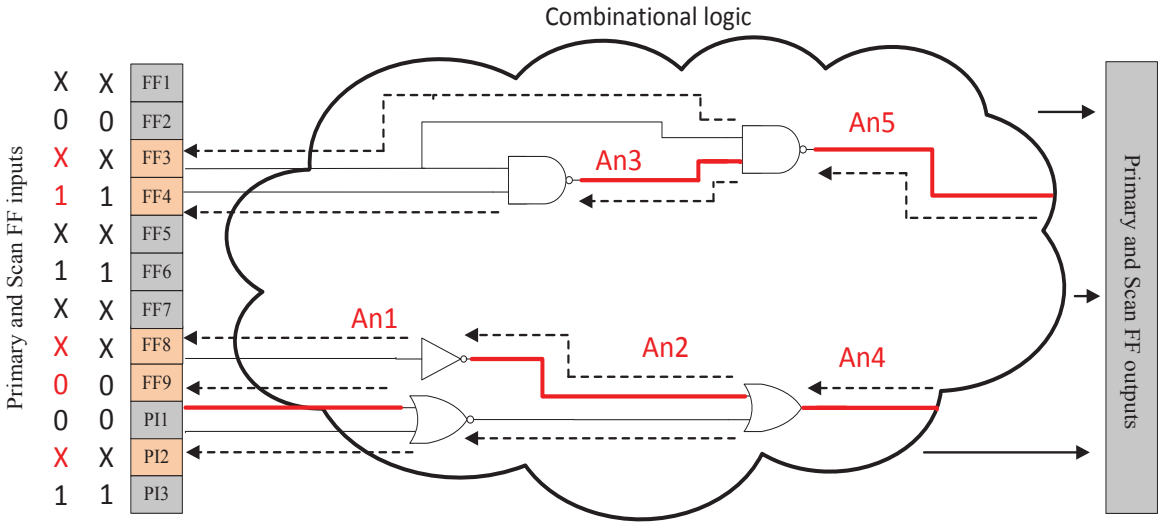


Figure 2.12: X-filling by backtrace approach

affected by crosstalk noise. This approach minimizes the number of X-bit filling from the total X-bits produced, which successively reduces the total number of input pattern

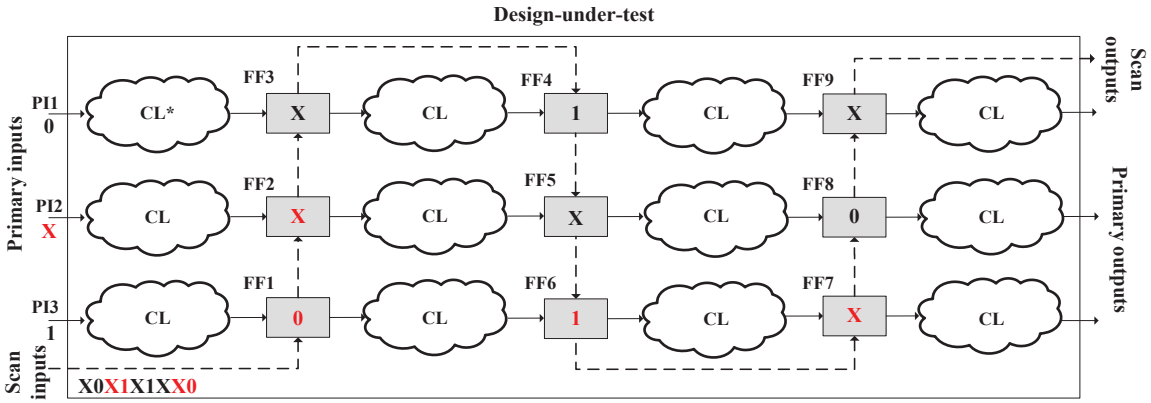
combinations (each X-bit can be filled and tested with '0'and '1') applied during the test for physical design defects. The relevant X-bits in the input patterns were filled in the STIL output file, generated by ATPG. STIL (Standard Test Interface Language) file gives the information about the insertion structure of the scan flip-flops and the location of the relevant X-bits that needs to be filled in the input patterns. Further efforts to elucidate the backtrace approach is mentioned below.

Table 2.7: Victim nets and aggressor nets

Approach	Scan inputs	Primary inputs	#patterns
Classical ATPG	X0X1X1XX0	0X1	2^6=64
Backtrace filling	X0x1X1Xx0	0x1	2^3=8

In this approach, all the identified multi-aggressors were backtraced to the individual scan flip-flop inputs and primary inputs. These input bits are the only relevant X-bits that impact the victim net and needs to be filled. Consider the combinational logic part of the circuit in Figure 2.12, it has 3 primary inputs (PI) with primary input insertion structure 'PI1 PI2 PI3' and 9 scan flip-flop inputs (SI) with scan input insertion structure 'FF1 FF2 FF3 FF4 FF5 FF6 FF7 FF8 FF9'. This is further explained by giving test inputs PI = 0X1 and SI = X0X1X1XX0 to the circuit. From this data, we can see that 6 X-bits needs to be filled in the input pattern. Therefore, $2^6 = 64$ X-filled input pattern combinations have to be applied to test the circuit for finding a worst-case path delay pattern. By applying our backtrace approach, we find that inputs to PI2, FF3, FF4, FF8 and FF9 are the most relevant bits in our analysis, refer Figure 2.12. Among them, FF4 and FF9 were already filled by the ATPG; since these bits are controllable but not observable from the primary inputs. Hence, only 3 X-bits are needed to be filled, so $2^3 = 8$ X-filled pattern combinations to test for worst-case path delay under the impact of crosstalk noise since they control the inputs through multi-aggressor nets. By our approach, we show a reduction of 87.50% in pattern count which consequently lowers the testing time of a path delay fault.

We adopted the launch-off-capture (LOC) scheme for the scan-based path delay fault test in the presence of multi-aggressor crosstalk noise. The structural representation of a design under test is shown in Figure 2.13. The scan chain with the scan input and scan output ports are shown in dotted lines with scan flip-flops between the combinational



CL* is the combinational logic blocks between the sequential elements

Figure 2.13: Path delay fault testing with LOC

logic blocks.

The step-by-step details of a typical scan operations for LOC delay test are described as follows:

- The first step is to put the scan flip-flops into scan mode. This is done by using the Scan Enable signal. In this case, forcing Scan Enable to 1 enables the scan mode. Note that initially all the scan flip-flops at unknown state (X).

- The scan-in process starts. Then the first 3 bits are scanned in. A single test bit is shifted-in at each clock cycle. Usually, the scan shift frequency is much slower than the functional operating frequency of the CUT.

- At this stage, the complete test vector is shifted-in. Scan mode can be disabled by forcing Scan Enable to 0. Note that the shifted-in test vector is currently applied to the combinational logic pieces that are driven by scan flip-flops. It means that 2nd, 3rd, and 4th combinational logic blocks are already forced test inputs.

- The next step is to force primary input (PI) values and measure the primary output (PO) values: force PI and measure PO.

- Now, it is required to create a second test vector to create signal transitions. The second vector will be the output responses of the combinational blocks. Each block will generate the second test vector for the next stage. Since there is no stage before the 1st combinational block, force PI needs to be applied one more time.

- In order to push the output responses of combinational blocks into scan flip-flops, the system clock is toggled. Once this is done and second PI force is applied, the second test vector for the delay test is generated. The second input vector will generate output responses similar to the first one. These output responses needs to be captured, similar to the first one, by toggling the system clock. However, now there is a difference: The system clock has to be toggled at the real operating frequency. This means that the period between the first clock toggle and second clock toggle should be equal to functional clock period. In this way, the delay-test responses are captured at the functional operating frequency. As a result, correct functionality of the circuit is tested at-speed.

- Finally, the captured responses are shifted-out using the slow clock frequency.

Here, we mention our add-on's to the existing LOC scheme for the example we were describing. During the scan mode, the scan inputs 'X0X1X1XX0'were shifted in serially at each clock cycle and applied to the logic blocks. Then forced primary inputs '0X1'. We selectively fill these X-bits obtained from our backtrace approach (as shown in Table 2.7, row 3) for effectively testing patterns under the impact of multi-aggressor crosstalk noise.

2.5.1.7 Stage VII : Path Delay Measurement

In this stage, we measure the delay variation due to the combined impact of multi-aggressor Xtalk, PSN and GB on a victim path. The SPICE circuit netlist is updated with all the calculated multi-aggressor cross-coupling capacitance's from stage V. We also included parasitic parameters extracted from the physical design layout of the circuit netlist along with the global power supply and the ground voltage as inputs. SPICE simulations were performed for capturing the worst-case path delay, with global variation of power and ground voltage level distribution (10% tolerance from the global values) as explained in section 2.4. The combined impacts were evaluated for all the relevantly filled X-bit patterns combinations of the primary and scan inputs. Then, SPICE simulations were performed for capturing the worst-case path delay. Path delay measurement can also be performed using STA tools. However, for establishing the realistic nature (by adding the physical design data such as parasitics in the interconnecting nets) of the multi-aggressor crosstalk noise, we utilize SPICE simulations to obtain a worst-case path delay pattern.

By our presented PDAPG method, we identify partially filled X-bit pattern that can capture worst-case path delay in a victim path. This pattern is compared with the random pattern generated in stage IV and their mismatches are highlighted to show effectiveness in the pattern we identify.

2.5.2 Experimental Results

Table 2.8: Functionality of ITC'99 Benchmark circuits [1]

Name	Functionality	# of Gates	# of FFs[a]
b01	FSM that compares serial flows	49	5
b02	FSM that recognizes BCD numbers	28	4
b05	Elaborate the contents of a memory	998	33
b06	Interrupt handler	56	8
b09	Serial to serial converter	170	27
b11	Scramble string with variable cipher	770	30
b14	Viper processor (subset)	10098	215
b22	A copy of b14 and two modified versions of b14	21772	611

[a] number of scan flip-flops.

In this section, experimental results for eight full-scanned versions of ITC'99 Benchmark [1] circuits are given. Our main goal is to identify a test pattern that can capture the worst-case path delay in the combined presence of multi-aggressor crosstalk, PSN and GB. Although, this method is implemented on a selected victim path, the same can be applied on any victim path. SPICE simulations are finally run to identify the worst-case path delay pattern. The delay is captured for the combined impact of multi-aggressor crosstalk, PSN and GB for different test patterns. The description of the benchmark circuits, the number of gates and the number of scan flip-flops for the circuits are shown in columns 1, 2 and 3 respectively in Table 2.8. Experimental results of ITC'99 benchmark circuits are summarized in Table 2.9. The total number of victim paths generated are given in column 2. In column 3, the %reduction of X-bit input pattern count for a selected victim path in shown. Runtime for testing all the patterns and the path delay variation (%) in comparison with the ATPG pattern are shown in column 4 and 5 respectively.

Table 2.9: Circuit description and Path delay variation results for ITC'99 Benchmark circuits

Ckt	#VP's	#X[a]	Time[b]	Delay variation[c]
b01	5	50%	194s	35.45%
b02	4	50%	155s	52.92%
b05	34	99.21%	42214s	60.81%
b06	8	75%	620s	47.46%
b09	28	93.7%	8691s	43.20%
b11	30	99.97%	74496s	67.66%
b14	215	99.9%	1229874s	71.03%
b22	613	99.99%	9936728s	79.34%
Average		83.47%		57.23%

[a]% Reduction of X-bit input pattern count [b]Runtime for testing all patterns based on PDAPG method (in sec) [c]Path delay variation (%)

We demonstrate our PDAPG method based on b06 benchmark circuit. Eight victim paths were generated for b06 circuit netlist. These paths are classified as ATPG untestable (3 paths), ATPG undetected (1 path) and ATPG detected (4 paths) based on their path delay fault class. Refer chapter 1, fault category subsection for more details. The ATPG detected paths were further analyzed for robust (3 paths) and non-robust (1 path) paths. From this list, we selected a strongly robust victim path (as described in section 2.5.1.2) for our path delay fault analysis. ATPG (SI pattern - 10011000 and PI pattern - 100110) and the X-bit pattern (SI pattern - XX0110XX and PI pattern - 100X1X) were generated. We identified all the multi-aggressors from the PD layout of the circuit netlist. Their individual coupling capacitance's (measured 0.0005pF in total) were calculated between each aggressor and victim interconnects. By our backtrace approach, the relevant X-bits (inducing crosstalk) that control the input of the X-bit pattern were detected. They are the first two X-bits of SI pattern (xx0110XX) and both X-bits of PI pattern (100x1x), denoted by small letter 'x'. This method reduces the complexity in testing of test pattern by 75% (i.e, from 2^5(5 X-bits) = 32 to 2^3(3 X-bits) = 8). Nominal path delay of 216ps is measured for these X-filled test patterns without considering any PD issues. After

considering the combined impact of PD issues, we measured a worst-case path delay of 319ps for multi-aggressor crosstalk capacitance, PSN and GB values of 0.0005pF, 0.9V and 0.1V respectively. The path delay obtained for the test pattern (PI pattern - 100111 and SI pattern -110110XX) is 47.46% larger than the nominal path delay (without considering crosstalk, PSN and GB impacts on an ATPG pattern); hence this is a high quality test pattern that should be identified during path delay testing. The high quality test pattern signifies that this pattern gives the worst-case path delay, when applied to the selected victim path. By implementing our PDAPG method on the selected ITC'99 benchmark circuits, we were able to obtain the following: 1) an average reduction of test pattern count by 83.47%, and 2) capture an average path delay variation of 57.23%.

Table 2.10 shows the pattern comparison for the scan inputs (SI) and the primary inputs (PI) obtained from ATPG tool and our PDAPG method. Test pattern generated by the ATPG tool (ATPG pattern) is shown in column 4 with the pattern identified by physical design aware pattern generation method (PDAPG pattern) shown in column 5, along with the X-bit pattern in column 3. The mismatch between the test patterns can be clearly distinguished. The worst-case path delay variation between the ATPG pattern and the PDAPG pattern for the selected victim path is quite high. Ignoring this pattern may cause path delay defect escape during path delay testing. Thus, we show that the PDAPG method is an accurate and efficient delay testing strategy for ensuring better path delay defect coverage in the combined presence of multi-aggressor crosstalk, PSN and GB. We acknowledge that the proposed approach is accurate after performing a detailed timing analysis but is also limited in considering all the multi-aggressors from layout due to the larger circuit size and high volume of X-bit filling, which is also the focus of our ongoing work.

2.6 Summary

In this chapter, we presented a novel physical design aware pattern generation (PDAPG) method for path delay testing. PDAPG method focuses on identifying the test patterns that can capture worst-case path delay on victim paths in a circuit. The path delay is captured in the combined presence of physical design issues such as multi-aggressor crosstalk, power supply noise and ground bounce. Our PDAPG method was implemented

Table 2.10: Input pattern comparison results of ITC'99 Benchmark circuits

Ckt	Input	SI	PI
b01	X-bit pattern	x100 X	1100 1x
	Random pattern	1100 0	1100 10
	PDAPG pattern	1100 X	1100 11
b02	X-bit pattern	xX00	0000 x
	Random pattern	1000	0000 0
	PDAPG pattern	1X00	0000 1
b05	X-bit pattern	1111 0111 0100 Xxxx 11xx xxXX XXX0 1111 1X	00x0 x
	Random pattern	1111 0111 0100 1000 1110 0101 1010 1111 10	0010 0
	PDAPG pattern	1111 0111 0101 X101 1111 00XX XXX0 1111 1X	0010 01
b06	X-bit pattern	xx01 10XX	100x 1x
	Random pattern	1001 1000	1001 10
	PDAPG pattern	1101 10XX	1001 11
b09	X-bit pattern	x100 1100 111X XxxX xX01 1001 1111	0000 x
	Random pattern	1100 1100 1110 0010 0101 1001 1111	0000 0
	PDAPG pattern	1100 1100 111X X01X 1X01 1001 1111	0000 1
b11	X-bit pattern	xxXX XX10 11Xx xxX0 1100 1000 X111 xx	xxXX XXX0 00x
	Random pattern	1000 1010 1101 0110 1100 1000 0111 10	1100 0110 001
	PDAPG pattern	10XX XX10 11X1 11X0 1100 1000 X111 11	10XX XXX0 001
b14	X-bit pattern	Xx11 X01x xX10 X001 x010 XX1x 1001 0X10 XX00 1101 xxxx X01x	1100 010x xxxx xxx0 0001 1xxx x001 10x
	Random pattern	1011 1011 0010 0001 1010 1011 1001 0110 1000 1101 1010 0010	1100 0100 1010 1000 0001 1100 0001 100
	PDAPG pattern	X111 X010 1X10 X001 0010 XX11 1001 0X10 XX00 1101 1110 X010	1100 0101 1110 0010 0001 1110 0001 100

on ITC'99 benchmark circuits and results were shown by pattern comparison and path delay measurement. By our backtrace approach, we also demonstrated its effectiveness in reducing pattern count during path delay testing. Although, we used a single robust victim path as our initial victim path repository, our method can be applied to any number of victim paths for efficiently identifying the high-quality test patterns. Results suggest the refinement of the existing test methods in timing-aware ATPG tools for incorporating the combined impact of different PD issues. We acknowledge that the proposed approach is accurate after performing a detailed timing analysis, but is exhaustive while performing SPICE simulations for all test patterns due to the larger circuit size and high volume of X-bit filling. This is also the focus of our next work to generate crosstalk-aware patterns.

Chapter 3

An ATPG Flow to Generate Crosstalk-Aware Path Delay Pattern

Contents

3.1 Introduction

As shown in chapter 2, crosstalk noise causes considerable path delay variations. The physical design aware pattern generation method previously discussed is an exhaustive SPICE-based simulation for bigger circuits. In this work, we aim at generating a dedicated crosstalk-aware pattern that can be utilized for path delay fault testing without using SPICE-based simulations. Due to the complexity in developing and modeling power supply noise and ground bounce effects to ATPG tool, we have only considered crosstalk noise issues in this chapter. The other effects will be dealt as future perspectives.

Several prior works were focused on crosstalk noise and delay-aware ATPG. Still, there is a growing concern in the semiconductor industry for devising better test pattern generation methods to identify an effective pattern at a lesser computational time. ATPG tools utilize the existing path delay fault models to generate test patterns that may or may not capture a crosstalk induced delay defect. These tools utilize static timing analysis (STA) inputs [97] that are dependent on the lengths of a path (consists of many gates in a path). With the objective of reducing computational time in pattern generation procedure, these delay fault models are based on zero interconnect delay in ATPGs. They target to capture as many faults as possible for an entire circuit with a lesser number of patterns in reduced time. Also, they do not consider the physical design data of a circuit during pattern generation. In other words, interconnect delays which are due to crosstalk noise are neglected during the ATPG path delay test. Therefore, a worst-case path delay pattern may not be identified during the delay test. Due to these reasons, path delay test using timing-aware ATPG tools need to be re-examined for taking into account crosstalk noise delay.

3.2 Prior Work

In recent years, path delay based pattern generation has received greater attention for capturing delay defects. Several techniques were proposed in the literature for considering crosstalk noise during path delay fault testing. A circuit redesign can reduce crosstalk noise, but cannot fully wipe out its impact on circuits in advanced technology models with higher density. Therefore, there is always a need for accurate crosstalk analysis and better pattern generation techniques.

Some of the earlier works based on delay testing, consider the gates and interconnect delays. Authors in [109] present a genetic algorithm based approach to find patterns that produce longer path delays. In [101], a pattern generation framework by inducing maximum crosstalk effects across targeted delay-sensitive paths are proposed. A test generation technique in the presence of worst-case coupling effects for critical delay paths is presented in [95]. Furthermore, [96] proposes pattern evaluation and selection procedure for the impact of crosstalk and process variations. The method proposed in [103] focuses on an ATPG solution for the multi-aggressor crosstalk noise and on finding the patterns

activating worst-case crosstalk effects. All these works were based on finding an accurate set of patterns, but may take higher execution time with bigger circuits. Also, path delay estimation on some of these works were based on gate level circuit netlist. The approach proposed in [110] uses the physical design layout information for multiple aggressor crosstalk faults. However, their estimations were deployed on heuristic techniques.

For reducing the complexity in ATPG, the works in [111] and [18] propose crosstalk-aware pattern generation in comparatively lesser computational time. Our efforts are to investigate the impact of multi-aggressor crosstalk noise on path delay variations and identify the test patterns that lead to worst-case path delay. To do this, we develop a method to generate crosstalk-aware test patterns.

3.3 Contributions and Chapter Organization

Our major contributions presented in this chapter are:

- We first show that the path delay in a circuit is significantly high due to crosstalk noise. Thereafter, demonstrated that an ATPG tool is incapable of generating patterns causing worst-case path delay.

- Then, we present our Crosstalk-Aware Test Pattern Generation (Xtalk-ATPG) method to identify high quality test patterns in the presence of multi-aggressor crosstalk noise. Xtalk-ATPG method can be computationally exhaustive.

- Therefore, we further aim at customizing the existing path delay fault ATPG to include crosstalk noise impact. This can be an effective alternative to SPICE-based simulation in delay testing [112] in terms of computational time.

- All these efforts lead to generate a high quality test pattern during path delay testing.

The rest of the chapter is organized as follows. In Section 3.4, we describe a motivational experiment to indicate the importance of adding crosstalk noise during the path delay test. Section 3.5 presents the detailed flow of Xtalk-ATPG method and their experimental results. In Section 3.6, we elaborate our proposed constrained ATPG method of pattern generation by using the path delay fault test. In Section 3.7, we provide the summary of our contributions in this chapter.

3.4 Motivational Experiments

The main motivation behind our work of crosstalk-aware pattern generation is presented in this section. We highlight the impact of crosstalk noise by identifying a worst-case path delay pattern from SPICE simulations. Simultaneously, we execute path delay fault test to generate a pattern that can capture path delay defect. This test utilizes the existing path delay fault model in the ATPG tool, TetraMAX® [10]. Basically, our goal is to show that a standard ATPG tool may not be able to generate good patterns that can capture worst-case path delay in the presence of crosstalk. And our experimental results will show, such situation occurs simply because any commercial ATPG tool does not take into account any physical design properties (i.e., parasitics for crosstalk, noise on power and ground lines, etc.) as they work on the logical level description (gate level netlist) of the circuit (i.e., VHDL or Verilog description) during pattern generation. The worst-case delay test patterns identified from SPICE simulations are compared with the patterns generated by the ATPG tool and their discrepancies are reported.

3.4.1 Path Delay Analysis

An ISCAS89 benchmark circuit s27 is developed in SPICE to analyse the impact of crosstalk noise on path delay variations. Here, we have considered a sequential circuit in comparison to the motivational experiment in chapter 2 in order to understand the path delay variation in a sequential circuit. This SPICE level circuit includes parasitics in interconnects which are usually ignored at gate level simulation. Measured path delays for all test patterns obtained from SPICE simulations helps us to understand which patterns and under what conditions will provide the worst-case delay in a circuit path.

Figure 3.1(a) depicts s27 circuit with 10 gates, 3 D-type flip-flops, 4 inputs and a single output. The circuit is synthesized, DFT scan-chains [53] inserted and circuit netlist with 7 functional standard cells and 3 scan-out D Flip-Flops are generated. This facilitates one-to-one comparison of the simulation results from path delay analysis and path delay fault testing (explained in section 2.4.2). Placement and routing of the circuit netlist are then performed to extract the parasitic parameters (in gates and interconnects) from the layout [108]. Interconnect parasitic parameters consists of resistances (R1, R2, R3) and ground coupling capacitances (C_{g1}, C_{g2}, C_{g3}, C_{g4}) as shown in Figure 3.1(b). Parasitic

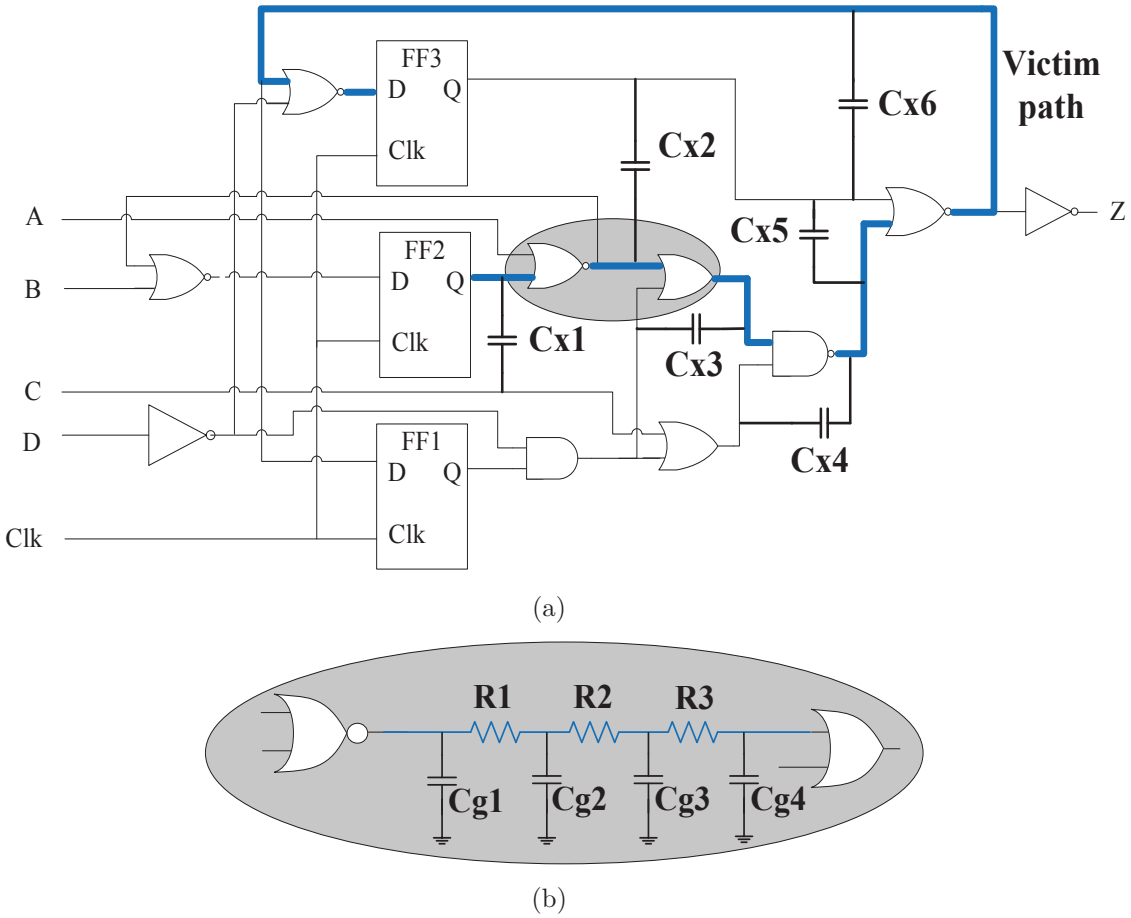


Figure 3.1: (a) s27 benchmark circuit-under-test (b) 3-pi network model for interconnects

inductances in interconnects are not considered in this work, as it may be too complex to analyse crosstalk effect on a simple circuit. Interconnecting nets between all the standard cells are modeled as a 3π network by the layout tool. This network model for interconnects gives the accurate delay analysis results compared to other simpler interconnect models. The CMOS models for the gates in the circuit are taken from 90nm [105] Predictive Technology Model (PTM).

Victim path we examine is highlighted in Figure 3.1(a). Cross-coupling capacitances (C_{X1}, C_{X2}, C_{X3}, C_{X4}, C_{X5}, C_{X6}) that causes crosstalk noise in this victim path are not generally extracted by the tools [108]. This capacitance can only be estimated based on the aggressor net proximity from the victim net. We identify all the aggressor nets affecting the entire victim path from the layout. As there are many aggressors [103] with varying signal transitions and with different arrival times, the path delay alters to a greater extent.

Sketch of s27 circuit layout is shown in Figure 3.2 to further detail the crosstalk

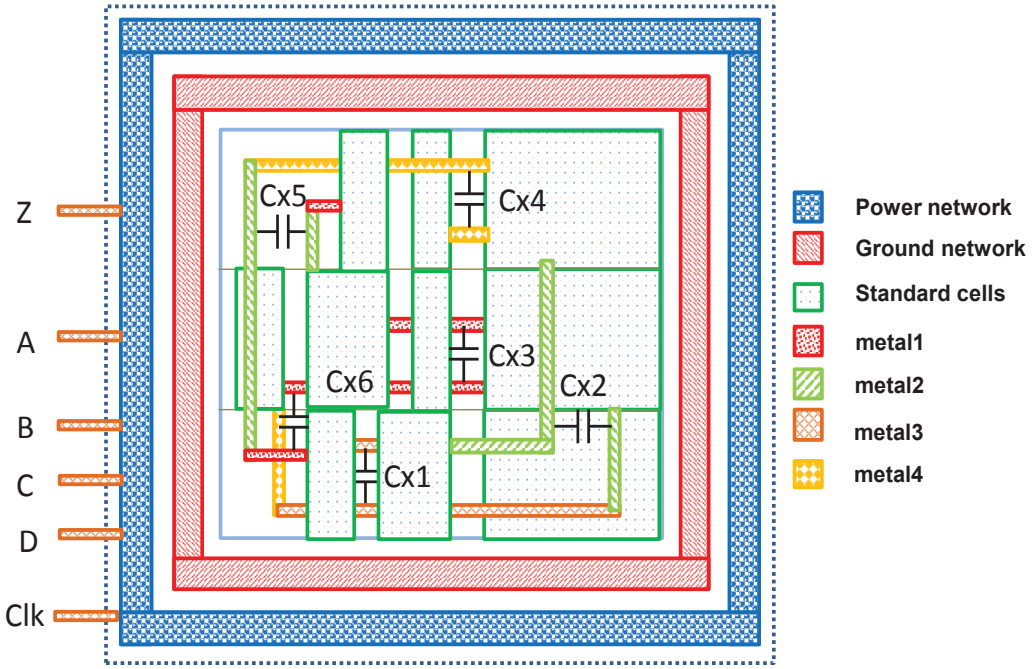


Figure 3.2: Layout sketch of s27 circuit

effect on a victim path. It can be seen from the sketch that the victim nets, as well as, the aggressor nets are routed in multiple metallic layers (metal1, metal2, metal3 and metal4). Also, C_{X4} and C_{X5} are the cross-coupling capacitances between a victim net (but resided in different metal layers) and different aggressor nets. This capacitance varies depending on the area of the aggressor net and their distance apart, so as, their degree of impact on the victim net. From all these, we can interpret that (1) a victim path can have many interconnecting victim nets, (2) a single victim net can get affected by many aggressor nets (3) parts of the aggressor nets or victim nets have resided in different metallic layers connected by vias, and (4) each aggressor net may have varying signal transition with respect to the corresponding victim net. We estimated crosstalk capacitances and included them during path delay analysis. Then computed victim path delay for all possible signal transitions and in varying arrival time at their circuit inputs. Figure 3.3 is derived from the actual circuit layout of s27 circuit.

The nominal power supply voltage, the ground reference voltage and switching frequency of 1V, 0V and 1GHz, respectively are used in this experiment. Path delay variations are observed at the target output FF3/D with the trigger input at the start of combinatorial victim path FF2/Q, refer Figure 3.1(a). All the possible vector pair signal transitions (test patterns) are applied at the circuit's primary input (A, B, C, D) and

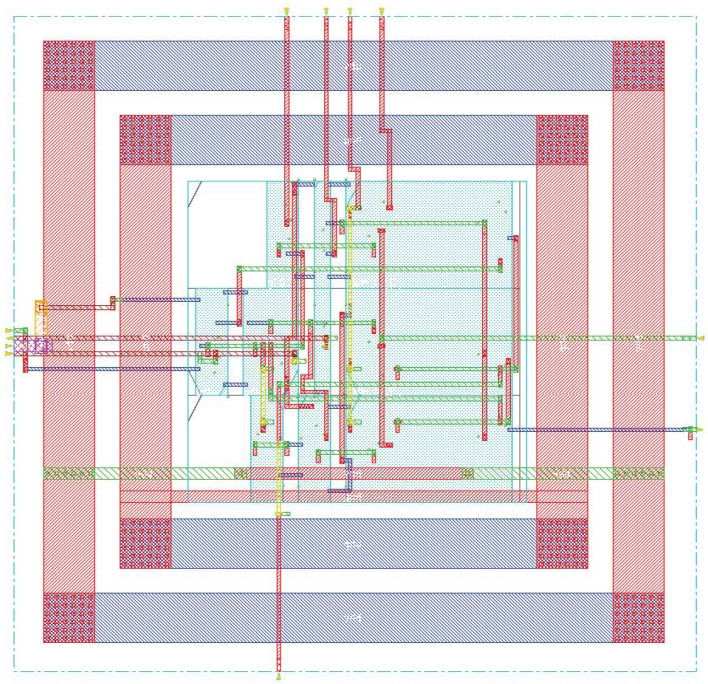


Figure 3.3: Layout of s27 circuit

scan-input (SI). SI's can be applied after implementing the DFT scan chains. Signal transitions such as rising signal (Rise), falling signal (Fall) and stable 0 (S0) and stable 1 (S1) condition are given at their inputs. Along with them, different input arrival times of {-200ps 0ps +200ps} based on the setup and hold timing constraints (10% tolerance with the fixed arrival time) were given. The +/- indicates the advance and the lag in the arrival time.

Table 3.1: Path delay variation due to the impact of crosstalk

Inputs	Signal transitions	Arrival time	Delay variation	Delay impact
A	Rise, Fall, S0, S1	{-200ps, 0ps, 200ps}	-3.59% to +5.07%	speedup(-ve) / slowdown(+ve)
B				
C				
D				
SI				

The combinations of all possible permutations of crosstalk noise, vector pair signal transitions and different arrival times result in a large data set. Therefore, the minimum to maximum path delay variations obtained in comparison with nominal delay (i.e., without crosstalk noise) is only shown in column 4 of Table 3.1. For a simple circuit like s27, the

maximum path delay variation of +5.07% is obtained. This indicates the importance of considering the impact of multi-aggressor crosstalk noise during the path delay analysis. The vector pairs (V1, V2) to the flip-flops (FF1, FF2, FF3) identified after the SPICE simulations, their worst-case path delay (δ_1) are listed in column 2-5 of Table 3.2.

3.4.2 Path Delay Fault Testing

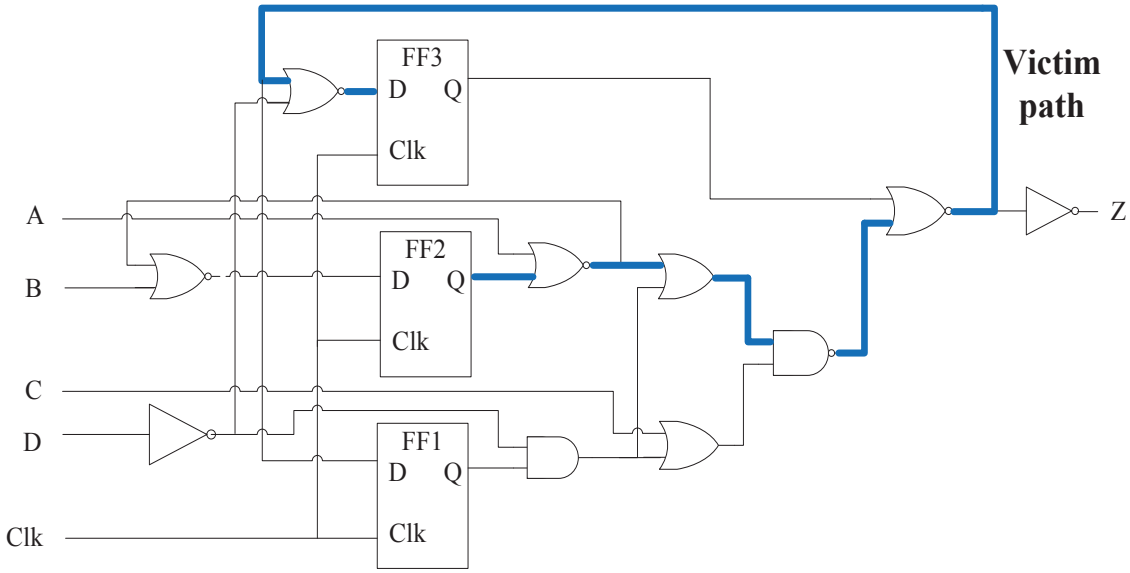


Figure 3.4: Path delay fault testing in TetraMAX

For path delay fault testing, we developed a sequential circuit module in Verilog (with scan-chains inserted) similar to the one shown in Figure 3.1(a), refer Figure 3.4. This module is delay tested with the existing model in the ATPG tool, TetraMAX® at 1GHz clock frequency. Gate and flip-flop models were derived from the 90nm standard cell library. As the tool doesn't allow to include interconnect models during pattern generation, so it is not possible to analyze crosstalk-induced delay faults. Vector pairs were generated for path delay fault test for the victim path from FF2/Q to FF3/D. Column 6-9 of Table 3.2 lists, the ATPG tool generated vector pairs (V1, V2) for the flip-flop (FF1, FF2, FF3) inputs and the worst-case path delay (δ_2) measured in SPICE by the respective vector pair.

3.4.3 Comparison of Vector Pairs

In this subsection, we compare the vector pairs identified from SPICE simulation with the one generated by the ATPG tool and their mismatches are highlighted.

Table 3.2: Pattern comparison and delay variation

	SPICE				ATPG				
	FF 1	FF 2	FF 3	δ_1 (ns)	FF 1	FF 2	FF 3	δ_2 (ns)	Δ_v (%)
V1	**1**	0	1	0.289	**1**	0	1	0.286	1.05
V2	**1**	1	0		**0**	1	0		

$$\Delta_v = \frac{(\delta_1 - \delta_2)}{\delta_1} \times 100 \tag{3.1}$$

In Table 3.2, we can see a mismatch in the vector pair at FF1 between the SPICE and ATPG patterns. The path delay variation (Δ_v) in equation 3.1, is the mean delay difference between the worst-case delays obtained from SPICE (δ_1) and ATPG (δ_2) patterns. The Δ_v of 1.05% (slowdown impact on the victim path) listed in column 10, implies that a path delay fault test is not performed with a good pattern by ATPG. This path delay is quite high for a small circuit like s27 and it may be even higher and at an unacceptable percentage in bigger circuits. Thus, we provide an evidence that a circuit may escape the test to detect crosstalk delay defect due to lower quality in the applied vector pair. This indicates a serious notice in customizing the existing path delay fault testing method in ATPG. Therefore, our work emphasizes the need for refining ATPG tools and proposing novel methods to include good patterns that can be utilized for capturing crosstalk noise delay defects. For bigger circuits, there may exist many large sets of test patterns. So, it is desirable to implement a method to include crosstalk noise effects during pattern generation.

3.5 Crosstalk-Aware Test Pattern Generation Method

3.5.1 Xtalk-ATPG Flow for Pattern Generation

Crosstalk-Aware Test Pattern Generation (Xtalk-ATPG) is similar to the physical design aware pattern generation (PDAPG) method shown in seven stages. Here, we consider only the impact of crosstalk noise. Please refer chapter section 2.5 for its description of pattern generation.

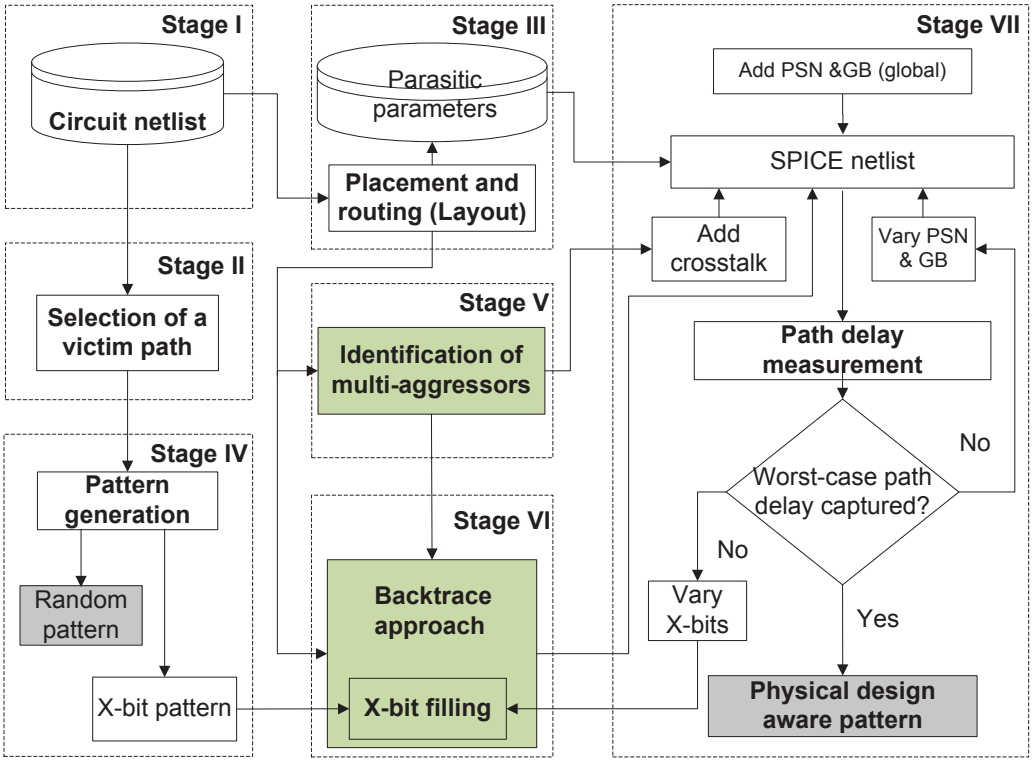


Figure 3.5: Crosstalk-aware pattern generation method

3.5.2 Experimental Results

Xtalk-ATPG method is conducted on ten full-scanned versions of ITC'99 Benchmark [1] circuits with their functionality briefly represented in Table 3.3. The number of gates, scan flip-flops after circuit netlist synthesis and scan chain insertion [53] are respectively shown in column 3 and 4. The total number of victim paths generated [97] are given in column 4. Our major objectives of this experimental analysis are to show the difference in the patterns and the variations in their path delay measured in the presence of crosstalk noise. They are the random pattern generated by the ATPG tool and the pattern identified by our Xtalk-ATPG method. We also, demonstrate that the pattern identified gives better results during the path delay fault test. It is capable of capturing worst-case path delay in the presence of multi-aggressor crosstalk noise on a victim path. Although, this method is implemented on a selected victim path, the same can be applied to any victim path. Selective SPICE simulations are run on a circuit to find a path delay pattern. The details and the experimental results of ITC'99 benchmark circuits are summarized in Table 3.4. In column 2, the number of aggressor nets identified from the layout (mentioned in section 2.5.1.5) is indicated. The total number of X-bits to be filled are given in column 3.

Table 3.3: Functionality of ITC'99 Benchmark circuits [1]

Name	Functionality	# of Gates	# of FFs[a]	# of VPs[b]
b01	FSM that compares serial flows	49	5	5
b02	FSM that recognizes BCD numbers	28	4	4
b03	Resource arbiter	160	31	30
b04	Compute min and max	737	66	66
b05	Elaborate the contents of a memory	998	33	34
b06	Interrupt handler	56	8	8
b07	Count points on a straight line	441	41	41
b08	Find inclusions in sequences of numbers	183	21	21
b09	Serial to serial converter	170	27	28
b10	Voting system	206	17	17
b11	Scramble string with variable cipher	770	30	30
b12	1-player game (guess a sequence)	1076	119	119
b13	Interface to meteo sensors	362	45	45
b14	Viper processor (subset)	10098	215	215
b15	80386 processor (subset)	8922	415	415
b17	Three copies of b15	32326	1311	1311
b18	Two copies of b14 and two of b17	114621	2754	2754
b19	Two copies of b14 and two of b17	231320	5510	5510
b20	A copy of b14 and a modified version of b14	20226	429	429
b21	Two copies of b14	20571	429	429
b22	A copy of b14 and two modified versions of b14	21772	611	611

[a]number of scan flip-flops, [b]number of victim paths.

Aggressor nets are backtraced to fill the relevant X-bits in the test pattern with their count mentioned in column 4.

We demonstrate Xtalk-ATPG method based on b06 benchmark circuit. Eight victim paths were generated for b06 circuit netlist. There exists many path delay fault class [113] based on the sensitization of a victim path. From the summary of the path delay fault class, they are classified as ATPG untestable (3 paths), ATPG undetected (1 path) and

Table 3.4: Circuit description and experimental results for ITC'99 Benchmark circuits

Ckt	#Anet[a]	#X-bits[b]	#X-bits filled[c]	Δ^{d}_{X-bit}	t^{e}_{r}	Δ^{f}_{Xtalk}
b01	8	2	1	50.00%	40s	18.23%
b02	10	2	1	50.00%	25s	12.77%
b03	11	31	10	99.99%	170342s	27.64%
b06	6	6	4	75.00%	353s	24.07%
b07	47	35	16	99.99%	61846978s	40.57%
b08	22	26	14	99.97%	2910945s	33.45%
b09	13	10	3	99.21%	1298s	36.06%
b10	27	22	10	99.97%	185661s	34.81%
b12	16	106	8	99.99%	2353725s	48.34%
b13	10	47	7	99.99%	54508s	41.33%
Average				87.41%		31.73%

[a]number of aggressor nets, [b]number of X-bits, [c]number of X-bits filled, [d]% reduction in input pattern combinations (with X-bits), [e]runtime for testing all patterns based on Xtalk-ATPG method (in sec), [f]path delay variation (crosstalk noise)

ATPG detected (4 paths). The ATPG detected paths were further analyzed for robust (3 paths) and non-robust (1 path) paths. Among them, we select a robust path for our path delay fault analysis. This ensures that at least a single delay fault is detected during the launch-off-capture scheme of pattern generation. Then the robust path (victim path under consideration) is path delay fault tested in ATPG tool. A random pattern and an X-bit pattern were generated as shown in Table 3.5. These patterns are given to the primary inputs (PI) and the scan flip-flops inputs (SI) of the circuit. This random pattern is kept as a reference to compare with the X-bits filled (in Xtalk-ATPG method) pattern in terms of worst-case path delay and the computational time.

All the cross-coupling capacitance's (measured 0.5fF in total) were individually calculated between the identified multi-aggressors and the victim nets. By our backtrace approach, the relevant X-bits (indicated by small letter **'x'**) in the X-bit patterns that control aggressor nets (by inducing crosstalk noise), thereby affecting the victim nets were

Table 3.5: b06 input patterns

Pattern	Scan FF input								Primary input					
type	FF1	FF2	FF3	FF4	FF5	FF6	FF7	FF8	PI1	PI2	PI3	PI4	PI5	PI6
random	1	0	0	1	1	0	0	0	1	0	0	1	1	0
X-bit	x	x	0	1	1	0	X	X	1	0	0	x	1	x
Xtalk-ATPG	1	1	0	1	1	0	X	X	1	0	0	1	1	1

also detected. They are the first two X-bits in the scan flip-flop input and both the X-bits in the primary input (those colored in Table 3.5). In other circuits, some of the relevant X-bits may be already filled by the tool, so they are not modified. Xtalk-ATPG method minimizes the complexity by 75% in delay fault testing by applying reduced X-bit input pattern combinations (i.e., 2^4(4 X-bits filled) = 16 instead of 2^6(6 X-bits) = 64). By this way, we selectively provide the pattern combinations for path delay testing. Computation for the percentage reduction in the X-bit filled pattern (Δ_{X-bit}) combinations given at the circuit input is based on the equation:

$$\Delta_{X-bit} = \frac{2^{\#X-bits} - 2^{\#X-bitsfilled}}{2^{\#X-bits}} \times 100 \tag{3.2}$$

Equation 3.3 gives the computational time for testing all combinations of input patterns (t_r) for crosstalk noise. This time includes the time taken for the generation of the X-bits filled pattern (t_{atpg}) and for performing their corresponding SPICE simulation (t_{SPICE}). The computational time taken by the different benchmark circuits are shown in column 6 of Table 3.4.

$$t_r = (t_{atpg} + t_{SPICE}) \times 2^{\#X-bitsfilled} \tag{3.3}$$

Nominal path delay (δ_n) of 216ps is measured for the random pattern without considering crosstalk noise. After considering the impact of multi-aggressors, 268ps crosstalk delay (δ_{xtalk}) is measured for the X-bit filled (Xtalk-ATPG) pattern. The path delay variation for the Xtalk-ATPG pattern, shown in the last row of Table 3.5 is 24.07% greater than the nominal path delay; hence this is a high quality input pattern that should be identified during path delay fault testing. The high quality test pattern signifies that this pattern gives the worst-case path delay, when applied to the selected victim path. Path delay variation is computed by:

$$\Delta_{Xtalk} = \frac{(\delta_n - \delta_{xtalk})}{\delta_n} \times 100 \tag{3.4}$$

These details of the other circuits are respectively shown in column 5, 6 and 7 of Table 3.4. By implementing our Xtalk-ATPG method on the ITC'99 benchmark circuits, we were able to obtain the following: 1) an average reduction of test pattern count by 87.41% for the 10 ITC'99 benchmark circuits mentioned, and 2) could capture an average path delay variation of 31.73%.

The objective of Table 3.6 is to show the variation between the random pattern obtained from ATPG tool and the pattern identified by our Xtalk-ATPG method. They are separately mentioned for scan inputs (SI) and the primary inputs (PI). X-bit patterns, random patterns and Xtalk-ATPG patterns are respectively shown in the 1^{st}, 2^{nd} and 3^{rd} rows of each circuit. The mismatch between the patterns can be clearly distinguished. The worst-case path delay variation between the random pattern and the Xtalk-ATPG pattern for the selected victim path is also quite high. Ignoring Xtalk-ATPG pattern may cause path delay defect escape during path delay fault testing. Thus, we show that the Xtalk-ATPG method is an accurate and efficient delay testing strategy for ensuring better delay defect coverage in the presence of multi-aggressor crosstalk. The scripts for aggressor net identification, cross-coupling capacitance calculation, backtrace approach and testing with different combinations of patterns were implemented in PERL. These experiments are performed on Linux ×86 64bit servers with 48 sockets and 1 core per socket and 176GB of available memory.

We acknowledge that the proposed approach is accurate after performing a detailed timing analysis but is also limited in their computational time needed to test all the pattern combinations of the generated X-bit pattern and further their selective SPICE simulation. This is due to the high volume of X-bit filled pattern combinations given during SPICE-level simulation. Minimizing the computational time for generating a crosstalk-aware path delay pattern is the focus of our following work.

3.6 Constrained ATPG (C_{atpg}) Method

In this section, we present a novel flow of pattern generation that can be effective in capturing crosstalk-induced delay defects. The assurance with this flow is the reduced

Table 3.6: Input pattern comparison results of ITC'99 Benchmark circuits

Ckt	Input	SI	PI	Δ_{Xtalk}
b01	X-bit pattern	001x0	11000X	18.23%
	Random pattern	01110	110001	
	Xtalk-ATPG pattern	00100	11000X	
b02	X-bit pattern	x000	0000X	12.77%
	Random pattern	1000	00001	
	Xtalk-ATPG pattern	0000	0000X	
b03	X-bit pattern	X0XX X1XX XXXx XxXx x0XX xXXX xxxx XX	001X XxX0	27.64%
	Random pattern	0010 1110 0010 0101 1001 0100 0111 10	0011 1100	
	Xtalk-ATPG pattern	X0XX X1XX XXX1 X1X0 10XX 1XXX 0100 XX	001X X1X0	
b06	X-bit pattern	xx01 10XX	100x 1x	24.07%
	Random pattern	1001 1000	1001 10	
	Xtalk-ATPG pattern	1101 10XX	1001 11	
b07	X-bit pattern	0XXX xXxx XxxX xXXx xx01 1xxX XxXX X100 1xxX xXXX X	X000x	40.57%
	Random pattern	0100 1011 0001 0010 1101 1010 1000 1100 1111 0101 0	10001	
	Xtalk-ATPG pattern	0XXX 1X01 X10X 0XX0 1001 101X X1XX X100 110X 0XXX X	X0000	
b08	X-bit pattern	xXx1 XxxX xxxx XxxX XxX1 1	001X xxX0 XXx0 X	33.45%
	Random pattern	0010 0010 0101 1010 1100 0	0011 1100 0100 1	
	Xtalk-ATPG pattern	1X11 X11X 1111 X01X X1X1 1	001X 10X0 XX00 X	
b09	X-bit pattern	1X00 110X 111X XxxX X101 1001 1111	0X0x X	36.06%
	Random pattern	1100 1100 1111 0010 1101 1001 1111	0000 1	
	Xtalk-ATPG pattern	1X00 110X 111X X11X X101 1001 1111	0X01 X	
b10	X-bit pattern	XXX0 xxx1 0011 Xxxx x	XXXx 0XX1 0xxX X0X	34.81%
	Random pattern	0100 0101 0011 1101 0	1110 0011 0011 000	
	Xtalk-ATPG pattern	XXX0 1101 0011 X110 1	XXX0 0XX1 001X X0X	
b12	X-bit pattern	011X 1XXX XXX0 0XX0 XxxX XXXX XXXX xXXX XXXX 1XXX XXXX xXXx XXXX X1XX XXXX 1XXx XXXX XXXX XXXX XX0X XXXX XXX1 XXXX XXXX X00X 0XXX 1XXX XXXX XX0X XX1	000X xXx0 X	48.34%
	Random pattern	1101 1000 1000 0101 1011 0101 0001 1110 1010 1100 1001 0111 0101 0001 0011 0111 1010 1100 1110 0001 1001 1100 1110 0110 0100 0101 1111 0010 0100 001	0001 1100 0	
	Xtalk-ATPG pattern	011X 1XXX XXX0 0XX0 X11X XXXX XXXX 1XXX XXXX 1XXX XXXX 1XX0 XXXX X1XX XXXX 1XX0 XXXX XXXX XXXX XX0X XXXX XXX1 XXXX XXXX X00X 0XXX 1XXX XXXX XX0X XX1	000X 1X10 X	
b13	X-bit pattern	0110 0111 xXXX 1XXX XX1X X01X X0XX XXXX XxXX XxxX XXXX XXXX X	0X0X xXXx XXxX 0X	41.33%
	Random pattern	0110 0111 0100 1101 1010 1000 1111 0101 0110 0100 1011 1100 1	0101 1001 0110 00	
	Xtalk-ATPG pattern	0110 0111 0XXX 1XXX XX1X X01X X0XX XXXX X0XX X10X XXXX XXXX X	0X0X 1XX0 XX0X 0X	

computational time in identifying a path delay pattern for testing.

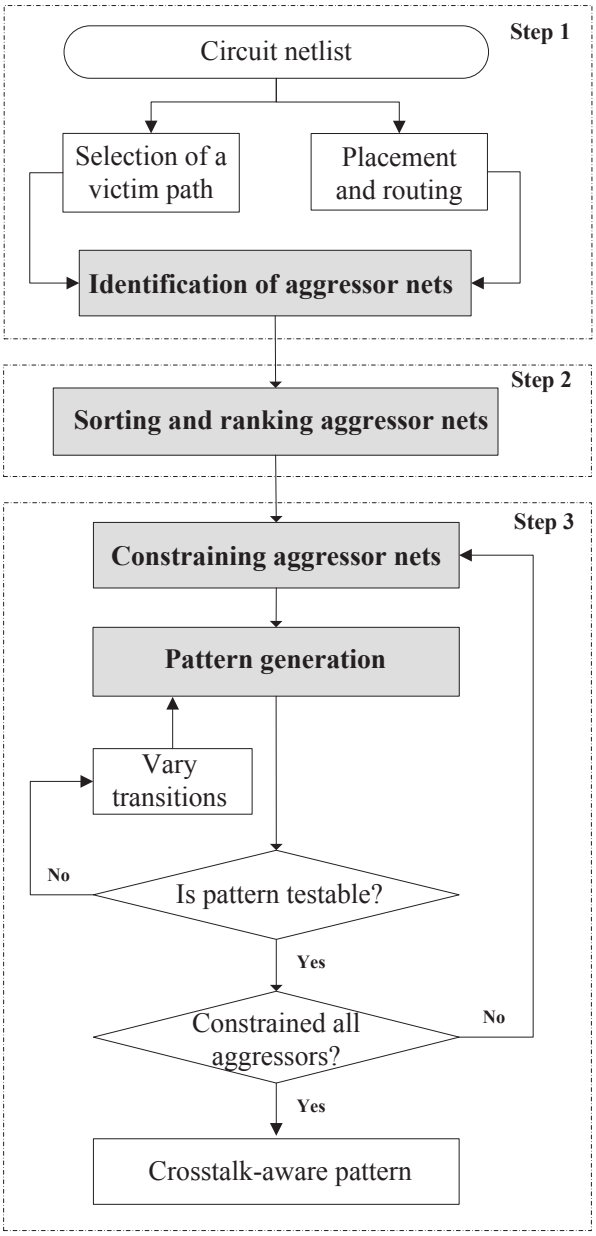


Figure 3.6: Constrained ATPG flow

3.6.1 C_{atpg} Flow for Pattern Generation

The C_{atpg} flow shown in Figure 3.6 is described in three stages: (1) identification of aggressor nets (to a victim path), (2) sorting and ranking aggressor nets (based on their impact), and (3) constraining aggressor nets and pattern generation (for a victim path). C_{atpg} method is a generic method and this method is presented on a sample circuit b08, an ITC'99 benchmark circuit. The relevance of the proposed flow shows a great promise to generate an effective worst-case path delay pattern similar to the patterns identified by the Xtalk-ATPG method mentioned in section 2.5. Even though, the latter method was

based on selective SPICE simulations (i.e., by selective X-filling), it takes higher runtime for bigger circuits.

3.6.1.1 Stage I : Identification of Aggressor Nets

We utilize industrial tools such as Cadence Encounter RC Compiler [53], Synopsys Prime-Time Static Timing Analysis (STA) tool [97] and Cadence SOC Encounter tool [108] to generate a circuit netlist, select a victim path and to perform placement and routing of the circuit netlist, in their respective order. From the circuit layout, the aggressor nets were determined for all the victim nets in the victim path. This is based on the separation between two nets (140μm), the degree of the aggressor net impact on a victim net and the width of the aggressor nets (for metal 1, 120μm and for metal2, 140μm were considered). Each aggressor net may reside in different metal layers with varying length, this information is fetched from the DEF (Design Exchange Format) file. Therefore, determining all the aggressor nets from the layout of the circuit gives a better estimate of the impact of crosstalk noise on path delay. We select a strongly robust victim path (as described in section 2.5.1.2) from b08 circuit to test our method. 22 aggressor nets were identified for the 5 victim nets, their notations are given in Table 3.7. Some of the aggressor nets were neglected, if it is (1) a clock net, or (2) a victim net acting itself as an aggressor net to another victim net, or (3) the same aggressor net affecting two different victim nets in the victim path. If it is the last case, then the choice is made based on their degree of impact on the victim net.

3.6.1.2 Stage II : Sorting and Ranking

Firstly, the identified aggressor nets (as shown in Figure 3.7) are sorted based on their impact on the victim net, i.e., the higher cross-coupling capacitance between them will have a higher impact. Capacitance values were measured between 0.02fF - 2fF for a b08 circuit path.

Secondly, we rank the aggressor nets (as shown in Figure 3.8), i.e., nets with lower capacitances are given a lower rank (Rank 22) as they have only a minimal influence on the victim path delay [114]. Aggressor nets with higher capacitance were given higher rank (Rank 1). Their ranking order is shown in column 3 of Table 3.7.

Table 3.7: Aggressor net ranking

Victim net	Aggressor net	Rank
Vn1	An1	12
	An2	20
	An3	6
	An4	13
Vn2	NA*	
Vn3	An5	2
	An6	1
	An7	10
	An8	16
	An9	18
	An10	15
	An11	21
	An12	9
	An13	7
	An14	17
	An15	5
Vn4	An16	19
	An17	11
	An18	4
	An19	3
	An20	14
	An21	8
Vn5	An22	22

NA* - No aggressor net for this victim net.

3.6.1.3 Stage III : Constraining Aggressor Nets and Pattern Generation

Stage III is our major contribution in the constrained ATPG method. We customize the existing ATPG tool, TetraMAX® [10] to generate a crosstalk-aware pattern that can

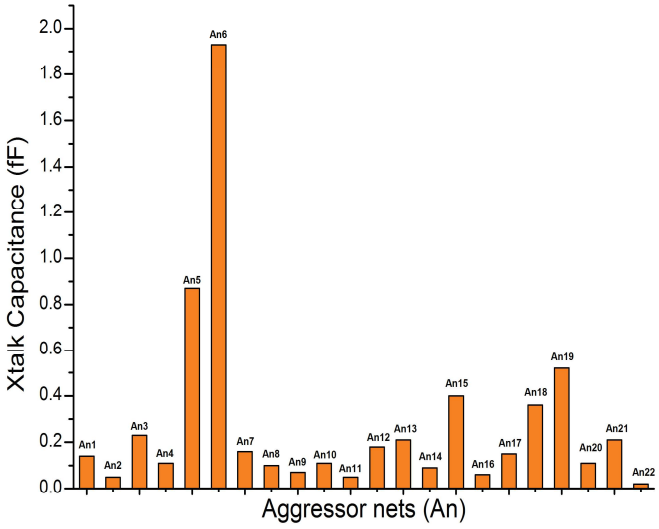


Figure 3.7: Aggressor nets

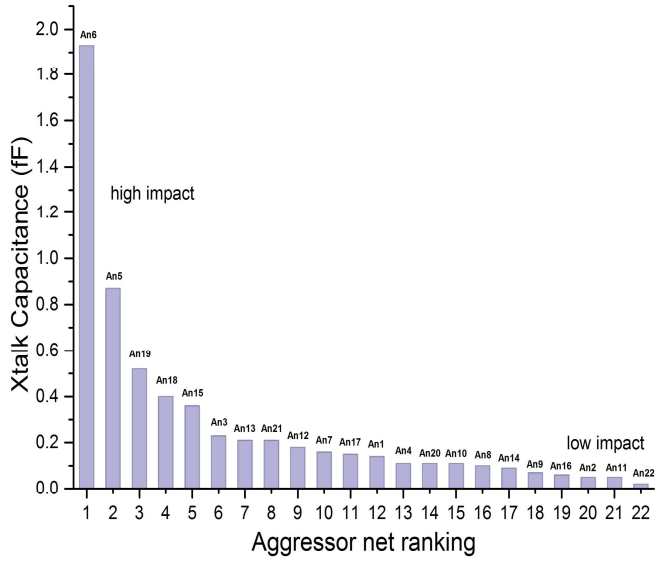


Figure 3.8: Aggressor net ranking

detect a path delay fault. This ATPG tool is unaware of the impact of crosstalk noise on victim path delay. Therefore, we aim at generating a crosstalk-aware pattern by giving constraints to the aggressor nets that in turn impact the victim path delay. In general, crosstalk delay defect can be determined beforehand by adding their impact during exhaustive or selective SPICE simulations, but it is computationally very expensive. We

compare the proposed constrained ATPG method with the earlier mentioned Xtalk-ATPG method [115] (section 2.5) in terms of (1) computational time in identifying a crosstalk-aware path delay pattern, and (2) validating the effectiveness of this pattern by evaluating their worst-case path delay. Xtalk-ATPG method is based on ATPG and selective SPICE simulation in contrast to the constrained ATPG method (based exclusively on ATPG). In the latter method, SPICE simulation is merely used to compare the worst-case path delays.

Table 3.8: Aggressor net transition and delay measurement

Rank	Vnet[a]	C_v^b	Anet[c]	C_1^d	FC[e]	Δ_v^f	C_2^g	FC	Δ_v	C_3^h	FC	Δ_v
1	Vn3	R[i]	An6	F[j]	UD/AU[k]		**0**	**DT**[l]	**0%**			
2	Vn3	R	An5	F	UD/AU		**0**	**DT**	**0%**			
3	Vn4	F	An19	R	UD/AU		0	UD/AU		**1**	**DT**	**0%**
4	Vn4	F	An18	R	UD/AU		0	UD/AU		**1**	**DT**	**-0.04%**
5	Vn3	R	An15	F	UD/AU		**0**	**DT**	**-0.04%**			
6	Vn1	R	An3	F	UD/AU		**0**	**DT**	**-0.04%**			
7	Vn3	R	An13	F	UD/AU		0	UD/AU		**1**	**DT**	**-0.04%**
8	Vn4	F	An21	**R**	**DT**	**-0.68%**						
9	Vn3	R	An12	F	UD/AU		**0**	**DT**	**-0.70%**			
10	Vn3	R	An7	F	UD/AU		**0**	**DT**	**-0.72%**			
11	Vn4	F	An17	R	UD/AU		0	UD/AU		**1**	**DT**	**-0.76%**
12	Vn1	R	An1	F	UD/AU		**0**	**DT**	**-0.79%**			
13	Vn1	R	An4	F	UD/AU		**0**	**DT**	**-0.79%**			
14	Vn4	F	An20	R	UD/AU		0	UD/AU		**1**	**DT**	**-0.80%**
15	Vn3	R	An10	F	UD/AU		0	UD/AU		**1**	**DT**	**-0.80%**
16	Vn3	R	An8	F	UD/AU		0	UD/AU		**1**	**DT**	**-0.94%**
17	Vn3	R	An14	F	UD/AU		**0**	**DT**	**-0.97%**			
18	Vn3	R	An9	**F**	**DT**	**-1.59%**	0	DT				
19	Vn4	F	An16	R	UD/AU		**0**	**DT**	**-1.71%**			
20	Vn1	R	An2	F	UD/AU		0	UD/AU		**1**	**DT**	**-1.73%**
21	Vn3	R	An11	F	UD/AU		0	UD/AU		**1**	**DT**	**-1.73%**
22	Vn5	F	An22	R	UD/AU		0	UD/AU		**1**	**DT**	**-1.73%**

[a]victim nets, [b] transition in the victim net, [c]aggressor nets, [d]contraining aggressor net with opposite trasition, [e]fault class, [f]path delay variation for a constrained ATPG pattern, [g]contraining aggressor net with stable 0, [h]contraining aggressor net with stable 1, [i]indicates a rising transition, [j]indicates a falling transition, [k]ATPG fault that is either undetectable or untestable, [l]ATPG detected fault class that is testable.

In this stage 3, we constrain all the aggressor nets one after the other depending on the signal transition in their corresponding victim net, shown in Figure 3.9.

The aggressor net with the higher rank is constrained first based on their impact. For a victim net with a Rising (R) signal transition, an opposite signal transition in the aggressor net is given initially, i.e., a Falling (F) transition can maximize the crosstalk-induced delay slowdown [111]. Then we check whether it generates a testable pattern or not; this ensures that a given signal transition is propagated during the launch clock cycle in an at-speed test [116]. Sometimes, this gives an unsuccessful pattern generation. Therefore, we constrain the victim net again with a stable 0 and then again with stable

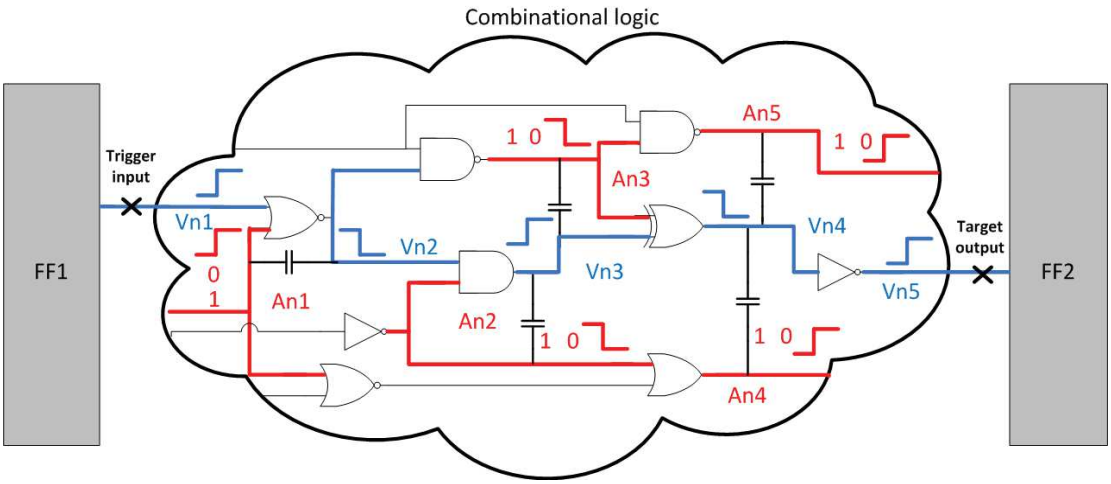


Figure 3.9: Constrained ATPG method

1 condition at the aggressor net, if the former generates an untestable pattern. The same direction signal transition in the aggressor net is ignored, as it may only induce a minimal delay slowdown or a speedup in the victim net. All aggressor nets are constrained one after the other until a final pattern is generated. This can be utilized for path delay fault testing in the presence of multi-aggressor crosstalk noise. Instead of constraining or modifying the victim net, we constrain all the aggressor nets simultaneously for pattern generation. This is the major thrust in this novel method.

Table 3.8 shows the results after applying the C_{atpg} ATPG flow on b08 benchmark circuit. The details are as follows. Aggressor net ranking in column 1, victim net, their signal transition pattern (i.e., R for rising signal and F for falling signal) in column 2-3, their corresponding aggressor net in column 4, their signal transition (i.e., opposite transition F or R, stable 0, or stable 1), their fault class and path delay variation are mentioned in subsequent columns.

There exist several fault class for testing a path delay fault [113], such as detected (DT), possibly detected (PT), undetectable (UD), not detected (ND) and ATPG untestable (AU); categorized based on the signal transition propagation, fault detection and pattern generation. Among them, we select the detected faults so that a signal transition in the victim net is propagated during the launch clock cycle. Then, we measure path delay in the victim net (Δ_v) in SPICE. This is to show the effectiveness of a pattern in determining the worst-case delay in a path. By simulating this pattern, we obtain the maximum possible worst-case delay of -1.73% on the victim path. Negative value shows the slowdown impact and a positive value notate a speedup in the victim path delay.

Table 3.9: Pattern comparison and Path delay variation

Scan	C^b_{atpg}		Xtalk-ATPG[c]		Aggressor	C_{atpg}		Xtalk-ATPG	
FF#[a]	V1	V2	V1	V2	nets[d]	V1	V2	V1	V2
FF1	1	X	X	X	An6	**0**	**0**	**X**	**X**
FF2	X	1	1	0	An5	0	0	0	0
FF3	X	X	0	X	An19	1	1	1	1
FF4	1	0	1	0	An18	**1**	**1**	**X**	**X**
FF5	0	1	X	0	An15	**0**	**0**	**X**	**X**
FF6	X	X	X	X	An3	0	0	0	0
FF7	X	X	X	X	An13	1	1	1	1
FF8	1	X	X	X	An21	**0**	**1**	**X**	**X**
FF9	0	0	X	0	An12	**0**	**0**	**X**	**X**
FF10	X	0	0	0	An7	**0**	**0**	**X**	**X**
FF11	0	0	0	0	An17	**1**	**1**	**X**	**X**
FF12	X	0	0	0	An1	**0**	**0**	**X**	**X**
FF13	X	0	0	0	An4	**0**	**0**	**X**	**X**
FF14	X	0	1	0	An20	**1**	**1**	**X**	**X**
FF15	X	0	1	0	An10	**1**	**1**	**X**	**X**
FF16	0	0	1	1	An8	1	1	1	1
FF17	1	0	0	0	An14	0	0	0	0
FF18	X	X	0	0	An9	**1**	**0**	**X**	**0**
FF19	X	X	1	1	An16	**0**	**0**	**1**	**1**
FF20	1	0	1	0	An2	0	1	0	1
FF21	**0**	**1**	**0**	**1**	An11	1	1	1	1
Δ^e_v	-1.73%		-2.12%		An22	1	1	1	1

[a]scan flip-flops from 1 to 21 from the synthesized DFT scan chain structure, [b]by constrained ATPG method, [c]by Xtalk-ATPG method, [d]all the 22 aggressor nets to the victim path, [e]path delay variation.

The path delay variation (Δ_v) for the constrained ATPG and Xtalk-ATPG methods were computed using the equations below:

$$\Delta_v(C_{atpg}) = \frac{(\delta_r - \delta_{Catpg})}{\delta_r} \times 100 \tag{3.5}$$

$$\Delta_v(Xtalk - ATPG) = \frac{(\delta_r - \delta_{Xtalk-ATPG})}{\delta_r} \times 100 \tag{3.6}$$

Here δ_r is the crosstalk delay due to a random pattern generated by ATPG; δ_{Catpg} and $\delta_{Xtalk-ATPG}$ are the worst-case path delays measured by the patterns obtained from the two methods C_{atpg} and Xtalk-ATPG respectively. ATPG generates a random pattern to test path delay fault. This pattern is kept as the reference for path delay fault detection and worst-case path delay comparison.

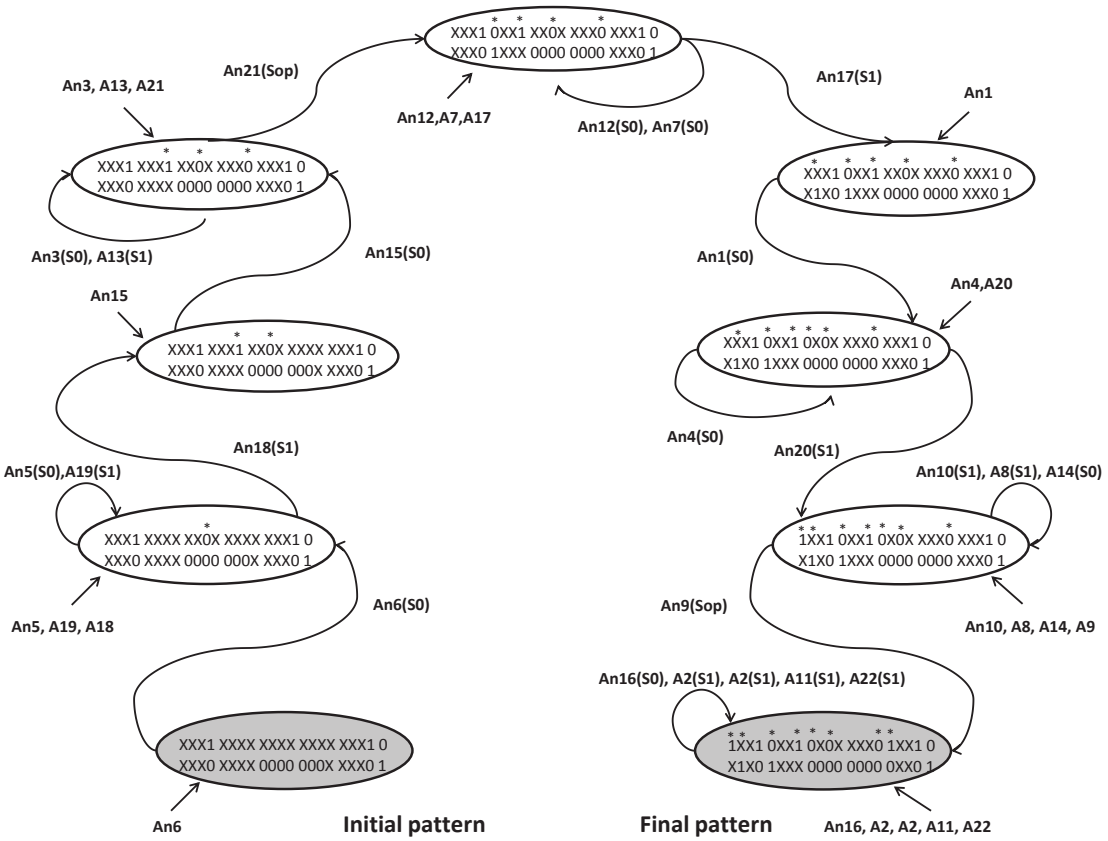


Figure 3.10: State diagram of constrained ATPG

The b08 circuit has 13 primary inputs and 21 scan flip-flop inputs (FF1 - FF21), varying these inputs can sensitize the victim path to generate a crosstalk-aware pattern. We describe explicitly one of the longest combinatorial victim path by giving trigger input at FF21 and observing its trigger output at FF4. According to the scan based launch-off-capture scheme, the scan inputs are shifted in. Then, the vector pair (i.e., test patterns)

$< V1, V2 >$ i.e., $< 01011, 10100 >$ are given as the circuit's primary input for initiating a signal propagation in the victim path. In Table 3.9, the patterns generated by C_{atpg} flow and the patterns identified from the Xtalk-ATPG method are compared. Also, path delays from both methods are verified. The path delay variation (Δ_v) obtained from each method is shown in the last row of this Table. By this, we show that C_{atpg} method is able to achieve a delay value (-1.73%) closer to the Xtalk-ATPG method (-2.12%) (refer to equation 3.6), without utilizing SPICE simulations. We, thus show that C_{atpg} method is an effective pattern generation method to detect crosstalk related delay faults.

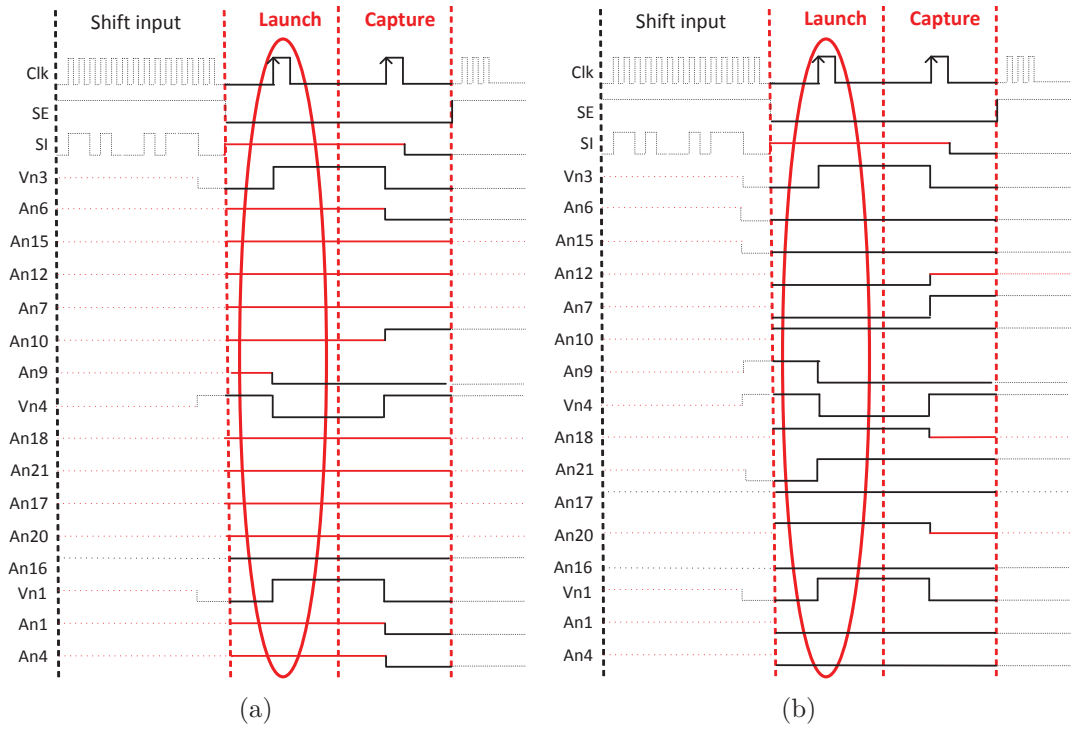


Figure 3.11: (a) Waveform of Xtalk-ATPG pattern (b) Waveform of constrained ATPG pattern

In Figure 3.10, the step by step changes to the DFT scan structure consisting of sequential scan flip-flops from FF1 to FF21 is shown in the form of a state diagram. Aggressor nets are constrained based on their rank and the patterns are generated. Depending on the signal transition in the aggressor net, some of the X-bits are getting filled (by '1' or '0') and the already filled X-bits are either getting modified or remain the same. Compared to the initial pattern, 8 scan flip-flop inputs are modified. SPICE simulation with this new pattern from C_{atpg} can sometimes provide a better capture of worst-case delay than the Xtalk-ATPG method. The latter method is based only on the filing of the X-bits in the test pattern and no changes to the already filled X-bits are made. These

changes in the C_{atpg} method help to define a new worst-case path delay.

In Table 3.9, we also show the modified patterns in the aggressor nets after constraining them. As aggressor nets are either primary input nets, scan input nets or other interconnecting nets between the standard cells, a constrained signal transition in them can force to change the generated pattern. Therefore, constraining these nets with opposite or stable 0, stable 1 signal transition, modifies the pattern generated by the tool. From the vector pair colored in Table 3.9, we can see that 13 aggressor nets are undergoing changes in their signal transition after applying C_{atpg} method. Comparison between the sketched waveforms is shown in Figure 3.11. After the clock, scan enable (SE), scan input (SI) signal, victim net transition is shown, followed by their respective aggressor net transitions. These nets are constrained and the pattern generated based on the victim net transition. Some of the nets that have an X-bit (don't care bit) input are getting filled either by '1'or '0'. The forced filling of these bits aggravates the victim nets to generate good patterns that can be utilized for testing crosstalk noise. Not only the X-bits are getting filled, the already filled bits in the input pattern are also undergoing changes (or getting modified) in order to activate the signal propagation. The newly generated test pattern shows the effectiveness in capturing the worst-case path delay. Launch and capture cycles are highlighted in this figure 3.11. X-bits are indicated in red color.

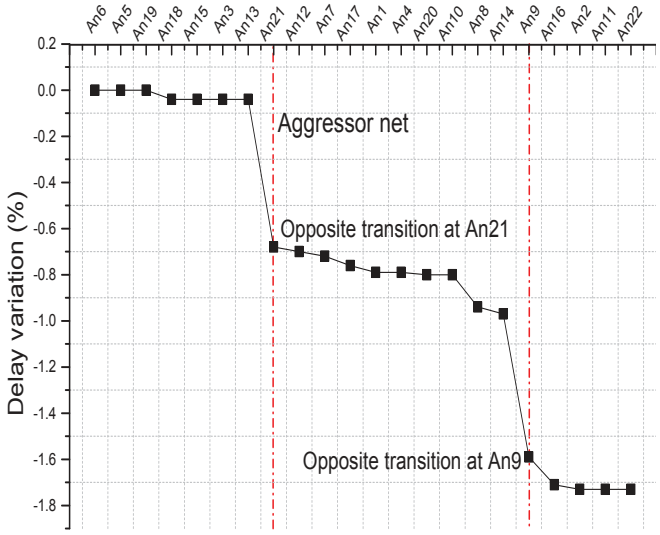


Figure 3.12: Path delay variation plot of a victim path

We show the delay variation in the victim path due to the impact of the aggressor net in Figure 3.12. Aggressors are constrained one after the other based on their rank, as

mentioned in C_{atpg} flow. From the plot, we can see that there are two successful opposite pattern transitions (first at An21 and then at An9) that are highly maximizing the slow-down of victim path delay. Path delay variation is comparatively low after constraining other aggressor nets, as they are stable conditions.

3.6.2 Experimental Results

In this section, we describe the experimental results after applying C_{atpg} flow on 21 full-scan versions of ITC'99 benchmark circuits [1]. Our aim is to generate a pattern that can be effective in path delay fault testing. This pattern is capable of capturing worst-case delay in the victim path in the presence of multiple aggressors with their varying degree of impact. Our method of pattern generation is fully based on ATPG tool. We could save a lot of computational time as this method has not used SPICE or selective SPICE simulations for pattern identification. This flow is scalable and can be applied to any sized circuits.

Table 3.10: Circuit description and experimental results on ITC'99 Benchmark circuits

Ckt	#X bits[a]	#X-bits filled bits[b]	#Anet[c]	Xtalk-ATPG[d]		C_{atpg}^{e}		δ_d^f	δ_t^g
				Δ_v^h	t_r^i	Δ_v^j	t_r^k		
b01	2	1	8	-0.01%	40s[l]	-0.01%	58s	0%	18s(-)
b02	2	1	10	-0.11%	25s	-0.24%	46s	0.13%(+)	21s(-)
b03	31	10	11	-2.06%	170342s	-2.87%	53s	0.81%(+)	170289s(+)
b04	73	22	21	-28.75%	4218924564s	-21.50%	124s	7.25%(-)	4218924440s(+)
b05	29	16	36	-3.34%	9133840s	-1.88%	278s	1.46%(-)	9133562s(+)
b06	6	1	6	-0.17%	353s	-0.17%	30s	0%	323s(+)
b07	35	20	41	-4.92%	818469s	-3.81%	201s	1.11%(-)	818268s(+)
b08	26	14	22	-2.12%	2910945s	-1.73%	104s	0.39%(-)	2910841s(+)
b09	10	3	13	-1.63%	1298s	-1.86%	68s	0.23%(+)	1230s(+)
b10	22	10	27	-2.59%	185661s	-2.23%	188s	0.36%(-)	185473s(+)
b11	21	13	22	-19.68%	5152358s	-14.14%	184s	5.54%(-)	5152174s(+)
b12	106	8	16	-22.30%	2353725s	-20.07%	102s	2.23%(-)	2353623s(+)
b13	47	7	9	-14.62%	54508s	-17.54%	52s	2.92%(+)	54456s(+)
b14	183	71	65	-	1E+26s	-27.93%	493s	-	(+)
b15	443	22	29	-	3E+11s	-17.66%	153s	-	(+)
b17	1334	101	199	-	3E+35s	-35.27%	1880s	-	(+)
b18	2776	133	224	-	4E+45s	-37.02%	4328s	-	(+)
b19	5535	67	72	-	8E+25s	-42.15%	1512s	-	(+)
b20	380	54	108	-	9E+21s	-38.32%	1272s	-	(+)
b21	439	21	42	-	1E+12s	-36.97%	494s	-	(+)
b22	572	126	200	-	4E+43s	-34.84%	1834s	-	(+)

[a]number of X-bits, [b]number of X-bits filled by backtrace approach, [c]number of aggressor nets, [d]by Xtalk-ATPG method, [e]by constrained ATPG method, [f]path delay variation difference between Xtalk-ATPG and C_{atpg} method, [g]computational time difference between Xtalk-ATPG and C_{atpg} method, [h]path delay variation (crosstalk noise) by Xtalk-ATPG method, [i]computational time for Xtalk-ATPG method, [j]path delay variation (crosstalk noise) by C_{atpg} method, [k]computational time for C_{atpg} method, [l]computational time in seconds.

Details of experimental results are summarized in Table 3.10. Number of X-bits in the scan input and primary input pattern are given in column 2. These inputs are generated by the ATPG tool by performing path delay fault test on a victim path. In column 3, the minimum number of relevant X-bits that need to filled in Xtalk-ATPG method is shown. The number of all possible aggressor nets to the selected victim path are given in column 4. Column 5-6 depict the path delay variation and computational time for producing the patterns by Xtalk-ATPG method (selective SPICE simulation). We haven't shown some results on the Xtalk-ATPG method in column 5, as the simulation of a large sized circuit takes days to complete. Similarly, in column 7-8, we show the results obtained after applying C_{atpg} ATPG flow on pattern generation. The path delay difference (δ_d) and the computational time difference (δ_t) obtained after comparing the two methods are shown in subsequent columns and also represented graphically in Figure 3.13 and Figure 3.14. The positive and negative notations in column 9 and 10, express the benefit and the relaxation margins, respectively between both methods. The smaller circuit consumes more time for pattern generation as it is proportional to the total number of aggressors constrained and patterns generated each time. C_{atpg} method indicates that, with a small relaxation in the path delay, we gain a very high margin in computational time.

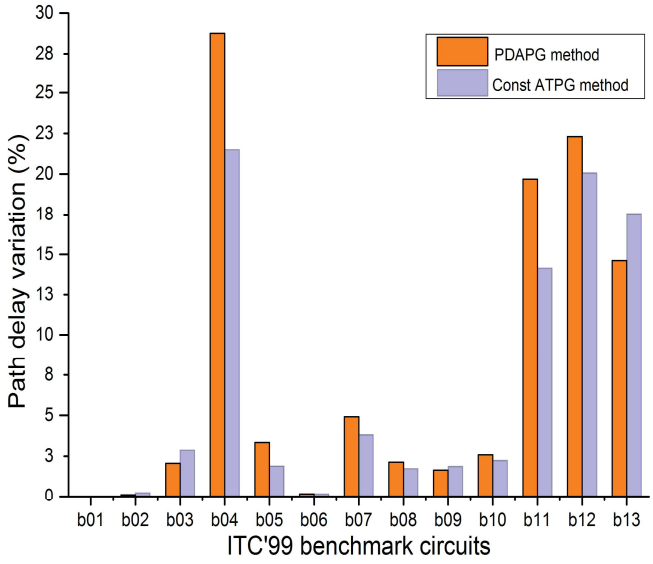


Figure 3.13: Comparison of 2 methods in terms of path delay variation

Our results show that the proposed ATPG flow is able to generate an effective pattern that can provide a delay value nearest to the expected worst-case delay (achieved by

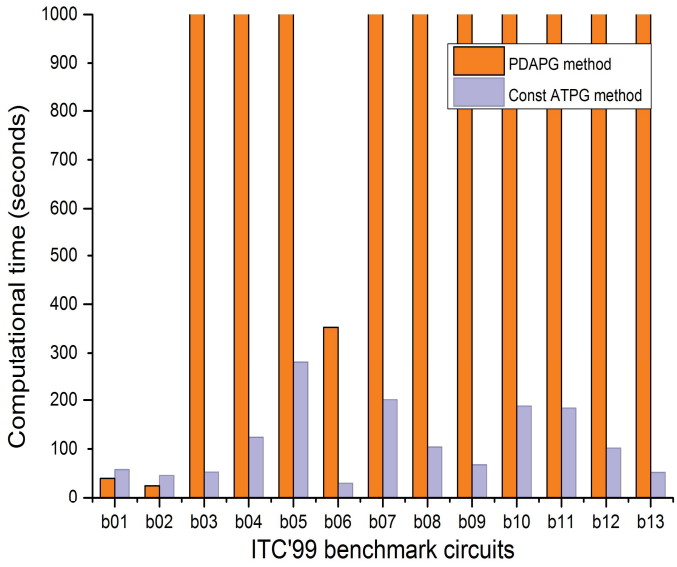


Figure 3.14: Comparison of 2 methods in terms of computational time

using a selective SPICE simulation pattern). Also, we have shown that our flow can be executed with extremely low computational time. For instance, pattern identification on a comparatively bigger circuit like b05 circuit requires 9133840 seconds to estimate a single worst-case delay pattern, whereas this C_{atpg} flow takes only 278 seconds to generate a pattern that can be path delay fault tested in the presence of multi-aggressor crosstalk noise. We acknowledge that the proposed flow is simple to implement, and it is better in terms of computational time and worst-case path delay patterns.

3.7 Summary

In this chapter, we presented a crosstalk-aware pattern generation method for emphasizing the impact of crosstalk noise on path delay of a circuit. This method focuses on identifying a test pattern that can capture worst-case path delay on a victim path. Although, we used a single victim path as our initial path repository, this method can be applied to any paths for identifying a high quality test pattern. Results from this method suggest the refinement of the existing ATPG path delay test methods for incorporating the impacts of crosstalk noise. Further, we proposed a novel flow of constrained ATPG method targeting path delay fault. This method could eliminate the selective SPICE simulation and it produces patterns to test the worst-case path delay in lesser computational time. All the

step by step procedures related to C_{atpg} method are completely automated. Our flow is implemented on ITC'99 benchmark circuits and the results were shown by comparison with SPICE simulation. By this flow, we also show the effectiveness in computational time and the generation of a high quality pattern. The explanation of this method is based on a specific ATPG tool, TetraMAX®, but the same can be adopted in any ATPG tool or method to identify the right set of PDF patterns.

Chapter 4

Delay Probability Metric Under the Impact of Process Variation and Supply Noise

Contents

4.1 Introduction

Ongoing technology scaling significantly increases the delay defects in IC's. In previous chapters, we have examined the impacts of crosstalk noise, power supply noise and ground bounce that causes delay defects. In this chapter, we discuss about the impacts of the

manufacturing process variations combinedly with the noise disturbances in the power supply and ground networks on delay. ATPG tools are commercially utilized to detect delay-related defects. As these tools ignore the physical design parameters affecting the gates, interconnects, power supply and ground networks, they are incapable of accurately generating the right pattern that can capture worst-case path delay in a circuit. Statistical static timing analysis (SSTA) [117] based techniques models gate or path delays, considers the variations in all interconnecting nets between the gates in a path. Also, SSTA and process corner based STA techniques [118] work with delay probability distributions. However, they are too complex and time consuming to work with realistic path delay distributions and identify an accurate delay pattern. This has motivated us to propose a simple and novel delay probability metric to identify a worst-case path delay pattern for capturing delay defects in the presence of process variations and supply noise.

Unpredictable process parameter fluctuations and changing operating voltage conditions cause random variations in the circuit parameters, thereby affecting the expected nominal path delay values. Fluctuations in the manufacturing process affect the gate parameters such as threshold voltage (V_{th}), oxide thickness (t_{ox}), transistor length (L_g) and width (W_g), as well as the width of interconnects by varying the interconnect resistances (R), inductances (L) and capacitances (C). Operating voltage of a circuit varies depending on the noise disturbances in their power supply (i.e., power supply noise) or ground networks (i.e., ground bounce), thereby varying the drive strength of the gates in a path. Also, path delay of a circuit varies randomly depending on the test vectors (input patterns) applied at their inputs and their arrival time difference between the applied vectors. The combined impact of all these effects makes path delay estimation very difficult. Therefore, a simple yet effective method for identifying the worst-case or the most effective path delay patterns that can capture a delay defect during testing is essential.

In this work, we propose a delay probability metric to identify a worst-case path delay pattern in the presence of reliability issues (PV) and power integrity issues (SN). This metric utilize 90nm Cadence Generic Standard Cell Library to run the process corner [119] based SPICE simulations to obtain the delay distributions for all the gates and interconnects in a path. Other approaches such as monte carlo simulations [120] will be too expensive, because of the increased number of corners and more conditions to evaluate for identifying a pattern in the presence of different combinations of PV and SN

parameters. Also, it will complicate more with bigger circuits.

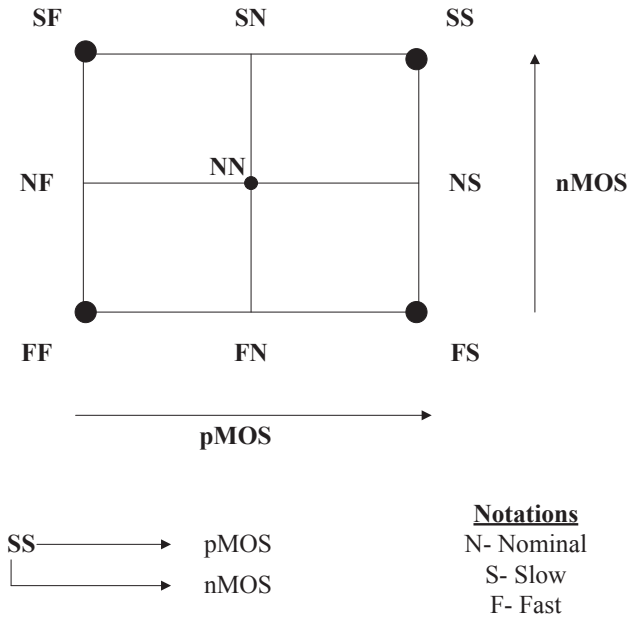


Figure 4.1: Different corner cases for process variations

Fig. 4.1 shows the different process corners employed in this work for analysing the variations in the manufacturing process. The process corners describe the behavioral differences of a chip from its normal conditions. The different corner conditions are SS, FF, NN, FS, SF, etc., where these abbreviations stand for slow nMOS slow pMOS, fast nMOS fast pMOS, nominal-nominal nMOS pMOS, fast nMOS slow pMOS, slow nMOS fast pMOS etc. These corner conditions are checked during SPICE simulation, which describes the behavior of the gates.

4.2 Prior Work

There are a number of contributions that investigate on the impact of PV's and SN. Based on the source of physical defects, they can be classified as delay defect: (1) due to a single source (i.e., either due to PV, SN or crosstalk noise), (2) from multiple sources, and (3) irrespective of the source. The first classification focuses only on process variations [121–124]. Francisco *et al.* [121] proposed a statistical timing analysis framework based on delay correlation information between two paths. Critical path delay measurement using a ring oscillator is presented in [122]. Authors in [123] have proposed an algorithm to detect a resistive interconnect defect for a path with minimum delay variance. An

optimization framework is suggested by Yu [124] based on SSTA for worst-case circuit analysis. In the second classification, different approaches for pattern generation from multiple sources were proposed [102, 125, 126]. Todri *et al.* [102] has analyzed power supply noise and ground bounce for capturing worst-case path delay patterns based on simulated annealing. Xu [125] described a statistical model for skitter with PV and power supply noise effects. Peng [126] explains their work of pattern evaluation and selection considering crosstalk and PV. In the third classification, worst-case delay of a circuit path is analyzed, with no detailed reference to the source of defects. A theoretical framework for statistical timing analysis is proposed by Orshansky and Keutzer [127] for detecting the worst-case path delay in a circuit.

In contrast to all these works, our goal is to re-examine the problem of path delay pattern generation by introducing a delay probability metric for ranking patterns under the impact of PV and SN. Using the probabilistic metrics, we can estimate and identify an efficient worst-case path delay pattern or set of patterns that can capture the worst path delay. Our method is practical and easily adaptable to be implemented on any existing pattern generation flow. Complementary to the previous works, we have additionally incorporated the impact of ground bounce in supply noise.

4.3 Contributions and Chapter Organization

Our major contributions in this chapter are summarized as follows:

- A probability metric is presented to identify worst-case path delay pattern while considering the combined impact of PV and SN. This metric aims at detecting the most-effective pattern for path delay testing from the subset of all input patterns.

- Ranking method is described based on the mean delay difference and the area of the delay probability distribution of all input patterns.

- Case study and simulation results are shown to validate our method.

The rest of the chapter is organized as follows. Section 4.4 demonstrates the motivational experiment for showing the difference in pattern generated by an ATPG tool and the worst-case path delay pattern identified from SPICE simulations. In Section 4.5, we formulate the problem of path delay distribution for detecting a worst-case pattern in the presence of PV and SN. An input pattern based ranking method is explained in section

4.6 to identify the right pattern in the presence of PV's and SN. Section 4.7 presents the simulation results on ITC'99 benchmark circuits. Finally, in Section 4.8, we summarize the findings in this chapter.

4.4 Motivational experiment

In this section, we show the main motivation behind our work of process variation-aware pattern generation. Essentially, our goal is to show that a standard ATPG tool may not be able to generate good patterns that can capture worst-case path delay in the presence of PV's. We utilize SPICE simulations to highlight the impact of PV on path delay of a circuit. The path delays for all the input patterns are measured and a single pattern is identified among them, which helps us to understand which patterns and under which conditions would provide the worst-case path delay. Such an input pattern is then compared with the ATPG tool pattern and their discrepancies are reported, hence showing that the circuit may escape the test due to the low quality of the applied test set.

4.4.1 Path Delay Analysis

An ISCAS89 benchmark circuit s27, is developed in SPICE to analyse the individual impact of PV on a circuit path. The schematic is shown in Figure. 4.2(a) with its magnified view in Figure. 4.2(b) and 4.2(c). This circuit is similar to the one shown in section 3.4, the only difference is that crosstalk noise is ignored and PV parameters are only considered. The SPICE level circuit includes the gate variation parameters such as Vthn (threshold voltage in nMOS), Vthp (threshold voltage in pMOS), oxide thickness (tox) and variations in the CMOS gate length (Lg) and gate width (Wg), as well as, interconnect variation parameters such as resistances (R1, R2, R3) and ground coupling capacitances (Cg1, Cg2, Cg3, Cg4). Victim path we examine is highlighted in Fig.4.2(a). As there are many variation parameters, along with varying signal transitions and with difference in signal arrival times, the path delay alters to a greater extend.

The nominal power supply voltage, ground reference voltage and switching frequency of 1V, 0V and 1GHz, respectively are used in this experiment. Path delay variations are observed at the target output FF3/D with the trigger input at the start of combinatorial victim path FF2/Q, refer Fig.4.2(a). All the possible vector pair signal transitions are

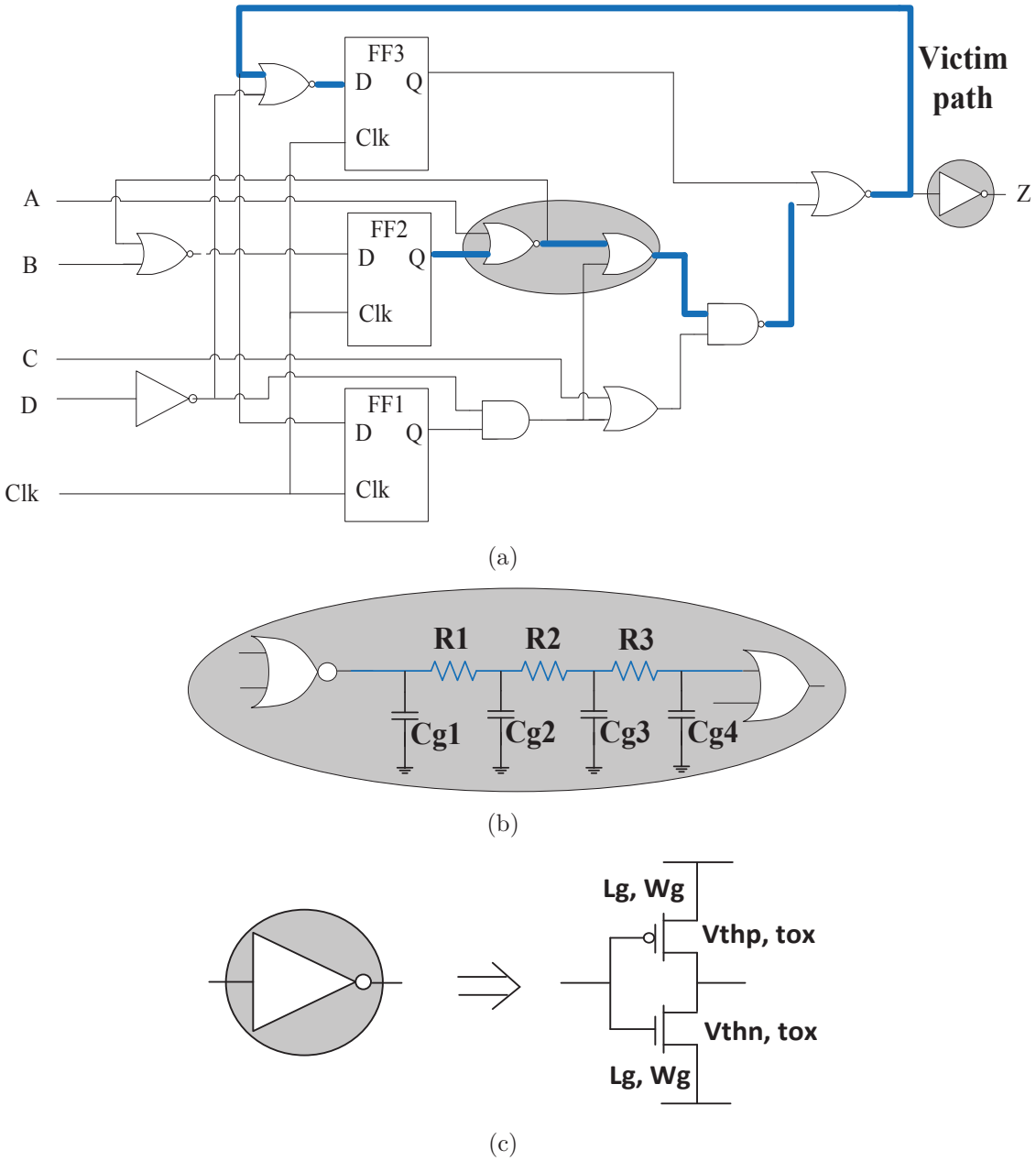


Figure 4.2: (a) s27 benchmark circuit-under-test (b) 3-pi network model for interconnects (c) CMOS model for NOT gate

applied at the circuit's primary input (A,B,C,D) and scan-input (SI). SI's can be applied after implementing the DFT scan chains. Signal transitions such as rising signal (Rise), falling signal (Fall) and stable 0 (S0) and stable 1 (S1) condition are given at their inputs. Along with them, different input arrival times of {-200ps 0ps +200ps} based on the setup and hold timing constraints (10% tolerance with the fixed arrival time) were given. The +/- indicates the advance and the lag in the arrival time.

The combinations of all possible permutations of process variation parameters, vector

Table 4.1: Path delay variation due to the impact of process variations

Inputs	Signal transitions	Arrival time	Delay variation	Delay impact
A	Rise, Fall, S0, S1	{-200ps, 0ps, 200ps}	-12.30% to +16.84%	speedup(-ve) / slowdown(+ve)
B				
C				
D				
SI				

pair signal transitions and different arrival times result in a large data set. Therefore, the minimum to maximum path delay variations obtained in comparison with nominal delay (i.e., without process variation) are only shown in column 4 of Table 4.1. For a simple circuit like s27, the maximum path delay variation of +16.84% is obtained. This indicates the importance of considering the impact of process variations during the path delay fault testing. The vector pairs (V1,V2) to the flip-flops (FF1, FF2, FF3) identified after the SPICE simulations, their worst-case path delay (δ_1) are listed in column 2-5 of Table 4.2.

4.4.2 Path Delay Fault Testing

For path delay fault testing, we reconstructed a sequential circuit module similar to the one in Figure. 3.4. Then, we tested with the existing path delay fault model in the ATPG tool, TetraMAX® at 1GHz clock frequency. This model is widely used in industry for avoiding IC's with delay defects, caused by manufacturing process variations.

The ATPG tool doesn't allow to include interconnect models and gate models during pattern generation, so it is not possible to analyze PV-induced delay faults. And our experiments will show that such situation occurs simply because any commercial ATPG tool does not take into account any physical design properties (such as parameter variations in the gates and in the interconnecting nets between them), as they work on logical level description of the circuit (i.e. VHDL or verilog description). Vector pairs were generated for path delay fault test for the victim path from FF2/Q to FF3/D. Column 6-9 of Table 4.2 lists, the ATPG tool generated vector pairs (V1, V2) for the flip-flop (FF1, FF2, FF3) inputs and the worst-case path delay (δ_2) measured in SPICE by their respective vector pairs.

4.4.3 Comparison of Vector Pairs

In this subsection, we compare the vector pairs identified from SPICE simulation with the one generated by the ATPG tool and their mismatches are highlighted. This comparison is to show that input patterns generated by the testing tools have to be effective enough to accommodate the path delay variations due to PV's.

Table 4.2: Pattern comparison and delay variation

	SPICE				ATPG				
	FF 1	FF 2	FF 3	δ_1 (ns)	FF 1	FF 2	FF 3	δ_2 (ns)	Δ_v (%)
V1	**0**	0	**0**	0.267	**1**	0	**1**	0.286	6.64
V2	**1**	1	**0**		**0**	1	**0**		

$$\Delta_v = \frac{(\delta_1 - \delta_2)}{\delta_1} \times 100 \tag{4.1}$$

In Table 4.2, we can see a mismatch in the vector pairs at FF1 and FF2 between the SPICE and ATPG patterns. The path delay variation (Δ_v) in equation 4.1, is the mean delay difference between the worst-case delays obtained from SPICE (δ_1) and ATPG (δ_2) patterns. The Δ_v of 6.64% (slowdown impact on the victim path) listed in column 10, implies that a path delay fault test is not performed with a good pattern. This path delay is quite high for a small circuit like s27 and it may be even higher and at an unacceptable percentage in bigger circuits. Thus, we provide an evidence that a circuit may escape the test to detect PV-induced delay defect due to lower quality in the applied vector pair. This indicates a serious notice in customizing the existing path delay fault testing method in ATPG. Therefore, our work emphasizes the need for refining ATPG tools and proposing novel methods to include good patterns that can be utilized for capturing PV-induced delay defects. For bigger circuits, there may exist many large set of test patterns. So, it is desirable to implement a method to include PV effects during pattern generation.

In the next section we formulate the problem of path delay estimation in the presence of PV's.

4.5 Problem formulation

In this section, we describe the problem that we address and propose our mathematical approach in identifying an input pattern that can capture the worst-case path delay under the impact of PV and SN. Initially, we estimate the dependent circuit parameters that affect the delay of a circuit path. Then, describe the proposed probabilistic metrics for ranking patterns based on their effectiveness. Finally, we describe our method of ranking patterns based on a delay probability metric.

4.5.1 Path Delay Estimation

In this subsection, we elaborate the proposed analytical method for computing the ranking order of input patterns. The problem of path delay test is a well-understood and widely investigated problem by the scientific community, in effectively identifying the test patterns for performance evaluation that would eventually lead to high-quality test patterns and that can lower delay defect escape rate. With technology scaling, increased circuit densities and faster switching circuits, identifying the highest quality patterns are getting even more challenging. Even more so when the impact of physical design issues and PV are taken into account. Due to the nature of the problem with many parameters that can cause a wide delay distribution, we exploit a probability based-approach to rank patterns based on their effectiveness in capturing the worst case delay under the impact of PV and SN.

Each input pattern triggers a given switching activity on the circuit and the victim (or paths that are critical) path under observation. As already shown in [102], critical paths can undergo drastic delay variation that can lead to slowdown and/or speedup impacts in a path. We expect that such delay variations will be even more pronounced when PV of transistors and interconnects are also included with SN.

Problem definition: We aim to identify the set of patterns that are the most effective in capturing the worst case path delay under the impact of PV and SN; based on the delay probability density function of each pattern.

Path delay on a circuit is computed by considering the delays of both interconnects and gates. Supply noise which exhibits itself as power (PSN) and ground (GB) voltage fluctuations can impact the operating regions of transistors, hence the delay behavior

of the gates. Additionally, different gates on a path can suffer from different amounts of supply noise. Random process variations induce deviation on the transistor's and interconnect's dimensions and carrier mobility. In this work, we consider process variation on threshold voltage, V_{th}, oxide thickness, t_{ox}, transistor gate length, L_g and width, W_g and interconnect length and width that impact interconnect parasitics, R, L, C. From supply noise perspective, we consider noise on power, V_{dd} and ground, G_{nd}.

The entire process flow of path delay estimation is shown in Fig.4.3. For an input pattern, the process variation and supply noise variation parameters are varied based on their tolerance shown in flow. From here, we obtain different path delays of the input patterns based on their varied parameters. These are further elaborated in the form of equations following.

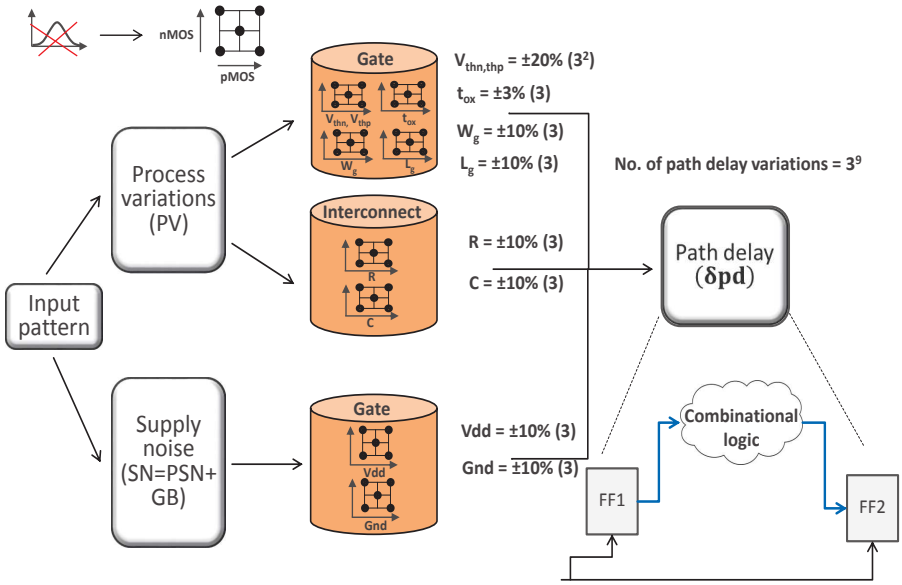


Figure 4.3: Path delay estimation

The delay of a pMOS transistor due to a rising input can be expressed as [128]:

$$\delta_r = \frac{2C_L}{\beta_p[(V_{dd} - G_{nd}) - |V_{thp}|]}[\frac{|V_{thp}| - 0.1(V_{dd} - G_{nd})}{[(V_{dd} - G_{nd}) - |V_{thp}|]} + \frac{1}{2}\log|\frac{19(V_{dd} - G_{nd}) - 20|V_{thp}|}{(V_{dd} - G_{nd})}|] \tag{4.2}$$

where C_L is the load capacitance including the next stage load and interconnect capacitance and V_{thp} is the threshold voltage of pMOS transistor. Similarly, the transistor delay for a falling input can be expressed as [128]:

$$\delta_f = \frac{2C_L}{\beta_n[(V_{dd} - G_{nd}) - V_{thn}]}[\frac{V_{thn} - 0.1(V_{dd} - G_{nd})}{[(V_{dd} - G_{nd}) - V_{thn}]} + \frac{1}{2}\log|\frac{19(V_{dd} - G_{nd}) - 20V_{thn}}{(V_{dd} - G_{nd})}|] \tag{4.3}$$

where V_{thn} is the threshold voltage of nMOS transistor and β is the transistor gain factor (in pMOS and nMOS), can be expressed as:

$$\beta = \frac{\mu\epsilon}{t_{ox}}\left(\frac{W_g}{L_g}\right) \tag{4.4}$$

where μ is the effective surface mobility of the carriers in the channel, and ϵ is the permittivity of the gate insulator. Based on the transistor delay, the gate delay can be computed, such as for an inverter the average gate delay can be computed as:

$$\delta_g = (\delta_f + \delta_r)/2 \tag{4.5}$$

Interconnects are usually modeled as π-networks with RLC parasitics, their delay, δ_{int} can be computed by applying Elmore delay formulation as a function of ζ_i at node i, as in [129]:

$$\delta_{int} = 1.047e^{\frac{-\zeta_i}{0.85}} + 1.39\zeta_i \tag{4.6}$$

where ζ_i is expressed as:

$$\zeta_i = \frac{1}{2}\left(\frac{\sum_k C_k R_{ik}}{\sqrt{\sum_k C_k L_{ik}}}\right) \tag{4.7}$$

where R_k is the interconnect resistance, C_k is the interconnect capacitance, L_k is the interconnect inductance and k represents the number of elements on the π-network interconnect model. Hence, the delay on a path can be computed as the sum of gate delays and interconnects delays (i.e. for n gates and $n-1$ interconnects on a given path) that are triggered by a given input pattern as:

$$\delta_{path} = \sum_{i=1}^{n} \delta_{g_i} + \sum_{i=1}^{n-1} \delta_{int_i} \tag{4.8}$$

4.5.2 Delay Probability Distribution

For a given path, the delay would be a function of many variables due to PV and SN. Path delay variations due to these variables can be expressed as a function of parameters as:

$$\begin{aligned} \delta_{path} &= f(V_{dd}, G_{nd}, V_{th}, t_{ox}, L_g, W_g, R, L, C, C_L) \\ &= f(SN, PV) \end{aligned} \tag{4.9}$$

Definition: In general terms, the path delay variation, δ_{path} for a given input pattern, PI can be represented as a normal distribution function. As path delay (due to δ_{g_i} or δ_{int_i}) can be real-valued random values whose distribution are unknown, the highest probability of worst-case path delay can be observed better using a normal delay distribution function. The path delay due to the input patterns for all parameters (PV and SN) can be expressed as normal distribution $N(\mu_{PI}, \sigma_{PI})$.

The mean and standard deviation of a path delay for a given input pattern, PI and all parameters can be expressed as:

$$\mu_{PI} = \frac{1}{N} \sum_{i=1}^{N} \delta_{path_i} \tag{4.10}$$

$$\sigma_{PI} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\delta_{path_i} - \mu_{PI})^2} \tag{4.11}$$

where N represents the total number of path delay measurements for a given path under PV and SN parameters.

4.5.3 Probabilistic Pattern Ranking Method

Here, we describe the concept of deriving pattern ranking method utilizing the path delay distribution function. Fig. 4.4 illustrates the probability density distribution of an input pattern under process variation and supply noise. Assuming that for a known design,

there is a predefined delay threshold with μ_{nom} that represents the tolerable delay of the circuit.

Definition: We define the *probability of identification*, $P_{identification}$ that can analytically estimate the likeliness of a pattern j under PV and SN conditions to cause a path delay at each node i, δ_{path_i} larger than the allowed delay threshold, μ_{nom}, and can be expressed as:

$$P_{identification_j}[\mu_{PI_j} \geq \mu_{nom}] = \int_{\mu_{nom}}^{\mu_{max_j}} \delta_{path_i}(t)\partial t \tag{4.12}$$

where μ_{max_j} of a pattern j is defined as:

$$\mu_{max_j} = \mu_{PI_j} + \frac{3\sigma_{PI_j}}{2} \tag{4.13}$$

Hence, for each pattern, the $P_{identification_j}$ allows us to compute the exposed area of the probability density function beyond a delay threshold also as shown in Fig.4.4.

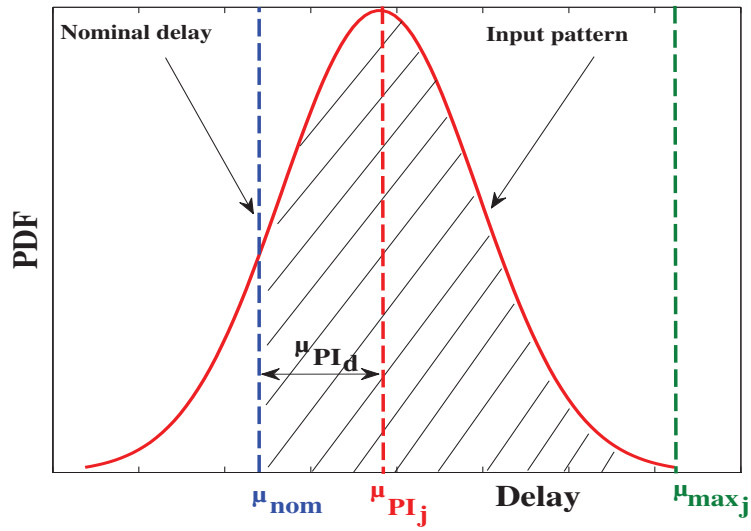


Figure 4.4: Delay Probability distribution of an input pattern

Utilizing this metric, we further define the pattern ranking method that considers both the mean, μ_{PI_j} and probability of identification, $P_{identification_j}$ of each pattern for classifying the patterns for inducing the worst path delay under PV and SN conditions. The ranking metric, $Rank_{PI_j}$ is defined as:

$$Rank_{PI_j} = \alpha_1 \mu_{PI_d} + \alpha_2 P_{identification_j} \tag{4.14}$$

where α_1 and α_2 are weight coefficients between 0 to 1 that can be given for taking into account both the changes in mean, μ_{PI_d} and the identification metric $P_{identification_j}$, where μ_{PI_d} is expressed as:

$$\mu_{PI_d} = \mu_{nom} - \mu_{PI_j} \tag{4.15}$$

The values for α_1 and α_2 can be chosen based on their priority during path delay testing i.e., either μ_{PI_d} or $P_{identification_j}$. We further utilize these probability metrics for ranking the patterns on a sample circuit to illustrate the effectiveness of the proposed ranking method.

4.6 Input Pattern Ranking Method

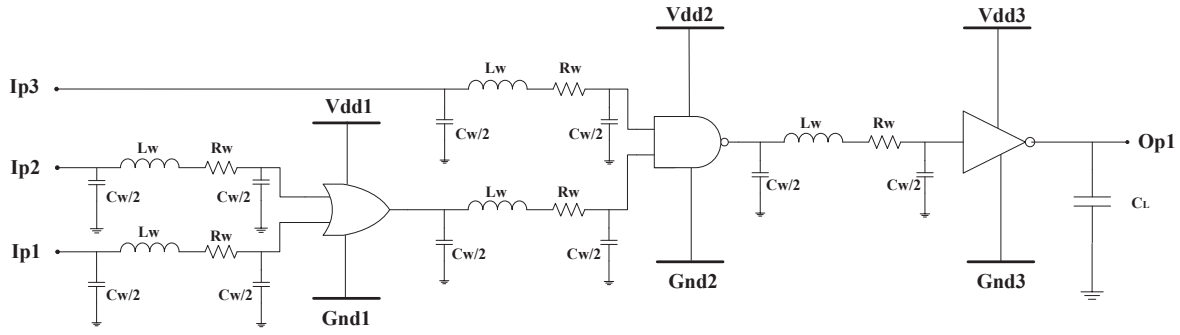


Figure 4.5: SPICE circuit under the impact of process variations and supply noise

In this section, we illustrate our proposed delay probability metric by applying it on a sample circuit as shown in Fig. 4.5. For simplicity, we have considered a small circuit as a case study, but our metric can be applied to any large circuit. The sample circuit comprises interconnect models and gates connected to a global power supply voltage and ground networks. To study the impacts of PV and SN on path delay, we incorporate parameter variations in gates (at transistor level) and interconnects (on their widths) and then control the power supply and ground voltage locally (at the gate level). The transistor and interconnect models are derived from the 90nm Predictive Technology Model (PTM) [105]. SPICE simulations are performed on the circuit for three different cases to analyze: (1) the impact of PV only, (2) the impact of SN only, and (3) the combined impact of PV and SN. For each case the following three steps are performed:

(i) estimate path delay (δ_{path_i}), (ii) compute mean (μ_{PI_j}) and standard deviation (σ_{PI_j}) from the delay probability distribution of each input pattern (PI_j), and (iii) identify the worst-case path delay pattern ($P_{identification_j}$) based on the ranking method. We utilize MATLAB to execute the mathematical computations of equations described in Section II.

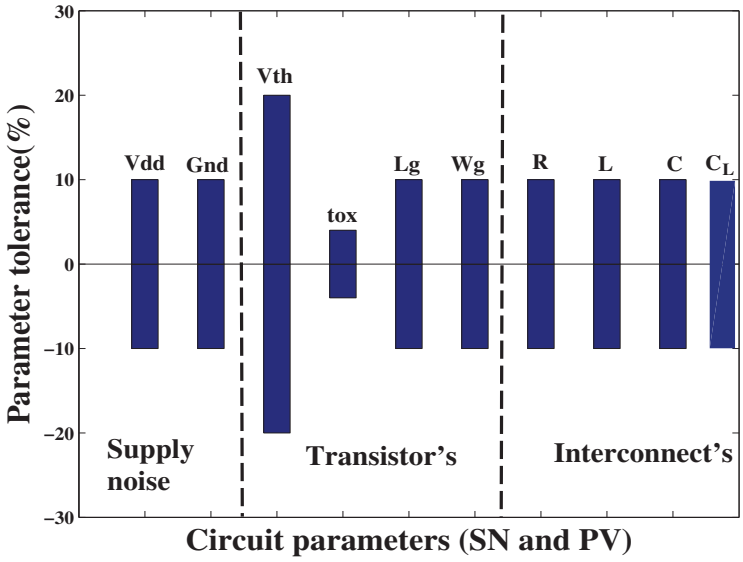


Figure 4.6: Tolerance range of circuit parameters

Input vectors (V_1V_2) are applied at each of the inputs {Ip1 Ip2 Ip3} and their respective path delays are measured at {Op1}. Local supply voltage and input operating frequency utilized in this experiment are 1V and 1GHz, respectively. We vary all the local supply voltages and the circuit parameters with their tolerance as shown in Fig. 4.6 [106]. Interconnects are modeled using RLC π-networks. Interconnect parameters (R, L, C), transistor parameters (V_{th}, t_{ox}, L_g, W_g) and load capacitance (C_L) are varied to model process variations. Local power supply voltages {Vdd1 Vdd2 Vdd3} and ground voltages {Gnd1 Gnd2 Gnd3} are adapted to model supply noise at their gate level. Path delay of the circuit can be measured between any two points; for our case study we observe between {Op1} and {Ip1}.

We perform SPICE (or HSPICE) simulations and measure the path delay for all the process corners in the circuit. Input pattern numbers, corresponding input vectors and their input transitions (i.e., rising and falling input signals) are shown in column I, column II and column III respectively of Table. 4.3, Table. 4.4 and Table. 4.5. Our delay probability metric can give all the possible path delays, but we are focused only on finding

a worst-case path delay. Their corresponding metrics will indicate the input pattern to be the most effective for capturing path delay defects under PV and SN conditions. Three different cases are explained below to show the individual and combined impact of PV and SN.

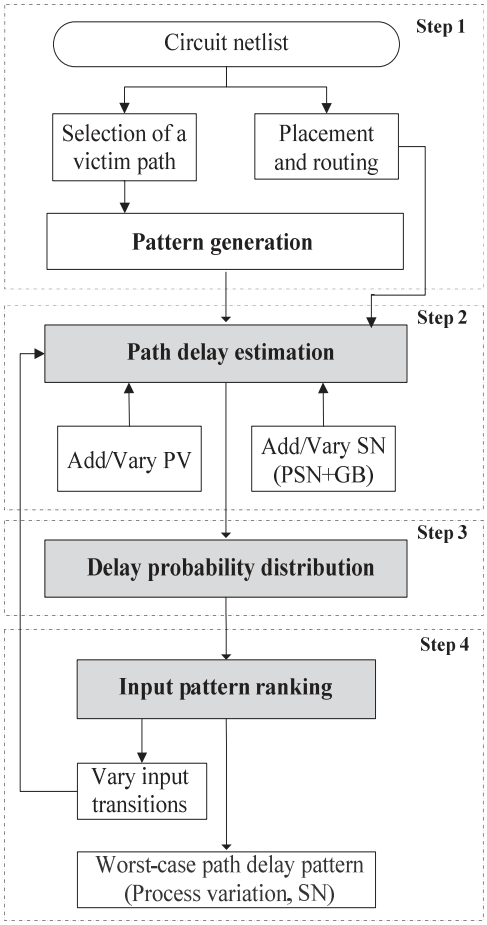


Figure 4.7: Flow of input pattern ranking method

The standard flow of input pattern ranking method explained in this section is shown in Figure 4.7. The entire flow are described in four different steps: (1) Pattern generation, (2) Path delay estimation, (3) Delay probability distribution, and (4) Input pattern ranking. Steps 2-4 are our major contributions. Step 1 is similar to the ones shown in Chapters 2 and 3. In step 2, we add process variation and supply noise variation parameters to estimate the path delay of a selected path. Then, we distribute all the delay values for obtaining a normalized curve. Finally, we apply our input pattern ranking method based on their maximum mean delay difference and the probability of likeness of pattern that causes worst-case path delay on a circuit path.

4.6.1 Impact of Process Variations

Case I: In the first case, we study only the impact of PV, by varying the interconnect and transistor parameters while applying a nominal global supply voltage at their gates. Fig. 4.8 depicts the probability density distribution function of all the input patterns under PV.

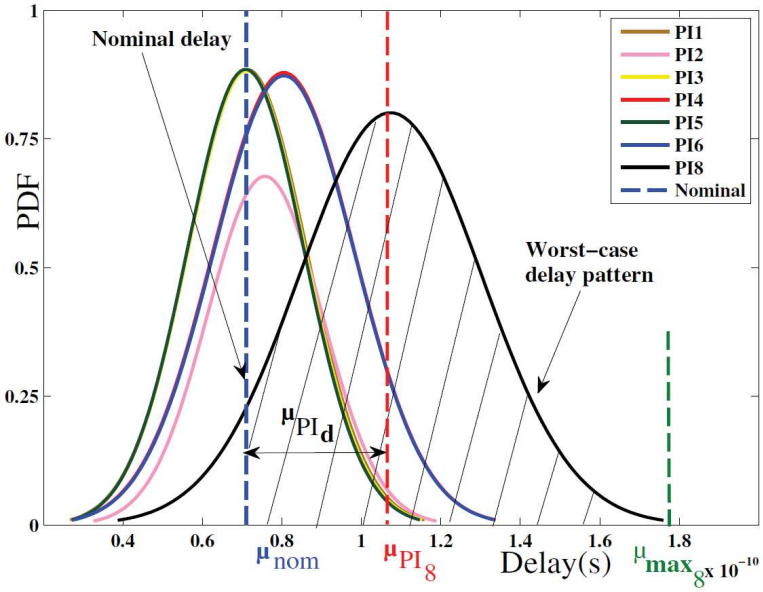


Figure 4.8: Identification of worst-case path delay pattern under PV

For each input pattern, their respective μ_{PId}, $P_{identification}$ and rank are listed in column IV, V and VI of Table. 4.3. Using our probabilistic pattern ranking method, we obtain PI_8 as the worst-case path delay pattern under the impact of PV. This is also shown in Fig. 4.8 as the pattern with the largest area exposed beyond the nominal delay threshold line.

4.6.2 Impact of Supply Noise

Case II: In this case, we study only the impact of SN, by locally varying power supply and ground voltage, while considering no process variation on transistors and interconnects. Fig. 4.9 depicts the probability density distribution function of all the input patterns under SN.

For each input pattern, their respective μ_{PId}, $P_{identification}$ and rank are listed in column IV, V and VI of Table. 4.4. Using our probabilistic pattern ranking method, we obtain PI_8 as the worst-case path delay pattern under the impact of SN. After comparing Fig.

Table 4.3: Ranking method patterns under the impact of PV

Pattern (PI_j)	Input vectors (V_1V_2) at {Ip1 Ip2 Ip3}	Input transition	Under PV		
			μ^a_{PId}	P^b_{idn}	Rank
PI1	{10 10 10}	{Fall Fall Fall}	0.27ps	0.51	5
PI2	{10 10 01}	{Fall Fall Rise}	4.67ps	0.60	4
PI3	{10 01 10}	{Fall Rise Fall}	0.07ps	0.50	6
PI4	{10 01 01}	{Fall Rise Rise}	9.47ps	0.70	3
PI5	{01 10 10}	{Rise Fall Fall}	0.13ps	0.49	7
PI6	{01 10 01}	{Rise Fall Rise}	9.47ps	0.70	2
PI7	{01 01 10}	{Rise Rise Fall}	NA[c]	NA	NA
PI8	{01 01 01}	{Rise Rise Rise}	35.9ps	0.94	1

[a]Difference between nominal delay (μ_{nom}) and delay mean of an input pattern $\mu(Pi)$, [b]$P_{identification}$ i.e., exposed area of the probability density function, [c]No output transition at launch cycle, no no delay measured.

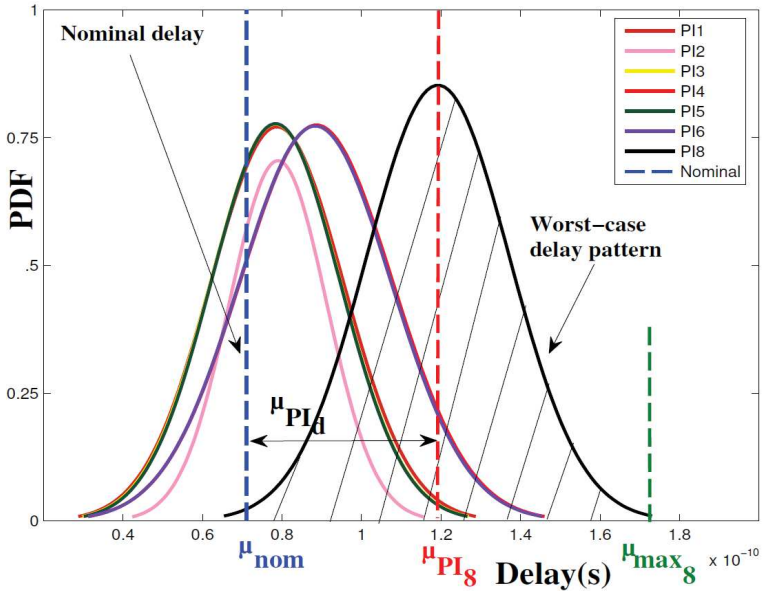


Figure 4.9: Identification of worst-case path delay pattern under SN

4.8 and Fig. 4.9, the changes in the delay distribution for the same input pattern can be noticed; indicating the higher impact of SN than PV.

Table 4.4: Ranking method patterns under the impact of SN

Pattern (PI_j)	Input vectors (V_1V_2) at {Ip1 Ip2 Ip3}	Input transition	Under SN		
			μ^a_{PId}	P^b_{idn}	Rank
PI1	{10 10 10}	{Fall Fall Fall}	7.73ps	0.67	5
PI2	{10 10 01}	{Fall Fall Rise}	7.97ps	0.58	7
PI3	{10 01 10}	{Fall Rise Fall}	7.33ps	0.66	4
PI4	{10 01 01}	{Fall Rise Rise}	17.6ps	0.81	2
PI5	{01 10 10}	{Rise Fall Fall}	7.36ps	0.67	6
PI6	{01 10 01}	{Rise Fall Rise}	17.3ps	0.80	3
PI7	{01 01 10}	{Rise Rise Fall}	NA[c]	NA	NA
PI8	{01 01 01}	{Rise Rise Rise}	48.2ps	0.98	1

[a]Difference between nominal delay (μ_{nom}) and delay mean of an input pattern $\mu(Pi)$, [b]$P_{identification}$ i.e., exposed area of the probability density function, [c]No output transition at launch cycle, no no delay measured.

4.6.3 Impact of Process Variations and Supply Noise

Case III: In the third case, we investigate the combined impact of PV and SN. Fig. 4.10 depicts the probability density distribution function of all the input patterns under PV and SN.

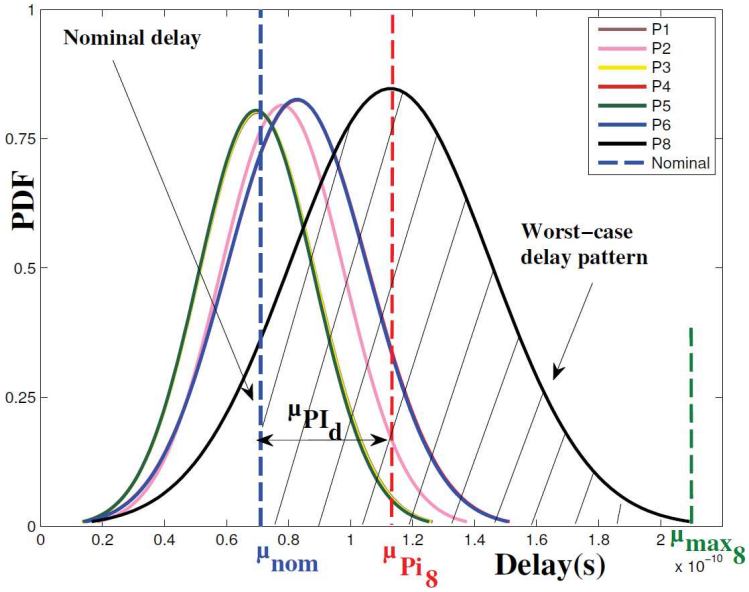


Figure 4.10: Identification of worst-case delay pattern under PV and SN

For each input pattern, their respective μ_{PId}, $P_{identification}$ and rank are listed in column IV, V and VI of Table. 4.5. Based on our probabilistic pattern ranking method, we obtain PI_8 as the worst-case path delay pattern under the combined impact of PV and SN. Please note that, while PI_8 pattern was also identified in case I and II, the value of the probability density function for path delay varies.

Table 4.5: Ranking method patterns under the impact of PV and SN

Pattern (PI_j)	Input vectors (V_1V_2) at {Ip1 Ip2 Ip3}	Input transition	Under PV and SN		
			μ^a_{PId}	P^b_{idn}	Rank
PI1	{10 10 10}	{Fall Fall Fall}	1.12ps	0.68	4
PI2	{10 10 01}	{Fall Fall Rise}	1.56ps	0.63	7
PI3	{10 01 10}	{Fall Rise Fall}	1.22ps	0.67	5
PI4	{10 01 01}	{Fall Rise Rise}	11.8ps	0.89	2
PI5	{01 10 10}	{Rise Fall Fall}	1.5ps	0.69	6
PI6	{01 10 01}	{Rise Fall Rise}	11.7ps	0.89	3
PI7	{01 01 10}	{Rise Rise Fall}	NA[c]	NA	NA
PI8	{01 01 01}	{Rise Rise Rise}	25.9ps	0.99	1

[a]Difference between nominal delay (μ_{nom}) and delay mean of an input pattern $\mu(Pi)$, [b]$P_{identification}$ i.e., exposed area of the probability density function, [c]No output transition at launch cycle, no no delay measured.

The results of three different case studies indicate that by applying the proposed ranking method, we can identify the pattern(s) that lead to the worst-case path delay when PV and SN conditions are present.

4.7 Experimental Results

In this section, we present the results based on eight full-scanned versions of ITC'99 benchmark circuits [1], their functionalities are briefly described in Table 3.3. We apply our probabilistic based ranking method on a single victim path to identify the pattern that cause worst-case path delay, even though this method can be applied to any path. Table. 4.6 and Table. 4.7 shows the summary of our experimental results. We utilized an ATPG tool for generating X-bit input patterns, as mentioned in the 1st row of each

benchmark circuit of Table 4.6. Then, we filled only the relevant X-bits (indicated by small letter 'x') based on X-filling method [115].

Table 4.6: Input pattern comparison results of ITC'99 Benchmark circuits

Ckt	Input	SI	PI
b01	X-bit pattern	001x0	11000X
	Random pattern	01110	110001
	PV pattern	00110	11000X
b02	X-bit pattern	x000	0000X
	Random pattern	1000	00001
	PV pattern	0000	0000X
b03	X-bit pattern	X0XX X1XX XXXx XxXx x0XX xXXX xxxx XX	001X XxX0
	Random pattern	0010 1110 0010 0101 1001 0100 0111 10	0011 1100
	PV pattern	X0XX X1XX XXX0 X0X1 00XX 1XXX 1101 XX	001X X0X0
b06	X-bit pattern	xx01 10XX	100x 1x
	Random pattern	1001 1000	1001 10
	PV pattern	0101 10XX	1000 10
b08	X-bit pattern	xXx1 XxxX xxxx XxxX XxX1 1	001X xxX0 XXx0 X
	Random pattern	0010 0010 0101 1010 1100 0	0011 1100 0100 1
	PV pattern	0X11 X10X 1000 X11X X0X1 1	001X 01X0 XX10 X
b09	X-bit pattern	1X00 110X 111X XxxX X101 1001 1111	0X0x X
	Random pattern	1100 1100 1111 0010 1101 1001 1111	0000 1
	PV pattern	1X00 110X 111X X10X X101 1001 1111	0X01 X
b10	X-bit pattern	XXX0 xxx1 0011 Xxxx x	XXXx 0XX1 0xxX X0X
	Random pattern	0100 0101 0011 1101 0	1110 0011 0011 000
	PV pattern	XXX0 0011 0011 X101 1	XXX1 0XX1 010X X0X
b13	X-bit pattern	0110 0111 xXXX 1XXX XX1X X01X X0XX XXXX XxXX XxxX XXXX XXXX X	0X0X xXXx XXxX 0X
	Random pattern	0110 0111 0100 1101 1010 1000 1111 0101 0110 0100 1011 1100 1	0101 1001 0110 00
	PV pattern	0110 0111 1XXX 1XXX XX1X X01X X0XX XXXX X1XX X00X XXXX XXXX X	0X0X 0XX1 XX0X 0X

We explain in detail the results of our delay probability metric for b06 benchmark circuit. For all the 16 set of X-filled input patterns (SI, PI) such as (000110XX, 100010), (000110XX, 100011), (000110XX, 100110), (000110XX, 100111), (010110XX, 100010), (010110XX, 100011), (010110XX, 100110), (010110XX, 100111), (100110XX, 100010), (100110XX, 100011), (100110XX, 100110), (100110XX, 100111), (110110XX, 100010), (110110XX, 100011), (110110XX, 100110), (110110XX, 100111) we computed the mean

delay difference μ_{PId} (i.e., 121.4ps, 123ps, 136.2ps, 188.2ps, 147.5ps, 171.9ps, 182.67ps, 127.4ps, 152.3ps, 139.2ps, 166.2ps, 122.5ps, 111.4ps, 157.8ps, 143.6ps, 132.5ps) and also the $P_{identification}$ (i.e., 0.82, 0.94, 0.81, 0.90, 0.88, 0.85, 0.80, 0.89, 0.88, 0.91, 0.84, 0.96, 0.84, 0.81, 0.94, 0.93), and then ranked (i.e., 15, 13, 10, 1, 7, 3, 2, 12, 6, 9, 4, 14, 16, 5, 8 and 11) for each input pattern, respectively. Then, we selected the input pattern with rank 1 i.e.,(000110XX, 100111), as by our method this pattern has the highest probability to give the worst-case path delay under the impact of PV and SN. Also, we selected the input pattern generated by the ATPG tool i.e., (10011000, 100110), whose rank is 13 as per our method. In column 8, the mean delay difference (i.e., 34.6%) between the two patterns (our method pattern and ATPG pattern) is mentioned. Such discrepancies further indicate the need to investigate the worst-case path delay problem and reveal the effectiveness of our method in ranking and selecting input patterns that take into account process variation and supply noise issues.

Table 4.7: Results of ITC'99 Benchmark circuits

Ckt	ATPG			Our method				
	μ^a_{PId}	P^b_{idn}	Rank	μ^a_{PId}	P^b_{idn}	Rank	% μ^c_{PId}	tr(s)^d
b01	136.00ps	0.91	2	140.77ps	0.95	1	3.3%	2K
b02	159.34ps	0.96	2	167.00ps	0.93	1	4.5%	3K
b03	148.60ps	0.83	187	179.34ps	0.86	1	17.1%	31K
b06	123.00ps	0.94	13	188.20ps	0.90	1	34.6%	7K
b08	310.90ps	0.86	289	342.60ps	0.94	1	9.2%	41K
b09	269.55ps	0.92	5	285.51ps	0.90	1	5.5%	33K
b10	373.21ps	0.94	462	499.86ps	0.98	1	25.3%	19K
b13	398.74ps	0.85	39	487.33ps	0.93	1	18.1%	72K

[a]Mean delay difference (between nominal delay and identified input pattern delay), [b]Exposed area under the curve, [c]Delay difference between two methods, [d]Runtime for testing all the patterns and finding a worst-case path delay pattern (PV + SN).

The pattern generated by random X-filling using the ATPG tool differs from the pattern generated by our probabilistic method. This indicates that, while a test pattern

sensitizes a path for path delay testing, it doesn't necessarily capture its worst-case path delay. Whereas, proposed method, investigates a set of patterns and aims to rank them based on the likeliness to obtain the worst path delay when process variation and supply noise variations are taken into account. The proposed method is practical to be embedded on the standard ATPG generation flow i.e., post-ATPG X-filling, which is also the focus of our future work.

4.8 Summary

In this chapter, we proposed a delay probability metric for identifying a worst-case path delay pattern under the impact of process variation and supply noise. The presented probabilistic pattern ranking method aims at capturing delay defects during path delay test. Our experimental results on ITC'99 benchmark circuits suggests to improve the existing pattern generation methods by incorporating the impacts of PV and SN. As future research, we aim to implement the probabilistic method in X-filling pattern generation flow.

Chapter 5

Thesis Summary and Future Works

Contents

5.1 Thesis Summary

With semiconductor technology scaling, the defect spectrum now includes more problems such as crosstalk noise, resistive shorts, resistive opens in interconnects, power supply noise and ground bounce in the supply networks, as well as, process variations in interconnects or at gate level. These are not always detected by the traditional fault models. Delay-related parametric failures in semiconductor devices increase the defect escape rate, yield loss and diminish the reliability rate. Therefore, different test techniques have been widely adopted in the industry to detect defects using patterns generated by ATPG tools (Automatic Test Pattern Generation). For instance, at-speed delay test uses the path delay fault model that targets delay defects during IC testing phase. They are commercially employed due to their minimal implementation cost and higher test coverage.

Signal and power integrity issues are mostly impacted by crosstalk noise, power supply noise and ground bounce, respectively. Also, due to the variations in the manufacturing process may cause reliability issues. All these issues negatively impact the timing characteristics in a circuit as they give rise to delay defects. In addition, the impact of these issues depends on the input test vectors provided during the scan-based test. In this thesis work, we propose methods to deal with the delay-related failures using at-speed scan test techniques for path delay test.

In Chapter 2, we present a novel physical design aware pattern generation (PDAPG) method for path delay fault testing. PDAPG method focuses on identifying the input patterns that can capture worst-case path delay on critical paths. The path delay is measured in the combined presence of physical design issues such as multi-aggressor crosstalk, power supply noise and ground bounce. As technology shrinks, the spacing between adjacent interconnect keeps decreasing, which increases the overall contribution of the coupling capacitances to the total interconnect capacitance. Also, gate noise sensitivity increases due to supply voltage scaling and limited scaling of the voltage threshold. As a result, crosstalk noise and supply noise play a greater role in sub-100nm technologies and creates signal integrity issues. Therefore, it is vital to consider both effects during design validation and the path delay test to ensure the performance and reliability of the chip.

PDAPG method is implemented on ITC'99 benchmark circuits and the results were shown by pattern comparison and path delay measurement. The basic principle behind this method is: all the aggressor nets closer to the critical path are backtraced till the relevant control input bits are found and then they are filled accordingly with different transition (rise, fall, stable 0, stable 1) signals. Experimental results demonstrate that our method is better in choosing the effective patterns, that can increase the impact of crosstalk noise, from the subset of all possible patterns to identify a worst-case path delay pattern. Although, we used a single robust critical path as our initial critical path repository, our method can be applied to any number of critical paths for efficiently identifying the high-quality input test patterns. The results suggest that the existing ATPG test methods have to be refined to incorporate the impact of physical design issues.

In Chapter 3, we propose an ATPG method that targets path delay fault by considering the impact of crosstalk noise. This work is the continuation of chapter 2, which is limited due to higher computational time, as the pattern identification method is based on SPICE simulations. Sometimes, SPICE-based simulation can be exhaustive for bigger circuits. Hence, the new method eliminates the exhaustive SPICE simulation and it is equally good for any sized circuits. The basic principle behind this method is: all the aggressor nets closer to the critical path are constrained depending on the victim net signal transition. Our method is implemented on ITC'99 benchmark circuits and the results were compared with SPICE simulation. With this method, we also show the effectiveness

in computational time and pattern quality.

Experimental results demonstrate that the proposed method is fast even after considering all the aggressor nets neighboring the functionally testable critical paths. This method reduces the very time-consuming validation phase. The generated patterns can potentially be used for speed binning to replace the functional patterns if applied to a proper set of critical paths. As future work, we plan to extend this flow to add the impacts of other signal integrity issues such as supply noise during pattern generation.

In Chapter 4, we propose a delay probability metric for identifying a worst-case path delay pattern in the combined presence of process variation and supply noise. This metric can detect the most effective pattern from the set of all input patterns and can be employed with the existing ATPG path delay fault test.

The basic principle behind this method is: All the input patterns are ranked based on their mean delay difference and the area of the delay probability distribution. Among them, the pattern giving worst-case path delay is identified. This pattern can be utilized to capture the delay defects during IC test phase. Our experimental results on ITC'99 benchmark circuits suggests that the existing pattern generation methods need to be improved by incorporating the impacts of process variations and supply noise.

5.2 Future works

This thesis work gives a number of exciting research directions. The application of our work can bring tremendous improvements to the IC test quality by ensuring better defect coverage and for an increased manufacturing yield during speed binning of IC chips. Here, we summarize some extensions to this work that are relevant for DFT and defect diagnosis.

Our work has covered the analysis of path delay variations induced by crosstalk noise, power supply noise, ground bounce and process variations, which are the major industrial concerns. We have developed constrained ATPG method by improving the existing classical ATPG method. Crosstalk noise issues are only considered in this method, due to the complexity in developing and modeling power supply noise, ground bounce and process variations effects to ATPG tool. Our method can be further developed to add the impacts of power supply noise, ground bounce and process variations.

Experimental results on the benchmark circuits have demonstrated the efficiency of the proposed techniques. Furthermore, after integrating to the ATPG tool, our constrained ATPG method could save significant CPU runtime. It is capable of identifying worst-case path delay pattern for the biggest ITC'99 benchmark circuit, b22 with 21772 gates. Anyways, we think this method should be further improved to be able to deal with circuits composed of billions of gates. This method still needs to be refined in terms of computational time (for generating patterns) and efficiency to deal with the real industrial circuits. In other words, the methods in this thesis still opens up a door to new and refined ATPG techniques.

Our procedures are still open to add more effects if necessary such as on-chip temperature variations, substrate coupling, etc. It is easy to add more effects to our proposed flow of ATPG:

(1) Model the new effect and map it to the parameter variant (i.e., gate or interconnect), where it impacts. For example, for crosstalk noise, we have mapped their variation to the interconnects.

(2) Combine all the new effects and sensitize the critical path for generating a worst-case path delay pattern.

Appendix A

Circuit netlist creation using Cadence Encounter RC compiler

This RTL code converts a circuit behaviour description (in vhdl) to a circuit design implementation (to verilog). The code mentioned is for b01 circuit of ITC'99 benchmark circuit.

```
set_attribute library {CORE90GPSVT_nom_1.00V_25C.lib}
read_hdl -vhdl b01.vhd
elaborate b01
synthesize -to_mapped
write -mapped > b01_scan.v
ungroup -flatten -all
write -mapped > b01_scan.v
define_dft shift_enable -active high -create_port TEST_SE
define_dft test_clock clock
check_dft_rules
define_dft scan_chain -sdi TEST_SI -sdo TEST_SO -create_ports
synthesize -to_mapped
connect_scan_chains -preview
connect_scan_chains
write_atpg -stil > b01_90nm.spf
write -mapped > b01flatscan.v
write_sdc > b01.sdc
report_timing >timing_report.txt
write_sdf -version 2.1 -setuphold split > b01.sdf
exit
```

Generation of victim paths using Synopsys Primetime STA

All the possible set of victim paths (critical paths) are generated using Synopsys Primetime STA tool. The code mentioned is for b01 circuit of ITC'99 benchmark circuit.

```
set link_path .CORE90GPSVT_nom_1.00V_25C.db
```

```
read_verilog b01flatscan.v
current_design b01
link_design
create_clock clock -period 1 -waveform {0.0 0.5}
set_case_analysis 0 [get_port TEST_SE]
source ./pt2tmax.tcl
set_input_delay -clock clock 0.5 [remove_from_collection [all_inputs] {clock}]
set_output_delay -clock clock 0.5 [all_outputs]
set_false_path -from [all_inputs] -to [all_outputs]
write_sdc b01.sdc
write_delay_paths -max_paths 1000 -slack 20 -clock clock b01_delaypaths.txt
exit
```

Placement and Routing using Cadence Encounter P&R tool

```
win
getenv ENCOUNTER_CONFIG_RELATIVE_CWD
setDoAssign
getIoFlowFlag
setUIVar rda_Input ui_gndnet gnd
setUIVar rda_Input ui_leffile cmos090gp_soc.lef mod_CORE90GPSVT.lef
setUIVar rda_Input ui_settop 0
setUIVar rda_Input ui_netlist mod_b01flatscan.v
setUIVar rda_Input ui_pwrnet vdd
commitConfig
fit
setDrawView fplan
getIoFlowFlag
setIoFlowFlag 0
floorPlan -site CORE -r 0.719266055046 0.7 6 6 6 6
uiSetTool select
getIoFlowFlag
fit
```

addRing -spacing_bottom 0.9 -width_left 1.8 -width_bottom 1.8 -width_top 1.8 -spacing_top 0.9 -layer_bottom M1 -stacked_via_top_layer M7 -width_right 1.8 -around core -jog_distance 0.42 -offset_bottom 0.6 -layer_top M1 -threshold 0.42 -offset_left 0.6 -spacing_right 0.9 -spacing_left 0.9 -offset_right 0.6 -offset_top 0.6 -layer_right M2 -nets gnd vdd -stacked_via_bottom_layer M1 -layer_left M2

addStripe -block_ring_top_layer_limit M3 -max_same_layer_jog_length 0.84 -padcore_ring_bottom_layer_limit M1 -set_to_set_distance 100 -stacked_via_top_layer M7 -padcore_ring_top_layer_limit M3 -spacing 0.42 -merge_stripes_value 0.42 -layer M2 -block_ring_bottom_layer_limit M1 -width 0.42 -nets gnd vdd -stacked_via_bottom_layer M1

setPlaceMode -fp false

placeDesign -prePlaceOpt

setDrawView place

setDrawView fplan

getNanoRouteMode -quiet

getNanoRouteMode -user -drouteEndIteration

getNanoRouteMode -user -drouteStartIteration

getNanoRouteMode -user -routeBottomRoutingLayer

getNanoRouteMode -user -routeTopRoutingLayer

getNanoRouteMode -quiet -envSuperthreading

getNanoRouteMode -quiet -drouteFixAntenna

getNanoRouteMode -quiet -routeInsertAntennaDiode

getNanoRouteMode -quiet -routeAntennaCellName

getNanoRouteMode -quiet -timingEngine

getNanoRouteMode -quiet -routeWithTimingDriven

getNanoRouteMode -quiet -routeWithEco

getNanoRouteMode -quiet -routeWithLithoDriven

getNanoRouteMode -quiet -droutePostRouteLithoRepair

getNanoRouteMode -quiet -routeWithSiDriven

getNanoRouteMode -quiet -routeTdrEffort

getNanoRouteMode -quiet -routeWithSiPostRouteFix

getNanoRouteMode -quiet -drouteAutoStop

getNanoRouteMode -quiet -routeSelectedNetOnly

getNanoRouteMode -quiet -drouteStartIteration

setNanoRouteMode -quiet -drouteStartIteration default

getNanoRouteMode -quiet -envNumberProcessor

getNanoRouteMode -quiet -envSuperthreading

getNanoRouteMode -quiet -routeTopRoutingLayer

setNanoRouteMode -quiet -routeTopRoutingLayer default

getNanoRouteMode -quiet -routeBottomRoutingLayer

setNanoRouteMode -quiet -routeBottomRoutingLayer default

getNanoRouteMode -quiet -drouteEndIteration

setNanoRouteMode -quiet -drouteEndIteration default

getNanoRouteMode -quiet -routeEcoOnlyInLayers

getNanoRouteMode -quiet -routeWithTimingDriven

setNanoRouteMode -quiet -routeWithTimingDriven false

getNanoRouteMode -quiet -routeWithSiDriven

setNanoRouteMode -quiet -routeWithSiDriven false

routeDesign -globalDetail

setDrawView place

verifyGeometry

verifyConnectivity -type all -error 1000 -warning 50

saveDesign b01.enc

saveDesign b01.enc

saveNetlist b01.v

streamOut b01.gds -mapFile streamOut.map -libName DesignLib -units 1000 -mode ALL

summaryReport -outdir summaryReport

extractRC

rcOut -setload b01.setload

rcOut -setres b01.setres

rcOut -spf b01.spf

rcOut -spef b01.spef

saveFPlan ./b01.fp

savePlace ./b01.place.gz

saveNetlist b01.v

global dbgLefDefOutVersion

set dbgLefDefOutVersion 5.5

defOut -floorplan -netlist -routing b01.def

set dbgLefDefOutVersion 5.5

streamOut b01.gds -mapFile streamOut.map -libName DesignLib -stripes 1 -units 1000 -mode ALL

summaryReport -outdir Snapshot

saveDesign ./b01.enc

createSnapshot -dir Snapshot -name b01 -overwrite

exit

Pattern generation using Synopsys TetraMAX ATPG tool

set_messages -log tmax.log -rep

read_netlist CORE90GPSVT.v -library

read_netlist b01flatscan.v

run_build_model b01

set_delay -common_launch_capture_clock

set_delay -launch_cycle system_clock

set_delay -nopi_changes -nopo_measures

add_po_masks -all

set_drc b01_90nm.spf

run_drc

set_fault -model path_delay -atpg_effectiveness -fault_coverage

add_delay_path b01_delay_pt.txt

add_faults -all

set_atpg -full_seq_atpg

run_atpg full_sequential_only

report_faults -all

report_po_masks

report_faults -class DT

write_patterns b01_pat_new.stil -internal -format STIL -unified_stil_flow -replace

write_testbench -input b01_pat_new.stil -output path_pat_serial_tb -parameter -serial -replace -log path_pat_serial_tb.log

exit

Scientific Contributions

International Conferences

[**ISVLSI15**] A. Asokan, A. Bosio, L. Dilillo, P. Girard, S. Pravossoudovitch, A.Virazel, “An ATPG Flow to Generate Crosstalk-Aware Path Delay Pattern,” accepted at IEEE Computer Society Annual Symposium on VLSI (ISVLSI) 2015.

[**ISVLSI14**] A. Asokan, A. Todri-Sanial, A. Bosio, L. Dilillo, P. Girard, S. Pravossoudovitch, A.Virazel, “A Delay Probability Metric for Input Pattern Ranking Under Process Variation and Supply Noise”, IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pp.226-231, 2014.

[**DDECS14**] A. Asokan, A. Todri-Sanial, A. Bosio, L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel, “Path Delay Test in the Presence of Multi-Aggressor Crosstalk, Power Supply Noise and Ground Bounce”, IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS), pp.207-212, 2014.

National Conferences

[**JNRDM15**] A. Asokan, A. Bosio, L. Dilillo, P. Girard, S. Pravossoudovitch, A.Virazel, "Layout-Aware ATPG for Path Delay Fault”, JNRDM 2015 : Bordeaux, France.

[**GDR14**] A. Asokan, A. Bosio, L. Dilillo, P. Girard, S. Pravossoudovitch, A.Virazel, "Crosstalk and Supply Noise-Aware Pattern Generation for Delay Testing”, GDR SOC / SIP 2014 : gdrsocsip, Paris.

International seminars

[**SETS11**] A. Asokan, A. Todri-Sanial, A. Bosio, L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel, Signal Integrity - Aware Pattern Generation for Delay Testing, South European Test Seminar 2013 (SETS13), Obergurgl, Austria.

Bibliography

[1] [Online]. Available: http://www.cad.polito.it/downloads/tools/itc99.html

[2] G. E. Moore, "Cramming More Components onto Integrated Circuits", Electronics Magazine," in *Electronics Magazine*, 1965, pp. Vol. 38, No. 8.

[3] More-than-Moore, white paper, International Technology Roadmap for Semiconductors (ITRS), 2010. [Online]. Available: http://www.itrs.net/ITRS%201999-2014%20Mtgs,%20Presentations%20&%20Links/2010ITRS/IRC-ITRS-MtM-v2%203.pdf

[4] G. Moretti. "What Is Not Testable Is Not Fixable", Systems design engineering community, September 17th, 2014. [Online]. Available: http://chipdesignmag.com/sld/blog/2014/09/17/what-is-not-testable-is-not-fixable/

[5] M. Alam, K. Roy, and C. Augustine, "Reliability- and Process-variation aware design of integrated circuits - A broader perspective," in *Reliability Physics Symposium (IRPS), 2011 IEEE International*, 2011, pp. 4A.1.1–4A.1.11.

[6] C. W. Mohammad Tehranipoor, "Introduction to hardware security and trust," in *1st Edition*. Springer, 2011.

[7] D. Frank, R. Dennard, E. Nowak, P. Solomon, Y. Taur, and H.-S. P. Wong, "Device scaling limits of Si MOSFETs and their application dependencies," in *Proceedings of the IEEE*, 2001, pp. 259–288.

[8] C. Hawkins, A. Keshavarzi, and J. Segura, "View from the bottom: nanometer technology AC parametric failures - why, where, and how to detect," in *Defect and Fault Tolerance in VLSI Systems, 2003. Proceedings. 18th IEEE International Symposium on*, 2003, pp. 267–276.

[9] H. S. P. Wong, D. J. Frank, P. M. Solomon, C. H. J. Wann, and J. J. Welser, "Emerging Nanoelectronics: Life with and after CMOS," in *A. M. Ionescu and K. Banerjee, Eds. Norwell, MA: Kluwer Academic Publishers*, 2004, pp. 259–288.

[10] TetraMAX ATPG, Automatic Test Pattern Generation, Synopsys Inc., Mountain View, CA.

[11] P. Chen and K. Keutzer, "Towards true crosstalk noise analysis," in *Computer-Aided Design, 1999. Digest of Technical Papers. 1999 IEEE/ACM International Conference on*, 1999, pp. 132–137.

[12] L. Ding, D. Blaauw, and P. Mazumder, "Accurate crosstalk noise modeling for early signal integrity analysis," in *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 2003, pp. 627–634.

[13] K. Agarwal, D. Sylvester, and D. Blaauw, "Modeling and analysis of crosstalk noise in coupled RLC interconnects," in *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 2006, pp. 892–901.

[14] N. Nagaraj, P. Balsara, and C. Cantrell, "Crosstalk noise verification in digital designs with interconnect process variations," in *VLSI Design, 2001. Fourteenth International Conference on*, 2001, pp. 365–370.

[15] M. Cuviello, S. Dey, X. Bai, and Y. Zhao, "Fault Modeling and Simulation for Crosstalk in System-on-Chip Interconnects," in *Computer-Aided Design, 1999. Digest of Technical Papers. 1999 IEEE/ACM International Conference on*, 1999, pp. 297–303.

[16] Aniket and R. Arunachalam, "A novel algorithm for testing crosstalk induced delay faults in vlsi circuits," in *VLSI Design, 2005. 18th International Conference on*, 2005, pp. 479–484.

[17] K. Ganeshpure and S. Kundu, "On atpg for multiple aggressor crosstalk faults," in *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 2010, pp. 774–787.

[18] S. Chun, T. Kim, and S. Kang, "ATPG-XP: Test Generation for Maximal Crosstalk-Induced Faults," in *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 2009, pp. 1401–1413.

[19] M. Tehranipour, N. Ahmed, and M. Nourani, "Multiple transition model and enhanced boundary scan architecture to test interconnects for signal integrity," in *Computer Design, 2003. Proceedings. 21st International Conference on*, 2003, pp. 554–559.

[20] S.-Y. Yang, C. A. Papachristou, and M. Taib-Azar, "Improving bus test via iddt and boundary scan," in *Design Automation Conference, 2001. Proceedings*, 2001, pp. 307–312.

[21] M. Tehranipour, N. Ahmed, and M. Nourani, "Testing soc interconnects for signal integrity using boundary scan," in *VLSI Test Symposium, 2003. Proceedings. 21st*, 2003, pp. 158–163.

[22] A. Sinha, S. Gupta, and M. Breuer, "Validation and test issues related to noise induced by parasitic inductances of vlsi interconnects," in *Advanced Packaging, IEEE Transactions on*, 2002, pp. 329–339.

[23] W. Chen, S. Gupta, and M. Breuer, "Test generation in vlsi circuits for crosstalk noise," in *Test Conference, 1998. Proceedings., International*, 1998, pp. 641–650.

[24] W.-Y. Chen, S. Gupta, and M. Breuer, "Test generation for crosstalk-induced delay in integrated circuits," in *Test Conference, 1999. Proceedings. International*, 1999, pp. 191–200.

[25] A. Krstic, J.-J. Liou, Y.-M. Jiang, and K.-T. Cheng, "Delay testing considering crosstalk-induced effects," in *Test Conference, 2001. Proceedings. International*, 2001, pp. 558–567.

[26] W. Chen, S. Gupta, and M. Breuer, "Test generation in VLSI circuits for crosstalk noise," in *Test Conference, 1998. Proceedings., International*, 1998, pp. 641–650.

[27] W.-Y. Chen, S. Gupta, and M. Breuer, "Test generation in VLSI circuits for crosstalk noise," in *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 2002, pp. 1117–1131.

[28] B. Kaushik, S. Sarkar, R. Agarwal, and R. Josh, "An Analytical Approach to Dynamic Crosstalk in Coupled Interconnects," in *Microelectronics Journal*, 2006, pp. 85–92.

[29] S. Bhardwaj, S. Vrudhula, and D. Blaauw, "Estimation of signal arrival times in the presence of delay noise," in *Computer Aided Design, 2002. ICCAD 2002. IEEE/ACM International Conference on*, 2002, pp. 418–422.

[30] W. Chen, S. Gupta, and M. Breuer, "Analytic models for crosstalk delay and pulse analysis under non-ideal inputs," in *Test Conference, 1997. Proceedings., International*, 1997, pp. 809–818.

[31] P. Chen and K. Keutzer, "Towards true crosstalk noise analysis," in *Computer-Aided Design, 1999. Digest of Technical Papers. 1999 IEEE/ACM International Conference on*, 1999, pp. 132–137.

[32] W.-Y. Chen, S. Gupta, and M. Breuer, "Test generation for crosstalk-induced faults: framework and computational results," in *Test Symposium, 2000. (ATS 2000). Proceedings of the Ninth Asian*, 2000, pp. 305–310.

[33] J. Lee and M. Tehranipoor, "A novel pattern generation framework for inducing maximum crosstalk effects on delay-sensitive paths," in *Test Conference, 2008. ITC 2008. IEEE International*, 2008, pp. 1–10.

[34] S. Chun, T. Kim, and S. Kang, "Atpg-xp: Test generation for maximal crosstalk-induced faults," in *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 2009, pp. 1401–1413.

[35] V. Mehta, M. Marek-Sadowska, K.-H. Tsai, and J. Rajski, "Timing defect diagnosis in presence of crosstalk for nanometer technology," in *Test Conference, 2006. ITC '06. IEEE International*, 2006, pp. 1–10.

[36] Q. Zhu, "Power distribution network design for vlsi," in *John Wiley and Sons, Feb*, 2004.

[37] S. Lin and N. Chang, "Challenges in power-ground integrity," in *Computer Aided Design, 2001. ICCAD 2001. IEEE/ACM International Conference on*, 2001, pp. 651–654.

[38] H. Chen and D. Ling, "Power supply noise analysis methodology for deep-submicron vlsi chip design," in *Design Automation Conference, 1997. Proceedings of the 34th*, 1997, pp. 638–643.

[39] S. Sapatnekar and H. Su, "Analysis and optimization of power grids," in *Design Test of Computers, IEEE*, 2003, pp. 7–15.

[40] H. Zheng, L. Pileggi, and B. Kratiter, "Electrical modeling of integrated-package power and ground distributions," in *Design Test of Computers, IEEE*, 2003, pp. 24–31.

[41] J. Saxena, K. Butler, V. Jayaram, S. Kundu, N. Arvind, P. Sreeprakash, and M. Hachinger, "A case study of ir-drop in structured at-speed testing," in *Test Conference, 2003. Proceedings. ITC 2003. International*, 2003, pp. 1098–1104.

[42] P. Girard, "Survey of low-power testing of vlsi circuits," in *Design Test of Computers, IEEE*, 2002, pp. 80–90.

[43] X. Wen, Y. Yamashita, S. Kajihara, L.-T. Wang, K. Saluja, and K. Kinoshita, "On low-capture-power test generation for scan testing," in *VLSI Test Symposium, 2005. Proceedings. 23rd IEEE*, 2005, pp. 265–270.

[44] X. Wen, S. Kajihara, K. Miyase, T. Suzuki, K. Saluja, L.-T. Wang, K. Abdel-Hafez, and K. Kinoshita, "A new atpg method for efficient capture power reduction during scan testing," in *VLSI Test Symposium, 2006. Proceedings. 24th IEEE*, 2006, pp. 6 pp.–65.

[45] S. Remersaro, X. Lin, Z. Zhang, S. Reddy, I. Pomeranz, and J. Rajski, "Preferred fill: A scalable method to reduce capture power for scan based designs," in *Test Conference, 2006. ITC '06. IEEE International*, 2006, pp. 1–10.

[46] A. Krstic, Y.-M. Jiang, and K.-T. Cheng, "Pattern generation for delay testing and dynamic timing analysis considering power-supply noise effects," in *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 2001, pp. 416–425.

[47] E. Liau and D. Schmitt-Landsiedel, "Automatic worst case pattern generation using neural networks genetic algorithm for estimation of switching noise on power supply lines in cmos circuits," in *Test Workshop, 2003. Proceedings. The Eighth IEEE European*, 2003, pp. 105–110.

[48] N. Ahmed, M. Tehranipoor, and V. Jayaram, "Supply voltage noise aware atpg for transition delay faults," in *VLSI Test Symposium, 2007. 25th IEEE*, 2007, pp. 179–186.

[49] C. Tirumurti, S. Kundu, S. Sur-Kolay, and Y.-S. Chang, "A modeling approach for addressing power supply switching noise related failures of integrated circuits," in *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, 2004, pp. 1078–1083 Vol.2.

[50] L.-C. Wang, Z. Yue, X. Lu, W. Qiu, W. Shi, and D. Walker, "A vector-based approach for power supply noise analysis in test compaction," in *Test Conference, 2005. Proceedings. ITC 2005. IEEE International*, 2005, pp. 10 pp.–526.

[51] A. Kokrady and C. Ravikumar, "Static verification of test vectors for ir drop failure," in *Computer Aided Design, 2003. ICCAD-2003. International Conference on*, 2003, pp. 760–764.

[52] I. Polian, A. Czutro, S. Kundu, and B. Becker, "Power droop testing," in *Computer Design, 2006. ICCD 2006. International Conference on*, 2006, pp. 243–250.

[53] [Online]. Available: http://www.cadence.com/products/ld/rtl_compiler/Pages/default.aspx

[54] M. Nourani and A. Radhakrishnan, "Power-supply noise in socs: Atpg, estimation and control," in *Test Conference, 2005. Proceedings. ITC 2005. IEEE International*, 2005, pp. 10 pp.–516.

[55] M. Tehranipoor and K. Butler, "Power supply noise: A survey on effects and research," in *Design Test of Computers, IEEE*, 2010, pp. 51–67.

[56] L.-R. Zheng, B.-X. Li, and H. Tenlunen, "Efficient and accurate modeling of power supply noise on distributed on-chip power networks," in *Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on*, 2000, pp. 513–516 vol.2.

[57] D. Mitra, S. Bhattacharjee, S. Sur-Kolay, B. Bhattacharya, S. Zachariah, and S. Kundu, "Test pattern generation for power supply droop faults," in *VLSI Design, 2006. Held jointly with 5th International Conference on Embedded Systems and Design., 19th International Conference on*, 2006, pp. 6 pp.–.

[58] K. Arabi, R. Saleh, and M. Xiongfei, "Power supply noise in socs: Metrics, management, and measurement," in *Design Test of Computers, IEEE*, 2007, pp. 236–244.

[59] J.-J. Liou, A. Krstic, Y.-M. Jiang, and K.-T. Cheng, "Modeling, testing, and analysis for delay defects and noise effects in deep submicron devices," in *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 2003, pp. 756–769.

[60] J. Wang, D. Walker, X. Lu, A. Majhi, B. Kruseman, G. Gronthoud, L. Villagra, P. van de Wiel, and S. Eichenberger, "Modeling power supply noise in delay testing," in *Design Test of Computers, IEEE*, 2007, pp. 226–234.

[61] J. Ma, M. Tehranipoor, O. Sinanoglu, and S. Almukhaizim, "Identification of ir-drop hot-spots in defective power distribution network using tdf atpg," in *Design and Test Workshop (IDT), 2010 5th International*, 2010, pp. 122–127.

[62] T. Tang and E. Friedman, "Estimation of transient voltage fluctuations in the cmos-based power distribution networks," in *Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on*, 2001, pp. 463–466 vol. 5.

[63] ——, "On-chip delta;i noise in the power distribution networks of high speed cmos integrated circuits," in *ASIC/SOC Conference, 2000. Proceedings. 13th Annual IEEE International*, 2000, pp. 53–57.

[64] G. Bai, S. Bobba, and I. Hjj, "Static timing analysis including power supply noise effect on propagation delay in vlsi circuits," in *Design Automation Conference, 2001. Proceedings*, 2001, pp. 295–300.

[65] R. Ahmadi and F. Najm, "Timing analysis in presence of power supply and ground voltage variations," in *Computer Aided Design, 2003. ICCAD-2003. International Conference on*, 2003, pp. 176–183.

[66] R. Saleh, S. Hussain, S. Rochel, and D. Overhauser, "Clock skew verification in the presence of ir-drop in the power distribution network," in *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 2000, pp. 635–644.

[67] Y.-M. Jiang and K.-T. Cheng, "Analysis of performance impact caused by power supply noise in deep submicron devices," in *Design Automation Conference, 1999. Proceedings. 36th*, 1999, pp. 760–765.

[68] W. Zhao, J. Ma, M. Tehranipoor, and S. Chakravarty, "Power-safe application of transition delay fault patterns considering current limit during wafer test," in *Test Symposium (ATS), 2010 19th IEEE Asian*, 2010, pp. 301–306.

[69] K. Butler, J. Saxena, A. Jain, T. Fryars, J. Lewis, and G. Hetherington, "Minimizing power consumption in scan testing: pattern generation and dft techniques," in *Test Conference, 2004. Proceedings. ITC 2004. International*, 2004, pp. 355–364.

[70] V. Devanathan, C. Ravikumar, and V. Kamakoti, "On power-profiling and pattern generation for power-safe scan tests," in *Design, Automation Test in Europe Conference Exhibition, 2007. DATE '07*, 2007, pp. 1–6.

[71] J. Lee, S. Narayan, M. Kapralos, and M. Tehranipoor, "Layout-aware, ir-drop tolerant transition fault pattern generation," in *Design, Automation and Test in Europe, 2008. DATE '08*, 2008, pp. 1172–1177.

[72] J.-J. Liou, A. Krstic, Y.-M. Jiang, and K.-T. Cheng, "Path selection and pattern generation for dynamic timing analysis considering power supply noise effects," in *Computer Aided Design, 2000. ICCAD-2000. IEEE/ACM International Conference on*, 2000, pp. 493–496.

[73] S. Zhao and K. Roy, "Estimation of switching noise on power supply lines in deep sub-micron cmos circuits," in *VLSI Design, 2000. Thirteenth International Conference on*, 2000, pp. 168–173.

[74] J. Rabaey, A. Chandrakasan, and N. B., "Digital Integrated Circuits, A Design Perspective (Second Edition)," in *Prentice Hall Publishers*, 2003.

[75] J. Roth, "Diagnosis of Automata Failures: A Calculus and a Method," in *IBM Journal of Research and Development*, 1966, pp. 278–291.

[76] P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," in *Computers, IEEE Transactions on*, 1981, pp. 215–222.

[77] H. Fujiwara, "FAN: A Fanout-Oriented Test Pattern Generation Algorithm," in *Proc. of the International Symp. on Circuits and Systems*, 1985, pp. 671–674.

[78] T. Kirkland and M. Mercer, "A Topological Search Algorithm for ATPG," in *Design Automation, 1987. 24th Conference on*, 1987, pp. 502–508.

[79] M. Schulz and E. Auth, "Advanced automatic test pattern generation and redundancy identification techniques," in *Fault-Tolerant Computing, 1988. FTCS-18, Digest of Papers., Eighteenth International Symposium on*, 1988, pp. 30–35.

[80] M. Schulz, E. Trischler, and T. Sarfert, "SOCRATES: a highly efficient automatic test pattern generation system," in *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 1988, pp. 126–137.

[81] M. Schulz and E. Auth, "Improved deterministic test pattern generation with applications to redundancy identification," in *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 1989, pp. 811–816.

[82] J. Giraldi and M. Bushnell, "EST: the new frontier in automatic test-pattern generation," in *Design Automation Conference, 1990. Proceedings., 27th ACM/IEEE*, 1990, pp. 667–672.

[83] ——, "Search State Equivalence for Redundancy Identification and Test Generation," in *Test Conference, 1991, Proceedings., International*, 1991, pp. 184–.

[84] M. Bushnell and J. Giraldi, "A Functional Decomposition Method for Redundancy Identification and Test Generation," in *Journal of Electronic Testing: Theory and Applications*, 1997, pp. 175–195.

[85] W. Kunz and D. Pradhan, "Recursive learning: a new implication technique for efficient solutions to CAD problems-test, verification, and optimization," in *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 1994, pp. 1143–1158.

[86] ——, "Recursive Learning: An attractive alternative to the decision tree for test generation in digital circuits," in *Test Conference, 1992. Proceedings., International*, 1992, pp. 816–.

[87] W. Kunz and D. Stoffel, "Reasoning in Boolean Networks: Logic Synthesis and Verification Using Testing Techniques," in *Boston: Kluwer Academic Publishers*, 1997.

[88] S. T. Chakradhar, V. D. Agrawal, and M. L. Bushnell, "Neural Models and Algorithms for Digital Testing," in *Boston: Kluwer Academic Publishers*, 1991.

[89] R. Gaede, M. Mercer, K. Butler, and D. Ross, "CATAPULT: concurrent automatic testing allowing parallelization and using limited topology," in *Design Automation Conference, 1988. Proceedings., 25th ACM/IEEE*, 1988, pp. 597–600.

[90] T. Stanion and D. Bhattacharya, "TSUNAMI: A Path Oriented Scheme for Algebraic Test Generation," in *Fault-Tolerant Computing, 1991. FTCS-21. Digest of Papers., Twenty-First International Symposium*, 1991, pp. 36–43.

[91] J. Savir and S. Patil, "On broad-side delay test," in *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 1994, pp. 368–372.

[92] J. Savir, "Skewed-load transition test: Part i, calculus," in *Test Conference, 1992. Proceedings., International*, 1992, pp. 705–.

[93] S. Wang, X. Liu, and S. Chakradhar, "Hybrid delay scan: a low hardware overhead scan-based delay test technique for high fault coverage and compact test sets," in *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, 2004, pp. 1296–1301 Vol.2.

[94] G. Bell, "Growing Challenges in Nanometer Timing Analysis," in *EE Times, Oct. 18*, 2004.

[95] M. Zhang, H. Li, and X. Li, "Path delay test generation toward activation of worst case coupling effects," in *IEEE Transactions on VLSI Systems*, 2011, pp. 1969–1982.

[96] K. Peng, M. Yilmaz, K. Chakrabarty, and M. Tehranipoor, "Crosstalk- and Process Variations-Aware High-Quality Tests for Small-Delay Defects," in *IEEE Transactions on VLSI Systems*, 2013, pp. 1129–1142.

[97] PrimeTime User Guide, Synopsys, Inc., Mountain View, CA,May 1999.

[98] A. Krstic, Y.-M. Jiang, and K.-T. Cheng, "Pattern generation for delay testing and dynamic timing analysis considering power-supply noise effects," in *IEEE Trans on Computer-Aided Design of Integrated Circuits and Systems*, 2001, pp. 416–425.

[99] J. Ma, J. Lee, and M. Tehranipoor, "Layout-Aware Pattern Generation for Maximizing Supply Noise Effects on Critical Paths," in *27th IEEE conf on VLSI Test Symposium*, 2009, pp. 221–226.

[100] J. Ma and M. Tehranipoor, "Layout-Aware Critical Path Delay Test Under Maximum Power Supply Noise Effects," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2011, pp. 1923–1934.

[101] J. Lee and M. Tehranipoor, "A Novel Pattern Generation Framework for Inducing Maximum Crosstalk Effects on Delay-Sensitive Paths," in *IEEE International Test Conference*, 2008, pp. 1–10.

[102] A. Todri, A. Bosio, L. Dilillo, P. Girard, and A. Virazel, "Uncorrelated Power Supply Noise and Ground Bounce Consideration for Test Pattern Generation," in *IEEE Transactions on VLSI Systems*, 2013, pp. 958–970.

[103] A. Sanyal, K. Ganeshpure, and S. Kundu, "Test Pattern Generation for Multiple Aggressor Crosstalk Effects Considering Gate Leakage Loading in Presence of Gate Delays," in *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 2012, pp. 424–436.

[104] M. Zhang, H. Li, and X. Li, "Path Delay Test Generation Toward Activation of Worst Case Coupling Effects," in *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 2011, pp. 1969–1982.

[105] Predictive Technology Model (PTM). [Online]. Available: http://ptm.asu.edu/

[106] X. Qi, S. Lo, Y. Luo, A. Gyure, M. Shahram, and K. Singhal, "Simulation and analysis of inductive impact on VLSI interconnects in the presence of process variations," in *Custom Integrated Circuits Conference, 2005. Proceedings of the IEEE 2005*, 2005, pp. 309–312.

[107] A. Todri, A. Bosio, L. Dilillo, P. Girard, S. Pravossoudovitch, and A. Virazel, "A study of path delay variations in the presence of uncorrelated power and ground supply noise," in *IEEE 14th International Symposium on DDECS*, 2011, pp. 189–194.

[108] [Online]. Available: http://www.cadence.com/products/di/soc_encounter/pages/default.aspx

[109] A. Krstic, J.-J. Liou, Y.-M. Jiang, and K.-T. Cheng, "Test Conference, 2001. Proceedings. International," in *Test Conference, 2001. Proceedings. International*, 2001, pp. 558–567.

[110] K. Ganeshpure and S. Kundu, "Automatic Test Pattern Generation for Maximal Circuit Noise in Multiple Aggressor Crosstalk Faults," in *Design, Automation Test in Europe Conference Exhibition, 2007. DATE '07*, 2007, pp. 1–6.

[111] D. Gope and D. Walker, "Maximizing crosstalk-induced slowdown during path delay test," in *Computer Design (ICCD), 2012 IEEE 30th International Conference on*, 2012, pp. 159–166.

[112] M. Chen and A. Orailoglu, "Examining Timing Path Robustness Under Wide-Bandwidth Power Supply Noise Through Multi-Functional-Cycle Delay Test," in *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 2014, pp. 734–746.

[113] K.-T. C. Angela Krstić, "Delay fault testing for vlsi circuits," in *1st Edition*. Springer US, 1998.

[114] A. Asokan, A. Todri-Sanial, A. Bosio, L. Dilillo, P. Girard, S. Pravossoudovitch, and A. Virazel, "A Delay Probability Metric for Input Pattern Ranking Under Process Variation and Supply Noise," in *VLSI (ISVLSI), 2014 IEEE Computer Society Annual Symposium on*, 2014, pp. 226–231.

[115] ——, "Path Delay Test in the Presence of Multi-Aggressor Crosstalk, Power Supply Noise and Ground Bounce," in *Design and Diagnostics of Electronic Circuits Systems, 17th International Symposium on*, 2014, pp. 207–212.

[116] H. Li, P. Shen, and X. Li, "Robust test generation for precise crosstalk-induced path delay faults," in *VLSI Test Symposium, 2006. Proceedings. 24th IEEE*, 2006, pp. 6 pp. – 305.

[117] B. Li, N. Chen, M. Schmidt, W. Schneider, and U. Schlichtmann, "On hierarchical statistical static timing analysis," in *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, 2009, pp. 1320–1325.

[118] H. Chang and S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single PERT-like traversal," in *Computer Aided Design, 2003. ICCAD-2003. International Conference on*, 2003, pp. 621–625.

[119] S. Onaissi and F. Najm, "A Linear-Time Approach for Static Timing Analysis Covering All Process Corners," in *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 2008, pp. 1291–1304.

[120] V. Agaram and J. Venegas, "Insights into process reliability through simulation," in *Reliability and Maintainability Symposium (RAMS), 2015 Annual*, 2015, pp. 1–6.

[121] F. Galarza-Medina, J. Garcia-Gervacio, V. Champac, and A. Orailoglu, "Small-delay defects detection under process variation using Inter-Path Correlation," in *VLSI Test Symposium (VTS), 2012 IEEE 30th*, 2012, pp. 127–132.

[122] X. Wang, M. Tehranipoor, and R. Datta, "Path-RO: A novel on-chip critical path delay measurement under process variations," in *Computer-Aided Design, 2008. ICCAD 2008. IEEE/ACM International Conference on*, 2008, pp. 640–646.

[123] R. Tayade, S. Sundereswaran, and J. Abraham, "Small-Delay Defect Detection in the Presence of Process Variations," in *Quality Electronic Design, 2007. ISQED '07. 8th International Symposium on*, 2007, pp. 711–716.

[124] G. Yu, W. Dong, Z. Feng, and P. Li, "A Framework for Accounting for Process Model Uncertainty in Statistical Static Timing Analysis," in *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, 2007, pp. 829–834.

[125] H. Xu, V. Pavlidis, W. Burleson, and G. De Micheli, "The combined effect of process variations and power supply noise on clock skew and jitter," in *Quality Electronic Design (ISQED), 2012 13th International Symposium on*, 2012, pp. 320–327.

[126] K. Peng, M. Yilmaz, M. Tehranipoor, and K. Chakrabarty, "High-quality pattern selection for screening small-delay defects considering process variations and crosstalk," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, 2010, pp. 1426–1431.

[127] M. Orshansky and K. Keutzer, "A general probabilistic framework for worst case timing analysis," in *Design Automation Conference, 2002. Proceedings. 39th*, 2002, pp. 556–561.

[128] N. Weste and K. Eshraghian, "Principles of cmos vlsi design: A systems perspective," in *2nd Edition*. Addison - Wesley, New York, 1993.

[129] Y. Ismail, E. Friedman, and J. Neves, "Equivalent Elmore delay for RLC trees," in *Design Automation Conference, 1999. Proceedings. 36th*, 1999, pp. 715–720.