Testability Issues in Nanometer Technologies: A New Test Methodology and Worst-Case Path Delay Estimation Technique

A THESIS SUBMITTED FOR THE DEGREE OF

Doctor of Philosophy

by

Ankush Srivastava

(Roll No. 124077002)

Under the guidance of

Prof. Virendra Singh and Prof. Kewal K Saluja

Department of Electrical Engineering

Indian Institute of Technology Bombay

Mumbai 400076, India

May 2018

Dedicated to my wife, Dr. Surabhi Srivastava and my Guru, Prof. Virendra Singh

---

Declaration

I, **Ankush Srivastava**, with Roll No. **124077002** hereby declare that the material presented in the thesis titled

Testability Issues in Nanometer Technologies: A New Test Methodology and Worst-Case Path Delay Estimation Technique

represents original work carried out by me in the **Deparment of Electrical Engineering and Digital Networking Group** at **Indian Institute of Technology Bombay and NXP Semiconductor India Pvt Ltd** respectively during the years **2013 - 2018**. With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discusions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Date: Student Signature

Certificate

In my capacity as supervisor of the above-mentioned work, I certify that the above statements are true to the best of my knowledge, and I have carried out due diligence to ensure the originality of the report.

Prof. Virendra Singh Signature

Acknowledgements

I express my sincere gratitude to **Prof. Virendra Singh** for providing me enough reasons to join the research community. It is unexplained pain to maintain a fair balance among the official work, the research activities and my family, however, Prof. Virendra Singh has always accelerated and provoked my unconscious mind to never give up even at the worse situations. The rest of the research oriented threads are supported, guided and flourished under **Prof. Kewal K Saluja** and **Prof. Adit Dev Singh** from University of Wisconsin, US and Auburn University, US respectively. They have helped me in identifying the real issues faced by many circuit designers and the way it can be solved by merging scientific and engineering solutions together. We explored most of the ideas over $Skype^{TM}$ through several hours of discussions and currently our proposed idea get attention from global NXP team for methodology adoption. I would also like to thank **Prof. Janak H Patel** from University of Illinois, US to promote and observe the minor aspects of the research to solve a bigger problem.

My colleagues at NXP, **Dipesh Gupta** and **Amit Aggarwal** have really worked hard to get acquaint me to the basics of physical design, place and route, GDS preparation and performing SPICE simulations on various benchmark circuits. This might be a biggest advantage to work with some of the best minds who play with various commercially available circuit designing tools. It is worth to mention the contributions of my managers at NXP, **Mr. Gulshan Miglani** and **Mr. Rakesh Bakhshi** who never complaint while I was absent during the critical phases of few projects. Along with approving leave applications and releasing financial assistance to attend international conferences.

Many members from **Computer Architecture and Dependable Systems Lab** (CADSL) have contributed in promoting the research ideas and reviewing the papers much before I submit it to international conferences or journals. Thanks to **Newton, Rahul, Toral, Suryakant, Binod, Satdev, Darshit, Jay, Rohini** and **Nihar** for friendly welcome at IITB campus and providing valuable feedbacks on various topics. I would like to thanks my wife **Dr. Surabhi Srivastava** for her constant support, encouragement, patience and unyielding love during the low phase of my life. She is the victim of loneliness since last 5 years when I was busy carrying out the research and writing papers. I would also like to thanks my parents and in-laws for keep asking me about the thesis completion, that fuels the excitement to finish it on time.

IIT Bombay has always a place of quest. Though I am an ordinary researcher but my friends have always kept my quest alive for knowing the laws and principles of Nature. It is always pleasure to visit lush green campus of IIT that typically helps in boosting the research instinct with calm in my behavior. Last but no the least, my research progress committee (RPC) members, **Prof. D. K. Sharma, Prof. Madhav Desai** and **Prof. Sachin Patkar**, of Indian Institute of Technology Bombay for keep asking me the basic questions that motivate the radical thinking behind the serious research.

Vita

I am an industry sponsored research scholar at the **Department of Electrical Engineering**, **IIT Bombay**, Mumbai, India since January, 2013 and also working with **Digital Networking Group** of **NXP Semiconductor India Pvt Ltd** since July, 2007. I have completed my Master of Engineering (in Microlectronics) and Bachelor of Technology (in Electronics and Communication Engineering) from BITS Pilani and Uttar Pradesh Technical University, Lucknow, India in year 2007 and 2005 respectively. My current research interests include defining structured test architecture of complex SoC/ASIC, design verification, test, diagnosis, and failure analysis. I also hold several international patents, defensive publications along with research papers in world's premier conferences dedicated to the electronic test of devices, boards and systems. Below is the list of granted international patents, defensive publications and research papers (other than included in this thesis),

- Ankush Srivastava and Reinaldo Silveira, System and method for handling memory repair data, ***United States Patent No. US9496052 B2***, granted on Nov 2016.
- Ankush Srivastava, Bose-Chaudhuri-Hocquenghem (BCH) Decoder, ***United States Patent No. US 8806308 B2***, granted on Aug 2014.
- Amar Deogharia and Ankush Srivastava, Memory Built-in-Self-Testing in Multi-Core Integrated Circuit, ***United States Patent No. US 8549368 B1***, granted on Oct 2013.
- Ankush Srivastava and Prokash Ghosh, Method and Apparatus for Preventing Side Channel Attack on Memory System and Sensitive Registers of SoC, ***Intellectual Property disclosure no. IPCOM000250953D***.
- Rajan Agarwal and Ankush Srivastava, Method of Generating Efficient Stuck-at and Transition Delay Fault Scan Test Patterns using Fault Grading of Stuck at Patterns in Transition Mode, ***Intellectual Property disclosure no. IPCOM000212638D***.
- Rajan Agarwal and Ankush Srivastava, A Novel Approach for Resolution of Test Cube Issues in TestKompress EDT, Intellectual Property disclosure no. IPCOM000198994D.

- Ankush Srivastava and Ajay Prajapati and Vinay Soni, A Novel Approach to Improve Test Coverage of BSR Cells, ***Intellectual Property disclosure no. IPCOM000196589D***.
- Ankush Srivastava, Simultaneous Secured Memory-Built-in-Self-Test in Heterogeneous Multi-Core Integrated Circuit, Poster at ***IEEE International Test Conference***, 2013.
- Ankush Srivastava, Ajay Prajapati, Vinay Kumar, A Novel Approach to Improve Test Coverage of BSR Cell, ***IEEE International Test Conference***, 2010.

Publications

Included in Thesis

1. Ankush Srivastava, Virendra Singh, Adit Singh, Kewal K Saluja, A reliability-aware methodology to isolate timing-critical paths under aging, ***Journal of Electronic Testing- Theory and Applications*** (JETTA), Dec. 2017, Volume 33, pp 721-739.

2. Ankush Srivastava, Adit Singh, Virendra Singh, Kewal Saluja, Exploiting path delay test Generation to develop better TDF test for small-delay defects, ***IEEE International Test Conference***, Nov. 2017.

3. Ankush Srivastava, Virendra Singh, Adit Singh, Kewal K Saluja, Identifying high variability speed-limiting paths under aging, ***IEEE Latin American Test Symposium***, Mar. 2017.

4. Ankush Srivastava, Virendra Singh, Adit Singh, Kewal K Saluja, A Methodology for Identifying High Timing Variability Paths in Complex Designs, ***IEEE Asian Test Symposium***, Nov. 2015.

5. Ankush Srivastava, Virendra Singh, Adit Singh, Kewal Saluja, Path-based Approach to Identify Timing Critical Paths under Aging, ***IEEE Workshop on RTL and High Level Testing***, 2016.

6. Ankush Srivastava, Amit Agarwal, Rakesh Bakhshi, Effect of Static Stress in Burn-in Environment on Yield of Complex Designs, ***IEEE Workshop on RTL and High Level Testing***, 2015.

Abstract

Localized small delay defects (SDDs), for example degraded transistor drive strength caused by under-grown fin(s), are a growing concern in current FinFET and emerging gate all around (GAA) technologies. These SDDs may appear either during manufacturing process steps, such as ion implantation, etching, photo-lithography etc., or due to aging effects over the period in MOS transistors and interconnects under certain workload. This thesis addresses the shortcomings of conventional approaches to target SDDs, and is broadly divided into two parts. First part of the thesis discusses a new test methodology to cover SDDs more efficiently to reduce pattern volume and automatic test pattern generation (ATPG) runtime as compared to traditional techniques. We also modifies the definition of conventional SDD metric to improve confidence in covering small timing defects. In order to uncover distributed timing defects in a circuit, it is utmost important to identify a set of performance limiting paths whose timing delay closely matches silicon behavior. Second part of the thesis introduces a new approach of identifying a set of long paths that may limit the circuit performance under spatial or temporal statistical variations.

Gross-delay defects which are typically targeted by transition delay fault (TDF) based ATPG is ineffective in targeting small timing defects. This is due to the fact that it excites and propagates the fault effect through easy to control short or intermediate structural path instead of choosing longest timing path of a circuit. On the contrary, timing-aware (TA) ATPG uses circuit level timing information to sensitize longest testable path. Typically due to this reason, it is fairly effective in targeting small distributed delay defects. However, TA-ATPG is expensive in pattern volume and runtime. As the requirement of sensitizing longest timing path for each fault puts constraints on ATPG during reduced pattern set generation. The behavior of sensitizing longest path for each fault prunes the ATPG efficiency of covering many faults through a single pattern using fault simulation and implication approaches.

Path delay fault (PDF) based ATPG can target distributed small delay defects, assuming a set of timing critical paths is well defined. The practical issues with PDF-based ATPG are low path coverage and test application time which grows exponentially with increase in circuit size. In order to address the demerits of conventional SDD detection approaches, we propose a new test methodology. Our approach exploits path delay test generation to develop better TDF test, which efficiently generates ATPG patterns to cover SDDs more effectively. For a suite of benchmark circuits, our test methodology

resulted in approximately 12% reduction in pattern volume, 35% reduction in ATPG runtime and a 5% improvement in delay test coverage (DTC) as compared to the commercially available TA-ATPG approach.

The extra small delays caused by distributed defects would be absorbed on those paths which have higher slack margin and would never result in erroneous circuit response. In order to target SDDs, it is utmost important to identify long paths of a specified circuit. Usually, static timing analysis (STA) is used to identify such paths in a circuit at specified worst-case corner(s). The principal reason of using STA for design sign-off is that, it is substantially faster in estimating path delays as compared to dynamic timing simulations. However, it adds pessimism during delay calculations which typically results in additional number of paths to be categorized under performance-limiting paths. More number of critical paths overburden the timing-aware ATPG which generally results in increased pattern volume and ATPG runtime. Dynamic or SPICE level simulations can reduce the extra pessimism added by STA, however, they are much slower, non-exhaustive and input pattern dependent.

This thesis addresses the issues in identifying real timing critical paths that are the potential candidates for SDD testing. We discuss a new timing estimation algorithm that is reasonably accurate and substantially faster than commercially available SPICE level simulators. By using this new approach, we have showed that an approximately 50% of the unwanted additional paths added by pessimistic STA can be removed. Furthermore, nearly 94% accuracy in worst-case path delay estimation is achieved and compared with dynamic timing simulation results obtained from various benchmark circuits such as ITC'99, IWLS'2005 and ISCAS'89. We also validated the algorithmic complexity and its performance on various benchmark circuits. It is found that our approach is approx. 10-15 times faster than SPICE level simulations in identifying critical paths under worst condition.

Keywords

VLSI Testing, Path-delay fault based Testing, Transition-delay fault based Testing, Timing-aware ATPG, Spatial and Temporal CMOS reliability, Systematic and random variations, Aging effects, Timing-critical paths

Notation and Abbreviations

SEU	Single Event Upset	PVT	Process Voltage Temperature
CMOS	Complementary MOS	LOC	Launch On Capture
NMOS	N-type MOS	LOS	Launch On Shift
PMOS	P-type MOS	PI	Primary Input
RDF	Random Dopant Fluctuation	PO	Primary Output
LER	Line Edge Roughness	GAA	Gate All Around
LWR	Line Width Roughness	S-TDF	Speed Limiting TDF
FET	Field Effect Transistor	ECSM	Effective Current Source Model
FinFET	Fin Field Effect Transistor	V_{th}	MOSFET Threshold Voltage
ELFR	Early Life Failure Rate	MuFET	Multi-gate FET
HTOL	High Temperature Operating Life	BI	Burn In
BTI	Bias Temperature Instability	DPPM	Defective Parts Per Million
NBTI	Negative Bias Temperature Instability	BIB	BI Board
PBTI	Positive Bias Temperature Instability	SoC	System-on-Chip
HCI	Hot Carrier Injection	IC	Integrated Circuit
MOSFET	Metal Oxide Semiconductor FET	ASIC	Application Specific IC
SEM	Scanning Electron Microscope	ATE	Automatic Test Equipment
CMP	Chemical Mechanical Planarization	PCB	Printed Circuit Board
TDF	Transition Delay Fault	JTAG	Joint Test Action Group
SDD	Small Delay Defect	DFT	Design For Test
PDF	Path Delay Fault	ATPG	Automatic Test Pattern Generator
SDF	Segment Delay Fault	BIST	Built In Self Test
STA	Static Timing Analysis	MBIST	Memory BIST
SSTA	Statistical STA	LBIST	Logic BIST
PF	Probability Density Function	VDD	Power Supply

GND	Ground
SPEF	Standard Parasitic Exchange Format
SPICE	Simulation Program with IC Emphasis
DSPF	Detailed Standard Parasitic Format
TA-ATPG	Timing Aware ATPG
DTC	Delay Test Coverage
DSM	Dropping based on Slack Margin
RTL	Register Transfer Level
TCF	Timing Critical Fault
C_L	Load Capacitance
L_g	MOS Gate Length
W_g	MOS Gate Width
t_{ox}	Oxide Thickness
F_{max}	Max Operating Frequency
LSSD	Level Sensitive Scan Design
MISR	Multiple Input Signature Register
MSDFF	Multiplexed Scan D Flip-flop
PPI	Pseudo Primary Input
PPO	Pseudo Primary Output
RAM	Random Access Memory
SE/SEn	Scan Enable
VLSI	Very Large Scale Integration
W_f	FinFET fin Width
H_f	FinFET fin Height

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Today's IC design involves dealing with complex VLSI systems and increasingly large number of design constraints. Modern technology demands techniques for efficiently designing high-performance low-power integrated circuits while requirements for shorter time-to-market push down the design time. Apart from design cycle time reduction, these circuits demand technology scaling from planar to 3D FinFETs structure to improve transistor propagation delay and associated static/dynamic power. Technology scaling beyond 28nm node comes with a penalty of more number of **localized small delay defects** (SDDs) which typically originates from spatial and temporal statistical variations in the circuit. SDDs are growing concern in current FinFET and emerging gate all around (GAA) technologies [4, 5]. Hence, the testing of circuit with millions of design components relies on effective and efficient test techniques for reliably screening the timing defects.

Deep-submicron integrated circuit manufacturing technology nodes are suffering from inherent within-die (intra-die), die-to-die (inter-die), wafer-to-wafer and lot-to-lot parameter fluctuations for both gates and interconnects [3, 5–10]. The changes in physical parameters usually impact the circuit performance and may result in permanent as well as temporary faulty response. The complementary metal-oxide semiconductor (CMOS) circuit reliability issues can be broadly categorized into two dimensions: space and time. The spatial unreliability can be defined as unintended changes in physical parameters

of the circuit when it is fabricated on different or the same wafer at distance apart. Whereas temporal unreliability is defined as unintended changes in physical parameters of the circuit over a period of time under certain environmental conditions and specified workload. Before we discuss the importance of testing for integrated circuits, electronic components and electronic systems, let us take a look at various dominant spatial and temporal statistical variations and how easily they can be detected.

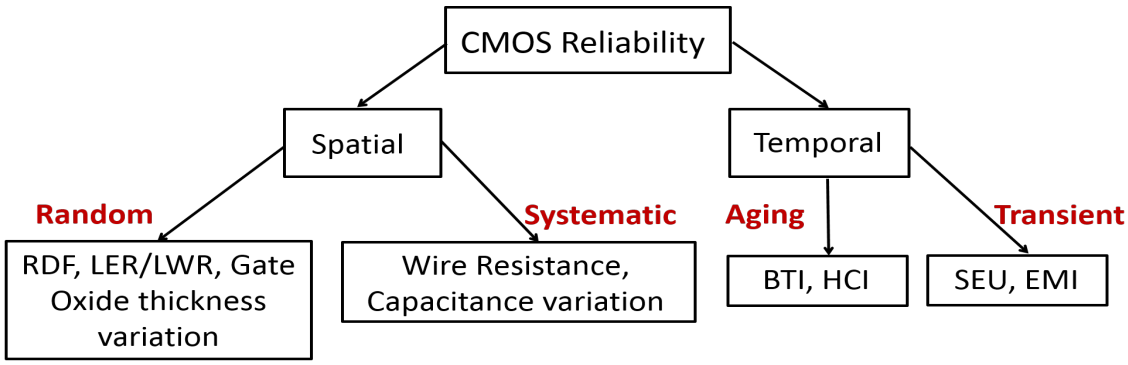


Figure 1.1: Categorization of reliability issues in CMOS circuits

1.1 Spatial CMOS Unreliability

The spatial reliability issues can be observed just after the chip production and can be classified into random and systematic variations as shown in Fig. 1.1. Systematic variations are usually layout dependent such as design rules for manufacturability, e.g. interconnect minimum width spacing (pitch) at different metal layers, polysilicon thickness (width) etc. Integrated circuit fabricated in nanometer technology nodes usually suffer from subtle additional delays due to process variations on both gates and interconnects [6, 9, 11]. The interconnect parameter fluctuations dominate the systematic process variations and usually have a smaller impact on circuit delays as compared to delay variations on transistors [11]. However, recent study shows that the interconnect delays are becoming dominant on sub-28nm CMOS technology nodes and assume to be more critical for future fabrication nodes [5]. Interconnect resistance changes either due

to structural and morphological effects [10] or during multi-step chemical mechanical planarization (CMP) [7, 8] process which typically results in dishing and erosion [8]. In order to improve the chip manufacturability and overcome metal thickness variations, dummy metal fills are used to fill vacant space on the specified metal layer. The dummy fill insertion alone can change the total interconnect capacitance by more than 10% [7, 8].

The variations in implanted dopant concentration, which forms the conducting channel in NMOS or PMOS transistors, is typically known as random dopant fluctuations (RDF). As the transistor size is getting reduced, a minimal dopant concentration is required [12–14] to build conducting channel in the transistors. Thus, even if fewer number of impurity atoms get added or deleted in the channel, it usually results in more than expected threshold voltage variations. In [15], it is shown that an approximately 60% of the total random variations in transistors, built on 45nm and 65nm planar CMOS technology nodes, are dominated by RDF. On planar CMOS technology nodes, RDF is assumed to be the major contributor of device mismatch when the transistors are fabricated on the same wafer. The standard deviation on threshold voltage is typically represented by [14],

$$\sigma(V_{TH}) \propto \frac{t_{ox}}{\epsilon_{ox}} \frac{\sqrt[4]{N}}{\sqrt{W_{eff} L_{eff}}} \tag{1.1}$$

where t_{ox} and ϵ_{ox} are the effective thickness and permittivity of the oxide respectively. N is the number of impurity/dopant atoms in the channel while W_{eff} and L_{eff} are the effective channel width and length of the transistor. It can be seen in Eqn. 1.1 that the absolute values of N and t_{ox}/ϵ_{ox} would reduce with transistor miniaturization. However, the reduction in V_{TH} deviation is offset by reduction in the multiplication factor of W_{eff} and L_{eff} in planar CMOS technology nodes. With the introduction of three dimensional (3-D) FinFET structure, it is found that V_{TH} can be adjusted by tuning the retrograde body doping depth [13]. In nanometer technology nodes, RDF continues to be the dominant contributor of overall random variations in the planar CMOS and 3-D FinFET structures [16].

Line edge and line width roughness (LER and LWR) originate from photo-lithographic

techniques while transferring the gate patterns on silicon using sub-wavelength lithography. The effects of LER and LWR are getting dominant as the transistor gate length and width [17] continue to shrink with technology scaling. The percentage variations in line edge and width of polygate result in increased sub-threshold current conduction, threshold voltage variations and short-channel effects [18, 19]. LER and LWR are expected to be more dominant in 3-D FinFET CMOS technologies and also expected to overcome the dominance of RDF in random variations [16]. However, we will not discuss the details of device level effects of LER and LWR on transistors. Our main focus is to translate the random variations in device parameters into percentage change in V_{TH} of transistors. This would certainly help us in identifying performance-limiting paths of a circuit which is discussed further in chapter 5.

1.2 Temporal CMOS unreliability

Aging is a phenomenon of gradual circuit degradation when it is operating under certain workload at specified environmental condition over the time. Unfortunately, the catastrophic failures or change in circuit performance due to aging (temporal effects) can not be observed just after the chip production. The devices which fail early in life or during stable operating life follow a failure curve that is typically known as the *bathtub curve* [20]. The failure curve helps in predicting the life cycle of such devices by broadly categorizing it into three periods: *early, stable, and wear-out*. The bathtub curve can also be broken down as per user's criteria. It defines the allowed failure rate of acceptable product by defining the allowed failure rate over a defined period of life. Typically, one requirement is for the failure rate during *Early Life Failure Rate* (ELFR) period [21], and the other requirement is for the failure rate over the longer period of life known as *High Temperature Operating Life* (HTOL) period [22].

Before we review the other aspects of integrated-circuit aging mechanism and its modeling, it is necessary to discuss the most important reliability issues, such as Bias Temperature Instability (BTI) and Hot Carrier Injection (HCI), observed in sub-32nm

CMOS technology nodes. NBTI (negative BTI) and PBTI (positive BTI) are the key reliability issues in PMOS-FETs and NMOS-FETs when negative-bias voltage is applied across MOS gate. HCI is an issue in NMOS-FETs when gate voltage is comparable to drain voltage [16, 23, 24] at elevated temperature. As holes are heavier than electron, they do not participate in hot carrier injection rather than tunnel into oxide defects and contribute in generation of oxide-silicon interface states. Nevertheless, NBTI and HCI remain a major reliability concerns for PMOS-FETs and NMOS-FETs respectively. The reliability concerns continue to exist as DC supply voltage scaling is much slower than the transistor size scaling for sub-32nm technology nodes [16]. A geometrical unification of the theories of NBTI and HCI are well studied and elaborated for planar MOSFETs and multi-gate field-effect transistors, e.g., MuFETs and FinFETs [25, 26] which will be discuss in chapter 5.

To guarantee the correct device operation over a defined period, usually burn-in stress tests are used to age the circuits in shortest duration of time without waiting for years to see final aging effects. Burn-In (BI) stress test is used in manufacturing flow to screen out weak units associated with process anomalies, assembly imperfections, gate oxide sensitivities, etc. If the BI tests are not performed on the devices shipped to customer then typically there is a high chance of chip failure in customer application boards. Device return from customer contributes to increased defective parts per million (DPPM). The objective of BI test is to toggle as many on-chip elements as possible and is typically needed for *device qualification* to begin full production. Depending on process maturity and required product quality, production BI time is defined per individual device conditions. For product qualification, there are two types of BI effects to be monitored:

1. Early life failure rate (ELFR) is exercised for validation of suggested manufacturing BI time and *Weibull analysis* [27](Weibull analysis is a tool that can be used to classify failures and to model failure behavior) is performed to estimate ELFR rate.

2. High temperature operating life (HTOL) is used in engineering mode to assess

long-term reliability of the product and its often called Life Test.

Both stresses are performed in the BI Oven, where a statistical valid sample size of units are loaded into BIB (burn in board) and stimulated by special pattern driven from the board driver. BI device conditions could differ from product to product and technology node to technology node.

Identifying a set of long paths, that potentially can fail or degrade the circuit performance during HTOL period, in stipulated design cycle time is a real challenge for system-on-chip (SoC) designers. The temporal reliability issues in CMOS circuit have attracted many researchers and engineers to develop more accurate models of BTI and HCI to predict failure rate of various circuits under different workloads and environmental conditions. Most of the previous work [23, 25, 28, 29] either aims at developing an accurate model to fix key reliability issues in terms of threshold voltage shift (or change in substrate current) or discusses the impact of aging on circuit level designs.

1.3 Importance of Testing

After discussing the causes of statistical spatial and temporal variations, let us find out the requirements of CMOS circuit testing. In early 70's, none of the circuit designers were worried in generating hierarchical test patterns to target manufacturing defects of a circuit. At that time, the circuits typically consist of few hundreds to thousands of transistors on the die. The small circuits are guaranteed to work based on functional patterns and it is potentially possible because the number of primary inputs are relatively small which can be covered exhaustively. The **Moore's law** has changed the paradigm of circuit testing after the integrated circuit (IC) manufacturers started putting double the transistors on same silicon area every 18 months and it is still going on below 14nm technology node. The whole process resulted in adding more than several million flip-flops on the same die that is currently reported for state-of-the-art multicore communication processor built by NXP^{TM} [30].

More number of transistors on a die enables more deteriorating interactions that

may result in faulty response due to power supply noise, cross-talk, single event upset, electromagnetic interaction etc. Apart from this, we already discussed the fluctuations in circuit parameters when the circuit is fabricated on the same wafer or on different wafer. The state-of-the-art ASIC/SoC usually consists of 200-600 inputs or outputs or bi-directional pads (primary input/output) and thus, it is impossible to test the circuit at-speed by conventional functional patterns using primary inputs. In order to cover all such potential issues that may result in circuit malfunction due to manufacturing defects, we must have a ad-hoc or hierarchal testing approach to ensure good devices are shipped to the end users. In the absence of such testing methodologies, no one can guarantee the operating life of the fabricated circuits over the time at user's end. Currently all commercial chip design houses sell their chip with desired defect coverage which usually shows the confidence of covering different class of faults (such as stuck-at or transition). This is very important in those application areas where embedded circuits are deployed to save human lives from accidental threats. Today, testing is being carried out at different levels to ensure the reliability of complete system. It starts from intellectual property (IP) design verification, IC level testing using automatic test equipment (ATE), printed circuit board (PCB) testing using joint-test action group (JTAG) standards, and system level testing etc. This thesis covers standalone IC testing approach using structural test patterns applied through ATE based drivers. The present work can be extended easily to target those delay defects which are yet not modeled in 3D FinFET structures.

1.3.1 Ad-Hoc and Structured Test Approach

Testability is a design attribute that measures how easy it is to create a program to comprehensively test a manufactured circuit. Traditionally, design and test processes are kept separate, with test considered only at the end of the design cycle. However in contemporary design flows, test merges with design much earlier in the process, creating what is called a design-for-test (DFT) process flow. Testable circuitry must have controllable and observable intermediate nodes either by replacing a normal flip-flop with a scanable flip-flop (discussed later in this section) or inserting test points in combinational

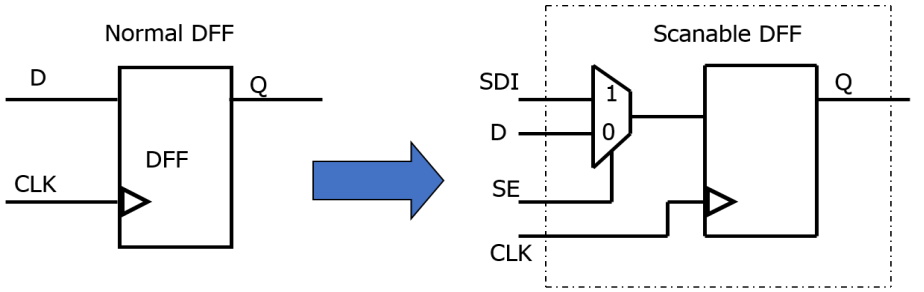
logic part of the circuit. In a testable design, specific values are applied on the primary inputs which results in some values on the primary outputs and based on the circuit behavior, it is decided whether or not the internal circuitry worked properly. To ensure maximum testability, designer must employ special DFT techniques at specific stages in the development process of integrated circuits. At the highest level, there are two main approaches to DFT: ad hoc and structured. **Ad hoc DFT** implies using good design practices to enhance a design's testability without making major changes to the design style. Some ad hoc techniques include:

- Minimizing redundant logic
- Minimizing asynchronous logic
- Isolating clocks from the logic
- Adding internal control and observation points

Using these practices throughout the design process improve the overall testability of a design. However, in practice, employing ad-hoc techniques during design-phase of complex SoC is cumbersome. Therefore, to avoid additional complexity, structured DFT techniques are used early in the design cycle. Structured DFT techniques can enhance the testability much higher than comparable ad-hoc techniques as it did not require any specialized circuit designer or experienced test engineer to improve testability. If the prerequisites of the structured test approach are followed during initial circuit synthesis then the pattern generation and test pattern conversion to ATE format are reasonably fast using commercially available automatic test pattern generation (ATPG) tools. Fortunately, structured DFT techniques are fully supported by various commercial tools.

Structured DFT provides a more systematic and automatic approach to enhance design testability with some additional changes in the circuit that never influence the functional logical operations. The main objective of structured DFT approach is to increase the controllability and observability of the intermediate nodes in a circuit such that it does not impact the functional behavior. The most common is the scan design

technique, which modifies the internal sequential circuitry of the design as shown in Fig. 1.2.



D	CLK	Q
0	0->1	0
1	0->1	1
0	1->0	Q_{n-1}
1	1->0	Q_{n-1}

D	SDI	CLK	SE	Q
0	-	0->1	0	0
1	-	0->1	0	1
-	0	0->1	1	0
-	1	0->1	1	1
-	-	1->0	-	Q_{n-1}

Figure 1.2: Normal and scanable D flip-flop

A normal D flip-flop usually consists of two inputs, D as data input and CLK as clock input, and an output Q. Assuming the flip-flop shown is positive edge triggered which captures the D input on each positive edge of clock CLK while it continues to restore the previous value of data D for negative clock edges as shown in the truth-table. In order to make D flip-flop 100% controllable and observable, a multiplexer is being added on data input of flip-flop whose control is SE (scan enable) and SDI as scan data input. If the SE is at logic 0 then flip-flop would work as usual sequential element, however, if SE is kept at logic 1 then SDI would be captured into the sequential element at each positive edge of clock CLK. The scanable flip-flip is the most adopted technique for chip testability by SoC/ASIC designers and DFT engineers as it is completely supported by the highly optimized tools at each design phase. There are multiple structured test approaches are available such as,

- Scan design [31] technique which ideally requires all sequential elements (except sequential elements inside embedded memories) to be 100% controllable and observable. This helps in achieving maximum test coverage using ATPG tools.

- Built-in Self-Test (BIST) [32], which requires inserting testing hardware within the device itself. This broadly categorized into two mainstream techniques: Memory BIST to test embedded memories using March algorithm [33] and Logic BIST [34] to test logic except memories.

- Boundary scan [35], which increases board level testability by adding circuitry to the IO pads of a chip. Essentially, boundary scan register (BSR) cells are added between external chip pad and core logic with an objective to control and observe the data going into or out of the chip during functional mode of operation. The BSR cells are stitched into a single chain which can be accessed through system JTAG controller [36]. It can also be optionally used to control or observe the data of internal functional registers during device normal mode of operation.

This thesis discusses a new test methodology viable for scan design only and does not add any extra hardware or logic while generating defect oriented test pattern for a specified circuit.

1.3.2 Understanding Scan

Scan is a design methodology that intend to replace all sequential elements (excluding embedded memories) in the design with their scannable equivalents, as discussed in Fig. 1.2, to increase intermediate node observability and controllability. In order to make use of scanable cells in effective and efficient manner, flip-flops are stitched (connected) into a single or multiple scan chains based on the design requirements. Fig. 1.3 shows a circuit which consists of 20 D flip-flops and various combinational logics to achieve certain functionality. In order to test the above circuit for manufacturing defects then it would require $2^{(6+4)}$ functional vectors, where 6 refers to the number of primary inputs and 4 to the sequential depth. If we replace all sequential elements with corresponding scanable versions then it would certainly help in increasing the intermediate node coverage. This can be achieved as shown in Fig. 1.4.

Now we have four scan-data input (SDI[0:3]) and scan-data output (SDO[0:3]) with

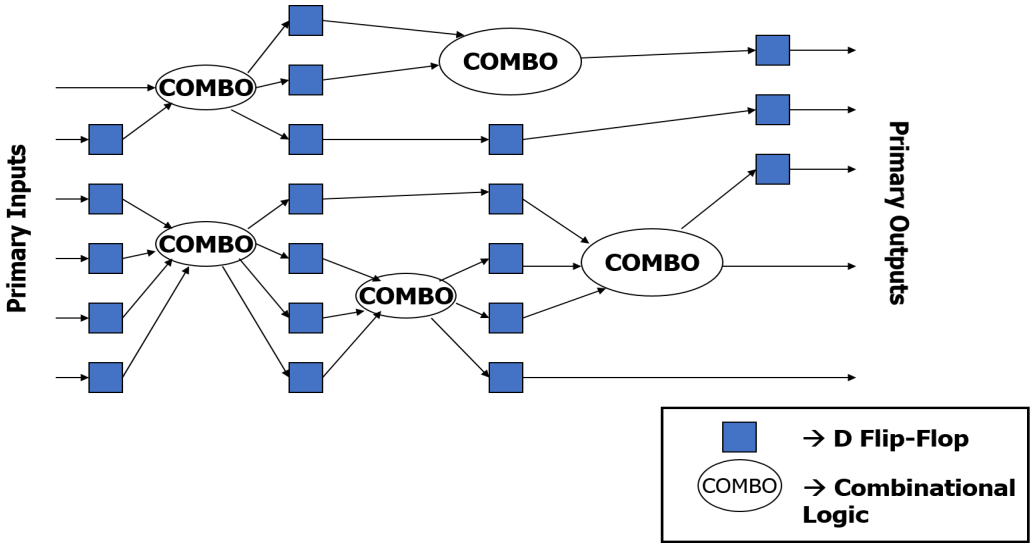


Figure 1.3: Normal representation of the circuit

a dedicated scan control *SE* broad-casted to each D flip-flop. Scan operation are divided into two modes. First is *Shift Mode*, in which the external driver shifts-in pre-identified values into multiple scan chains via scan data inputs *SDIs* of the circuit. Second is *Capture Mode*, in which flip-flop captures the functional data launched from an another set of flip-flops. In scan mode, scan chains behave as serial-in, serial-out shift registers during scan data load/unload (shift mode) while it work as parallel-in, parallel-out registers during capture mode. The idea is to control and observe the values of design's sequential elements such that test generation and fault simulation tasks can be simplified. This would certainly help in leveraging the combinational ATPG to cover maximum intermediate nodes for defect coverage. A higher coverage can only be achieved if the data input and out of sequential elements are controlled through external driver.

1.4 Delay Fault Models

To begin with, it is fairly important to understand the various nomenclatures traditionally used in the literature on testing. Defect, error and fault are the common words used interchangeably by various engineers [37]. ***Defect*** *can be defined as an unintended differences between the desired design and implemented hardware which may or may not result*

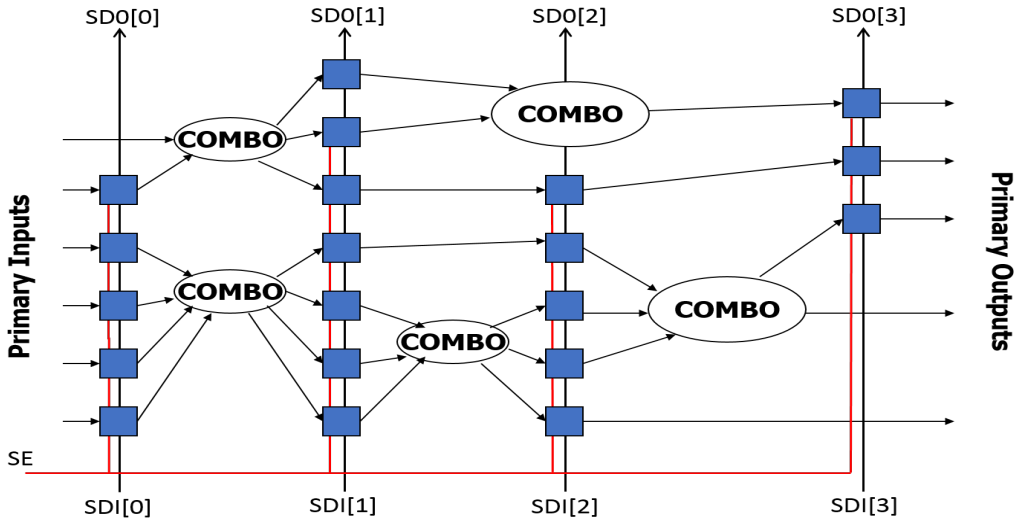


Figure 1.4: Scan representation of the circuit

in erroneous circuit response. The origin of defect can be categorized under one of the following: material defect (bulge, cracks, crystal imperfection etc. in semi-conducting material), process defect (random dopant fluctuation, line edge/width roughness etc.), packaging defect (edge crack, increased inductive/capacitive interaction between bond etc.) and aging defect (degradation due to bias temperature instability, hot carrier injection etc.). The defects which incorporate into device during fabrication or packaging can be tested just after the chip production using ATE or test bed. While few of them may appear over the period of time under certain workload and environmental conditions, which require various aging models to identify a probable set of timing critical defects.

Error *can be defined as the incorrect logical response of the circuit due to various defects. The model of a defect at an abstract functional level is called* ***fault model***. In other words, if a physical defect can be described or modeled in terms of logical function(s) then it can be defined as *fault*. Some defects are catastrophic such as stuck faults which result in intermediate circuit nodes either stuck at logic 0 or logic 1. Whereas the resistive bridges, resistive opens, transistor stuck-open/stuck-short etc. translate to increase in gate or wire delays which must be tested at rated functional frequency.

1.4.1 Stuck-at Fault

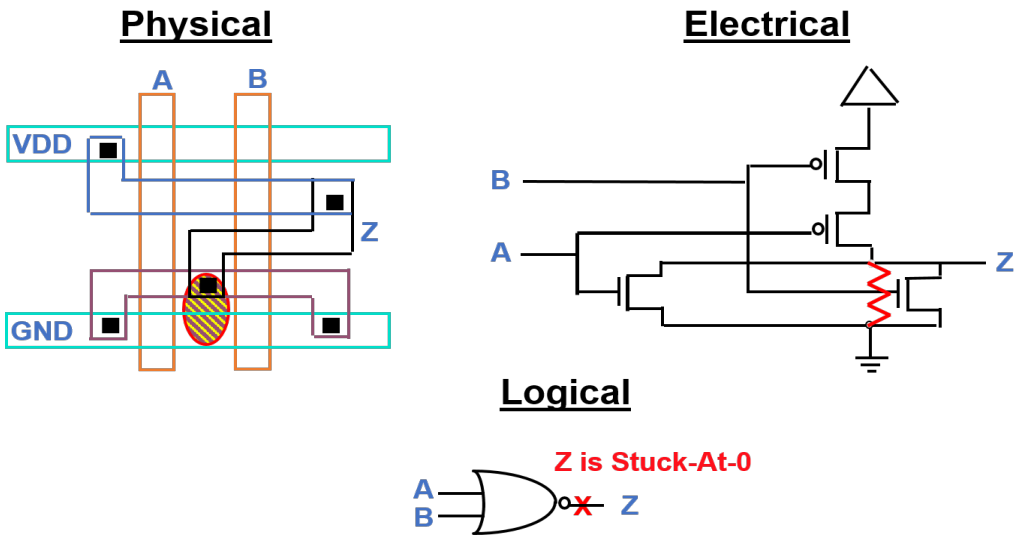


Figure 1.5: Fault Modeling

The stuck-at fault model assumes that a specified node is either shorted to power supply *VDD* or ground *GND* through an unintentional resistive path that affects one node at a time [38]. Fig. 1.5 shows a two input NOR gate with physical, electrical and logical views. In physical view, a low resistive path can be observed between output Z and ground GND rail. If we translate this defect at circuit level then a resistive path connects the output to ground and similarly in logical view, output Z of two input NOR gate must be tested for the presence of stuck-at 0. A scanning electron microscope (SEM) image can be seen in Fig. 1.6 where voltage supply (FA_Vdd) rail and metal layer M9 are connected through unintentional resistive bridge during device manufacturing processes. This results in stuck-short to power supply while a case of stuck-open can be observe in Fig. 1.7. It clearly shows a void formation between via and metal layer, due to which two metal layers are virtually disconnected. In order to test the stuck-at fault at any node, it must be excited through an opposite value and propagate the fault effect through one of the sensitizable path.

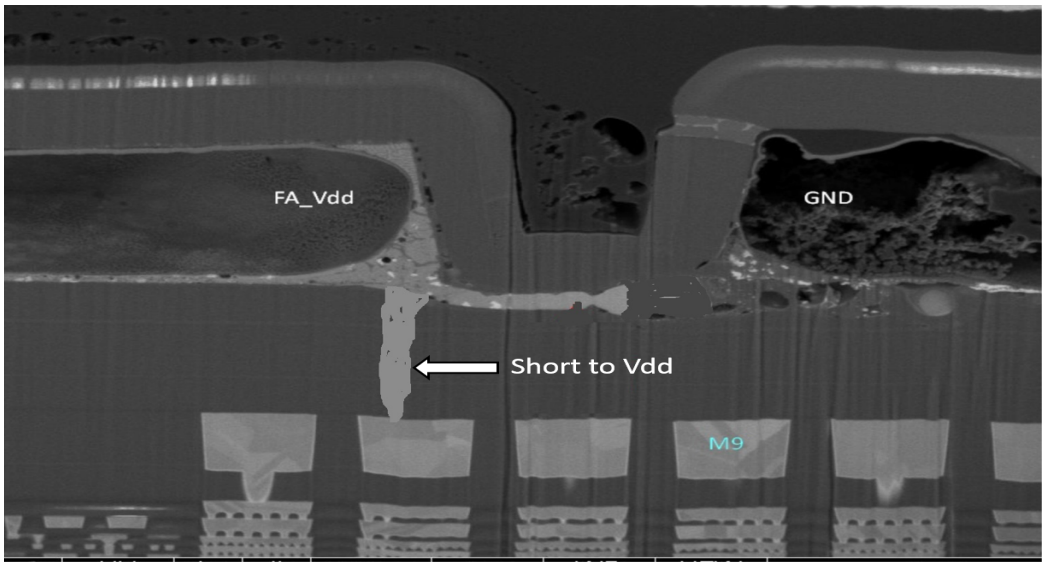


Figure 1.6: Stuck-short Defect: A SEM cross-section shows a short bridge between power supply (FA_Vdd) and metal layer (M9)

1.4.2 Transition Delay Fault

The transition delay fault model [39] (TDF) assumes that the delay defect at a specified circuit node is large enough such that any signal transition passing through this node will be delayed past the clock period. TDF is sub-divided into slow-to-rise and slow-to-fall faults which require a two-pattern sequence <V1, V2> to initialize and launch transition through the specified node. Vector V1 initializes the intermediate nodes which fall in the fan-in cone of target faulty node of the circuit. Whereas V2 creates the desired transition to excite and propagate the fault effect to observable output or pseudo-output (scanable flip-flop). Fig. 1.8 shows a logical view of a circuit, having slow-to-rise transition fault at intermediate node A, with the circuit having three primary inputs IN1, IN2, IN3 and two primary outputs OUT1, OUT2 respectively. In order to initialize the circuit to test TDF at node A, vector V1 drives IN1, IN2 to logic 0 and IN3 to logic 1. Thus vector V1 initializes the node A to 0. Vector V2 creates the transition 0 to 1 at IN2, and hence to node A which propagates the fault effect to an observable primary output, OUT1 in our case. A resistive bridge between the metal layers can slow down the signal transition which typically results in increased propagation delay across the victim interconnect.

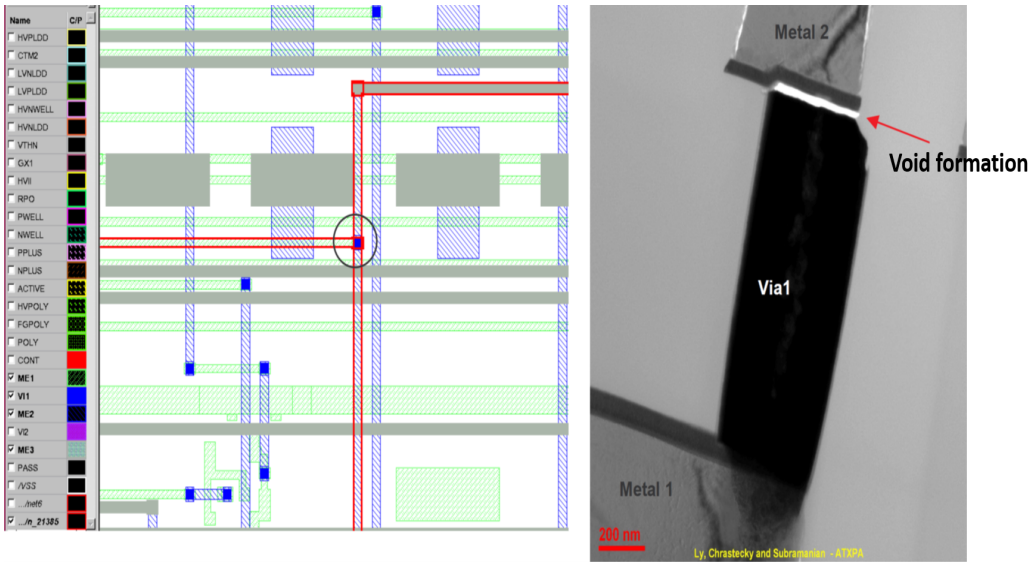


Figure 1.7: Stuck-open Defect: A SEM cross-section shows a void formation between Metal 2 and Via1

One such case can be seen in Fig. 1.9 where a unintentional high resistive bridges can be observed across the M9 metal rails.

In today's state-of-the-art networking/automotive/mobile chips, TDF is most prevalent fault model in the industry usually because, 1) number of TDF faults increases linearly with increase in number of gates [37], 2) ATPG algorithm and stuck-at fault simulator can be easily modify to cater transition delay faults, 3) it can potentially detect delay defects like shorts, coupling defects, opens etc. that are missed by stuck-at tests, and 4) fault lists, coverage metrics are similar to stuck-at faults. However, the main disadvantage of using TDF based ATPG are, 1) it may miss distributed small delay defects as fault propagation typically sensitizes short or intermediate paths [38], 2) it assumes delay fault only affects one gate at a time which is not practical, and 3) it does not choose structurally longest path to propagate fault effect.

1.4.3 Path Delay Fault

Instead of targeting individual nodes or gates in a circuit, Smith [40] has introduced path delay fault (PDF) model that targets specified path of a circuit as shown in Fig. 1.10. A

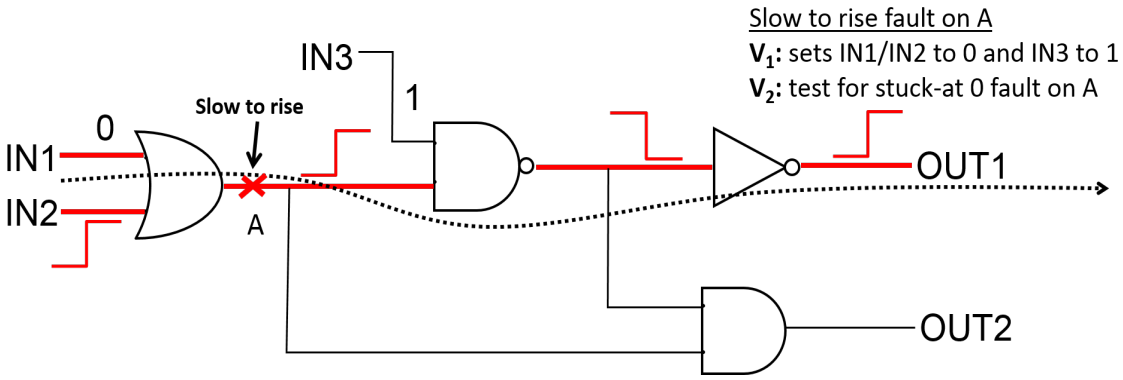


Figure 1.8: Transition delay fault: logical view

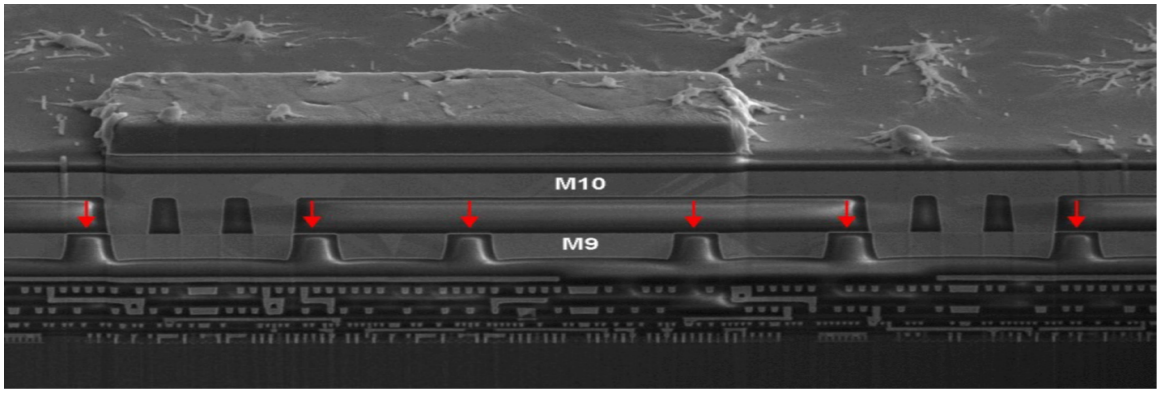


Figure 1.9: Resistive Bridge Defect: A SEM cross-section image showing incomplete metal polish (over M9), shown in red arrows

path can be defined as an ordered set of gates and interconnects traversing from primary or pseudo-primary input to primary or pseudo-primary output. The pseudo primary input or output refers to scanable D flip-flop. In Fig. 1.10, the transition starts from input IN2 and propagates along the target path that terminates at output OUT1. Since PDF covers the complete path, it has capability to target distributed small-delay defects (SDDs) that fall along it. An erroneous response of the circuit, while targeting path delay faults, points to a fact that the timing delay(s) of path(s) is(are) more than the specified clock period of the circuit. PDF is most capable fault model to target SDDs, however, it has issues in practical adoption.

Before the shortcomings of PDF are being discussed, it is important to differentiate

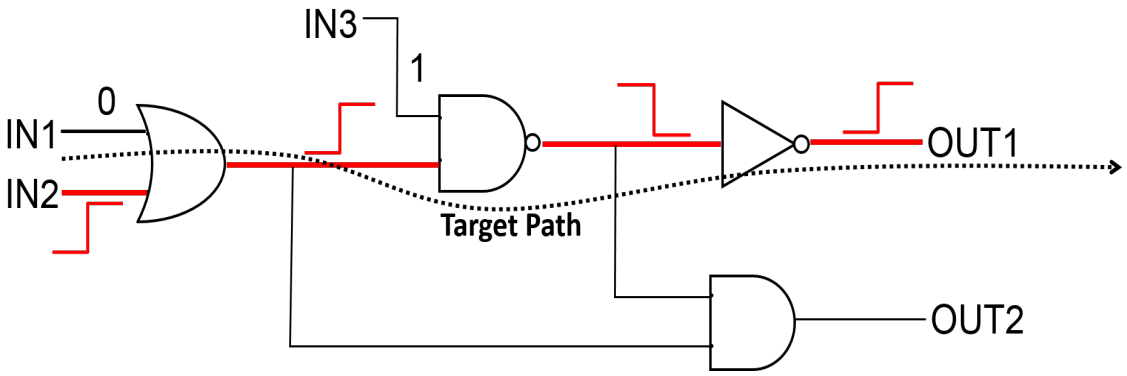


Figure 1.10: Path delay fault: logical view

Robust and *Non-robust* pattern generation criteria. If the sensitization of the fault through a target path is independent of off-input signal arrival time such that it never blocks the fault effect propagation then it is categorized as robust test. In case, the fault detection is dependent on off-input arrival time then the test pattern belongs to a non-robust category. We will further discuss the pattern generation criteria in section 2.1.3. The major problem with adoption of PDF based ATPG is, the number of target faults grow exponentially with increase in circuit size. This becomes unmanageable from ATPG runtime and pattern volume perspective. This specific problem can be solved by generating patterns for limited set of paths such that we do not loose any valid small timing defects on performance limiting paths. The another problem is, low path delay fault coverage even if the non-robust test patterns are used to target a limited set of paths. In order to partially fix this issue, *Heragu et al.* have proposed *segment delay fault* [41] model which is discussed in next sub-section.

1.4.4 Segment Delay Fault

If a complete path can not be tested robustly or non-robustly then it is fairly a good approach to atleast robustly test a part of the path to improve confidence of covering small timing defects. A segment of a path may consists of one or more than one gates depending on the chosen segment length. The segment delay fault [41](SDF) model is a trade-off between transition delay fault model and path delay fault model while targeting

delay defects. Assume that the segment length is denoted by L and the maximum combinational depth is L_{max}. If the chosen segment length is 1 then the SDF based approach reduces to TDF model based testing, while choosing a length of L_{max} scaled to PDF model based testing. Even though, the segment delay model combines the advantages of TDF and PDF model, its major limitations are, the number of segment faults to be considered while targeting long paths that usually results in increased test pattern volume and higher test generation time. Even though the number of faults (while considering segment delay model) is less than the TDF based approach, however, it is still expensive due to the constraints on ATPG while fault sensitization and propagation. The segment based ATPG test generation is discussed further in section 2.1.4.

1.5 Yield and DPPM

So far we have discussed the timing defects originated due to spatial and temporal statistical variations in CMOS circuits along with motivation to test the circuit for manufacturing defects using structured approach. It is always expected that few manufactured ICs to be faulty due to defects incorporated during device fabrication, packaging and aging over the period under certain workload and environmental conditions. As discussed earlier, spatial variations in the circuit can be screened just after the chip fabrication while aging related variations can be observed during device operation over the time. Typically, stress test (burn-in) is performed to promote temporal variations much faster which results in electrical failures and helps in screening defective devices. The **yield** can be defined as the number of devices that qualify the test program which include functional and DFT related test patterns, from a set of devices under test. For example, assume that 900 devices passed the test program from a set of 1000 manufactured parts received from fabrication house, then the calculated yield is 90%.

The yield can be categorized as per users application and the manufacturing stage at which defects are observed on the device. Usually the chip manufacturer claims a **natural yield** of the packaged parts which are shipped to design houses. The chip

manufacturer does not use any dedicated test program (or automatic test equipment) to claim any desired test yield. However, they use undisclosed techniques to screen for imperfections present on silicon wafer such as pre-production wafer handling, active device patterning and metal routing issues etc. Based on the screening techniques, chip manufacturer provides natural yield number.

Once the packaged parts arrive at design house, they use dedicated test programs to test the packaged parts through ATE for functional or silicon defect related failures. Additionally, probe (or wafer) testing is also performed to ensure the health of wafers (without package), initial test program bring-up and enhance the maturity level of program for packaged parts. Once the test program matures, test engineers first claim **functional yield** and after circuit level repairing of few more devices, they claim **production yield**. The circuit level repairing may include logic as well as embedded memories, however, only memory repair is mostly adopted due to a well defined structural approach of testing them using **March algorithm**[42]. There is another terminology used which is called **parametric yield** typically defined at various bins at different frequencies.

Let us take an example to understand the definition of yield at different stages. Assume that the chip manufacturer fabricate 1200 dies on a single wafer. Out of 1200 wafers, 1100 dyes are screened to be healthy while 100 did not qualify the screening test. The natural yield, as defined by the chip manufacturer, would be 91.6% (1100/1200). Now these 1100 devices are shipped to design houses for further testing on ATE. Now further assume that the test program categorizes all device to be "good" if they pass a functional rated frequency of 800MHz. The test engineers found that out of 1100 parts, only 950 passed the test then functional yield is now 86.3%(950/1100). However, by using different repair techniques, they are able to revive 50 more parts such that the production yield get to 90.9%(1000/1100). Now, the parametric yield can be defined for three bins. BIN1 assumes a set of all devices which can work at 1200MHz, BIN2 and BIN3 assume to work at 1000MHz and 800MHz respectively. Assume that 300 devices qualified for BIN1, 500 devices qualified for BIN2 and 200 devices for BIN3, then the parametric yield at 1200MHz (BIN1), 1000Mhz (BIN2) and 800Mhz (BIN3) are 27.4% (300/1100),

45.4% (500/1100) and 18.1% (200/1100) respectively. Hence, the production yield is the summation of all parametric yield at different bins. For above example, parametric yield is only valid with an assumption that the customers are available to buy the devices at each bins and that is how a yield of 90.9% is calculated. DPPM (Defective Parts Per Million) is a measurement used by many engineers to measure quality performance. One DPPM means one (defect or event) in a million or 1/1,000,000. In the past a good chip maker supplier would have a defect rate of less than 1%, (10,000 DPPM). If a yield of 90.9% is obtained then it can be translate to DPPM as (100 - Yield(%))/100 which is 0.091.

1.6 Small Delay Defects

Small delay defects (SDD) can be defined as the distributed small timing defects in a circuit which add up enough propagation delay on one of the path such that device start showing erroneous response at rated functional frequency. Usually such type of defects escape untested as the traditional fault excitation and propagation techniques are timing unaware. It typically chooses easy to control paths for fault effect propagation and easy to observe for response capture. Timing tests that screen ICs for SDDs have been of interest for over a decade [1, 2, 4, 43, 44]. In order to target them, typically the fault effects are propagated through paths with least setup slack. This is being done as it guarantees the detection of small timing defects and improves defect coverage. SDDs are assumed to add a small localized delay to a switching transition propagated along the target path which may exceed the functional clock period. The primary sources of SDDs in earlier technologies are believed to be resistive vias, gate oxide breakdown, increase in interconnect resistance or capacitance etc. However, the introduction of FinFET transistors at the 22nm node by Intel, and most other foundries at 16/14nm, have given rise to a new type of physical defect in ICs - the *broken fin defect*. In a FinFET, the conducting channel is in the thin vertical fin which makes the device three dimensional (3D) as can be seen in Fig. 1.11.

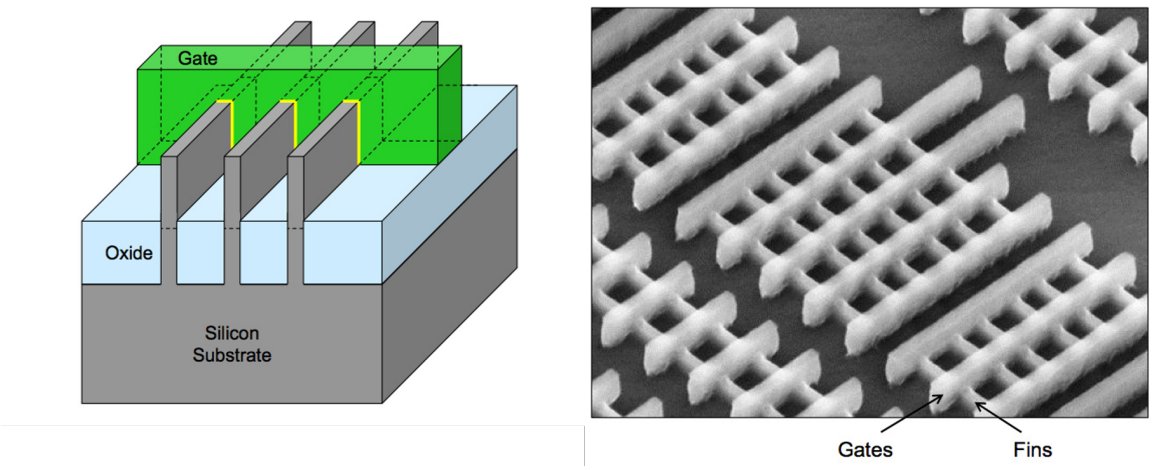


Figure 1.11: 3D Tri-Gate FinFET transistor can have multiple fins connected together to increase total drive strength for higher performance

Increasing the drive strength of a transistor (equivalent to increasing device width W_f of a traditional planar transistor) requires increasing the height (H_f) of the fin [4]. However, this is not very practical from a manufacturing perspective. For structural strength, FinFET transistors are fabricated with a fixed and relatively small fin height [45]. Multiple FinFETs of this fixed size are connected in parallel to achieve the desired drive strength of a gate [46]. The 3D structure of FinFET transistor can be seen in Fig. 1.11 that consists of three fins to increase drive strength. A single broken fin or under-grown fin does not generally result in catastrophic failure of the circuit. Only the gate drive strength is degraded, causing an increase in gate delay. For example, the loss of a single FinFET out of three parallel FinFETs driving the output of an inverter would increase the driving resistance and gate delay by approximately 50%. This would manifest as a small delay defect (SDD). Logic circuits designed in emerging gate all around (GAA) transistor technology, for 7nm and beyond, are also expected to display similar SDDs from transistor failures.

Now assume a circuit as shown in Fig. 1.12 with three primary inputs (IN1, IN2, and IN3) and two outputs (OUT1 and OUT2) respectively. Assuming one of the fin of OR gate, with orange color background, is not grown properly which resulted in 0.6ps increase in propagation delay across the gate at worst-corner. For sake of clarity, the

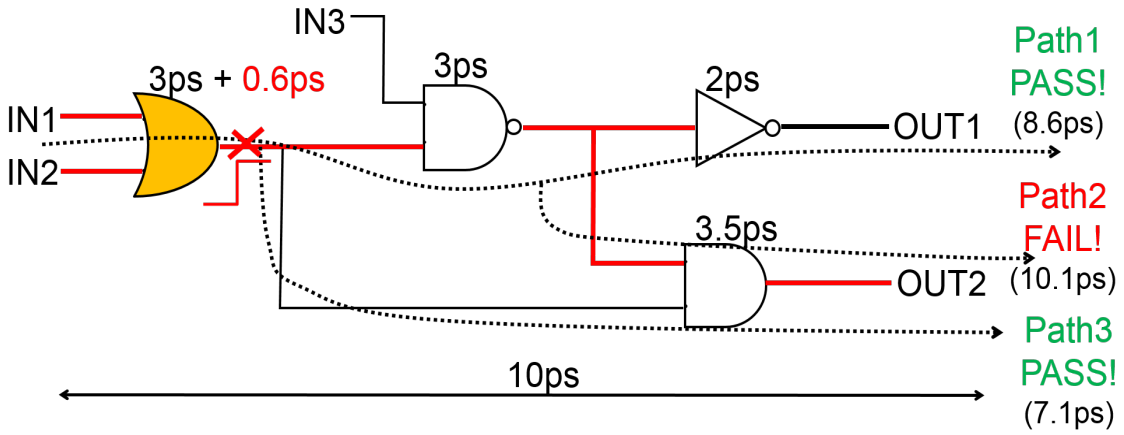


Figure 1.12: Small Delay Defect: logical view

small timing defect is not distributed and can be propagated along three possible paths (Path 1, Path 2, and Path 3) which are shown clearly in above figure. If the ATPG is able to excite and propagate the fault effect along least slack path *Path 2*, then only it guarantees the SDD detection during production test. If ATPG chooses *Path 1* or *Path 2* then the generated test patterns would never guarantee the defect detection at functional clock period of 10ps. This puts challenges to any of the test methodology because if valid small timing defects are untested then it may start limiting the circuit performance in customer applications. The customer device return typically results in increased defective parts per million (DPPM). In order to select the longest path through an identified delay fault, ATPG needs to be more intelligent by taking decision based on the timing delays of gates and interconnects. The timing-aware ATPG [47] has been introduced a decade back by commercial CAD tool vendors [48, 49] and it did reasonably good in targeting small timing defects. However, a higher SDD coverage is achieved at an expense of 10-15 times more pattern volume with relatively larger runtime.

1.7 Conventional Critical Path Identification Techniques

This subsection is dedicated to discuss a fact that the small-timing defects are true only if they fall along the performance-limiting paths that typically have least timing slack. There might be a case such that delay defects may disappear after a modest increase in functional clock period. On contrary, many defects start limiting the circuit performance after a modest decrease in functional clock period. An actual and realistic SDD coverage can only be claimed if the chip designers know a probable set of long paths which is going to limit the circuit functional operating frequency under spatial and temporal statistical variations. Usually, a set of critical paths are identified by two approaches: one is dynamic timing simulation [50, 51] and another is static timing analysis [52]. Even though, there are few more techniques like statistical static timing analysis [53] which represents the slack in terms of probability density function (PDF). PDF accounts for the variability of all process factors being modelled. Such kind of methodology targets a specific percentage yield for timing analysis and optimization. However, it is not yet adopted in practice because of the requirement of specialized timing library files and high tool runtime.

1.7.1 Dynamic timing simulation

Dynamic timing simulation or SPICE level simulation is very accurate in estimating timing delays. It uses statistical analysis that typically combines process deviations and device mismatch with circuit optimization capability. In order to capture the effects of interconnection's resistance and capacitance variations, it uses distributed RC network as obtained by RC extraction tools. However, the only drawbacks are, 1) they are input vector dependent and hence, 2) non-exhaustive and very slow, even for a subset of design. For complex designs, SPICE level simulation atleast requires subset of a circuit which include fan-in and fan-out logic cone of the target path, which needs to be analyzed for timing estimation in reasonable simulation time. Even if a complete circuit is provided to

the simulator, it would work only on a subset of a circuit and this is potentially a reason to call it non-exhaustive. Identifying a set of input vectors for bigger design is usually tedious and not well defined which typically consumes much more time in estimating paths delays of a circuit.

1.7.2 Static timing analysis

Since the last few decades, static timing analysis (STA) is widely accepted technique in validating the circuit timing such that the design can be signed-off to meet customer's timing specifications. STA reads,

- Gate level netlist created by synthesis tool
- Timing libraries that usually contain timing information in ASCII format of all standard cells, blocks and IO pad cells etc.
- Timing constraints and boundary conditions such as input/output delays, slew rates, false path, multi-cycle path etc.
- Parasitic data for more accurate timing analysis using a standard parasitic exchange format (SPEF) file [54].
- Physical data such as placement, floor plan, routing etc.

For large designs, STA is very fast in meeting timing specifications as it is vector-less, exhaustive in nature to cover all structurally possible paths of a circuit that guarantee circuit performance on a specified PVT corner. It works on two fundamental assumptions [55]:

1. *Single input switching assumption*: while performing timing estimation, it enables a signal transition (logic 0 to logic 1 or vice-a-versa) on the primary input of the target path with user-defined worst absolute transition time. The transition time can be defined as the time taken by the signal to rise from 10% to 90% or fall from 90% to 10% of the supply voltage. This assumption greatly reduces the complexity

and enhances the computational efficiency at the expense of adding pessimism in delay estimation.

2. *Least propagation algorithm*: while estimating multi-input gate delays, the input signal with latest arrival time would be selected to extract the propagation delay from the timing library. The propagation delay is measured from 50% of the input signal voltage swing to 50% of the output signal voltage swing across any specified gate or interconnect.

Due to such assumptions, typically STA provides more pessimistic timing delays as compared to dynamic timing simulation techniques such as SPICE level simulations [51]. The pessimism may also be added while extrapolating the delays (in the timing library files) due to out of range input slew (or transition time) or output capacitance [56]. The another source of pessimism is, "the parasitics data used for interconnect delay". The SPEF file used by STA for delay estimation typically contains a lower granular level details of resistance and capacitance values which are being captured for wires. In contrast, SPICE level simulations uses detailed standard parasitic format (DSPF) [54] which includes all resistance and capacitance distribution over the wire. In the next sub-section, we have compared the outcome of STA and SPICE on few benchmark circuits under controlled simulation environment. Based on the experiments and already published data [55, 56], it is fair to conclude that STA is much faster than dynamic simulation techniques for large circuits, however, at the expense of added pessimism in estimating path delays.

1.8 Motivation

Given the increasing concern over small delay defects originated due to spatial and temporal unreliabilities in CMOS circuits, as discussed in Subsection 1.1 and 1.2, there is a strong motivation for developing a new test methodology that must be viable, scalable, efficient and effective in comparison to conventional SDD detection techniques. As discussed in Subsection 1.4.2, transition delay fault based ATPG is ineffective in targeting

small delay defects as it does not have capability to choose longest structural path. On contrary, timing-aware (TA) ATPG uses circuit level timing information due to which, it is reasonably effective in targeting SDDs at an expense of increased pattern volume and ATPG runtime as discussed in Subsection 1.6. In nutshell, TA-ATPG [48, 49] may be a viable option, however, the major drawbacks are ATPG runtime and pattern volume which is 10-15 times more in comparison to traditional timing-unaware ATPG which makes it hard to adopt in practice.

We performed experiments on ITC'99, IWLS'2005 and ISCAS'89 benchmark circuits to evaluate the contribution of TA-ATPG in total pattern volume and ATPG runtime while targeting small delay defects. Now assume that a circuit has total k number of nodes that can be divided into two sets, 1) timing-critical nodes (lets say m) that fall along the long paths that must be target by TA-ATPG to claim SDD coverage, 2) non-timing critical nodes (lets say n) that must be targeted by conventional TDF based ATPG. In this work, any path whose absolute timing delay exceeds 80% of the total clock period is categorized as long path or critical path. It is a user-defined value that is chosen based on statistical data or circuit designer obligation.

Fig. 1.13 shows the percentage of total number of nodes either being categorized as timing critical or non-timing critical nodes. The figure shows the average data being captured from twelve different benchmark circuits. It can be seen that 37% of the total nodes are critical (shown under **Critical Nodes**) that must be targeted by TA-ATPG. Similarly, 63% of the total number of nodes are non-timing critical (shown under **Non-Critical Nodes**) that must be targeted by TDF based conventional ATPG. An approximately 7% of the total number of nodes that belong to critical category are not covered by TA-ATPG. Similarly, an approximately 12% of total number of nodes, belong to non-critical category, are not covered through traditional ATPG and hence, must be categorized as untestable faults. In order to cover 30% of the total number of nodes which belong to critical one, timing-aware ATPG on an average contributes 57.4% and 70% of overall pattern volume and ATPG runtime for a suite of benchmark circuits. If the TA-ATPG is expensive to cover just 30% of the total nodes then there is an opportunity

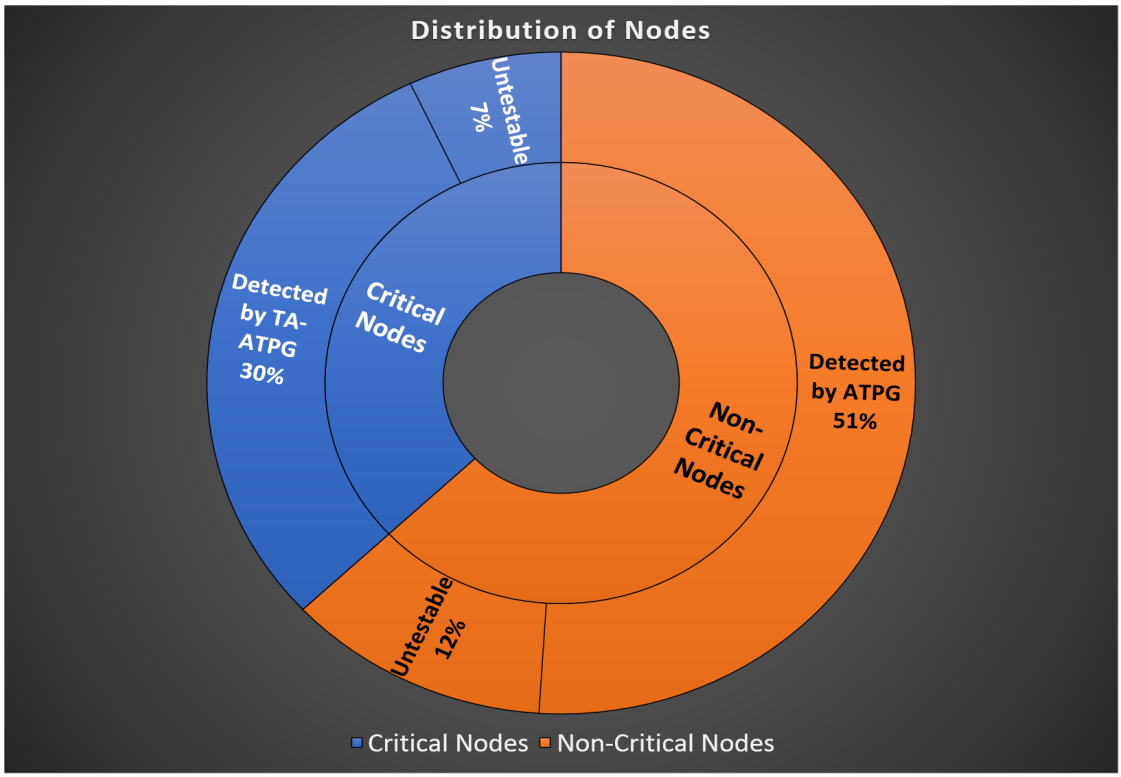


Figure 1.13: Percentage of circuit nodes as covered by timing-aware ATPG and conventional ATPG

to improve the effectiveness with better efficiency. This motivated us to explore a new test methodology that must be introduced to reduce the burden on TA-ATPG to cover timing-critical nodes.

The SDD coverage can only be reported for 37% of the total timing-critical nodes, which are isolated based on the circuit timing information read by TA-ATPG. Typically, the circuit level timing information is captured in standard delay format (SDF) file [57], IEEE standard for the representation and interpretation of timing data for use at any stage of an electronic design process, generated at specified PVT corner. The SDF files are generated by static timing analyzer (STA) that inherently adds pessimism while delay calculations as discussed in subsection 1.7.2. Lets take an example to pinpoint the issue with the delay estimations on a path as shown in Fig. 1.14.

The circuit has three primary inputs, one primary output and three gates. The

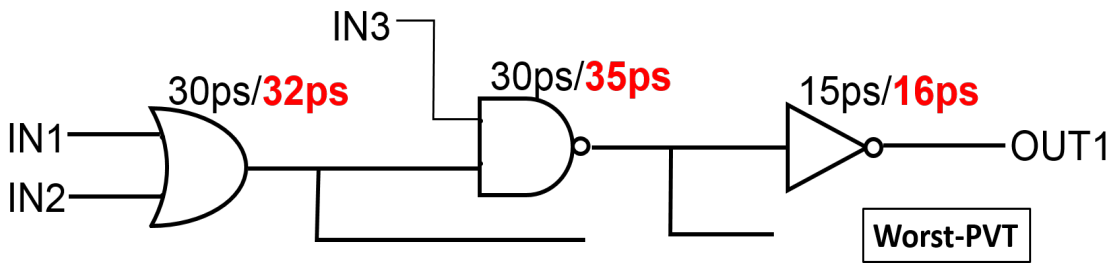


Figure 1.14: Path delay estimation by dynamic timing simulation and static timing analysis

path shown in the Fig. 1.14 is a part of a circuit synthesized to work at 100ps with an assumption that TA-ATPG would generate patterns for all those paths whose path delay exceeds 80ps (user-defined critical path delay threshold). The delay estimated by dynamic simulation (Fast SPICE) [51] and STA [58] are shown clearly in Fig. 1.14, where first absolute delay belongs to dynamic simulation result while second absolute delay (highlighted in red) belong to STA result. The worst-case path delay estimated by SPICE level simulation, PD^{sp}_{max}, would be 75ps (30ps + 30ps + 15ps) while path delay as estimated by STA, PD^{sta}_{max}, would be 83ps (32ps + 35ps + 16ps). Everything looks good until we realize that PD^{sta}_{max} is 1.5% more than PD^{sp}_{max} and user-defined critical path delay threshold value of 80ps would enable this path to be included in timing-aware ATPG analysis. A path which must be non-critical as pointed by dynamic timing simulation, is considered as critical by STA due to its inherent pessimistic nature.

To further validate the pessimistic behavior of STA during path delay estimation, we performed experiments on various benchmark circuits, as shown in Fig. 1.15. The benchmark circuits shown in the figure are synthesized and timed at 1000ps with user-defined critical path delay threshold limit as 800ps. A set of timing-critical paths, as identified by STA, is compared against the dynamic timing simulation results for the same value of user-defined critical-path delay threshold value on each benchmark circuit. The X-axis shows the various benchmark circuits while Y-axis denotes the percentage of total number of paths which are categorized as timing critical (delay greater than or equal to 800ps) on a specified circuit. It can be observed that for the circuit b15_1,

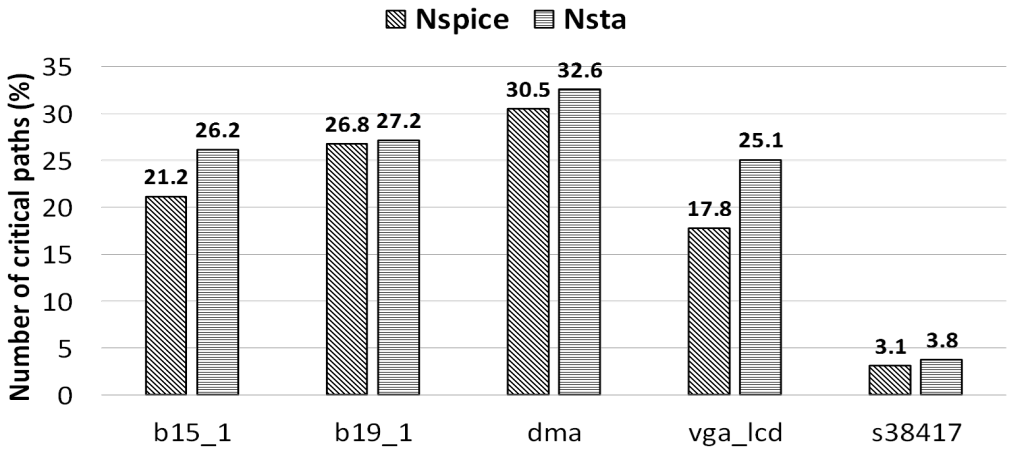


Figure 1.15: Critical path identification by dynamic timing simulation and static timing analysis

21.2% of total number of paths are isolated as timing critical by SPICE level simulation (*Nspice*) while STA identifies 26.2% of the total number of paths to be critical (*Nsta*). Typically we observe a 20-25% more paths are being categorized as critical by STA in comparison to dynamic timing simulation. These extra paths which are not worthy to be included for ATPG analysis must be removed. *This motivates us to explore a new timing estimation approach that should be simple and fast in compare to SPICE level simulation, with improved accuracy in comparison to STA estimated delay.*

In order to reduce the additional timing-critical paths being added by pessimistic STA, we need to identify a new approach that must include the strength of each technique, dynamic timing simulation (in terms of path delay estimation accuracy) and static timing analysis (in terms of fast runtime). This will certainly help in achieving objective of improving the overall effectiveness and efficiency of our test methodology and discussed further in Chapter 3. Fig. 1.16 shows the requirements of new path delay estimation approach. The shortcomings of the conventional techniques motivate to define a new path delay estimation approach whose accuracy should be more than STA based techniques, and must be substantially faster than dynamic timing simulation techniques. By accomplishing the above two objective, we would be able to reduce the number of

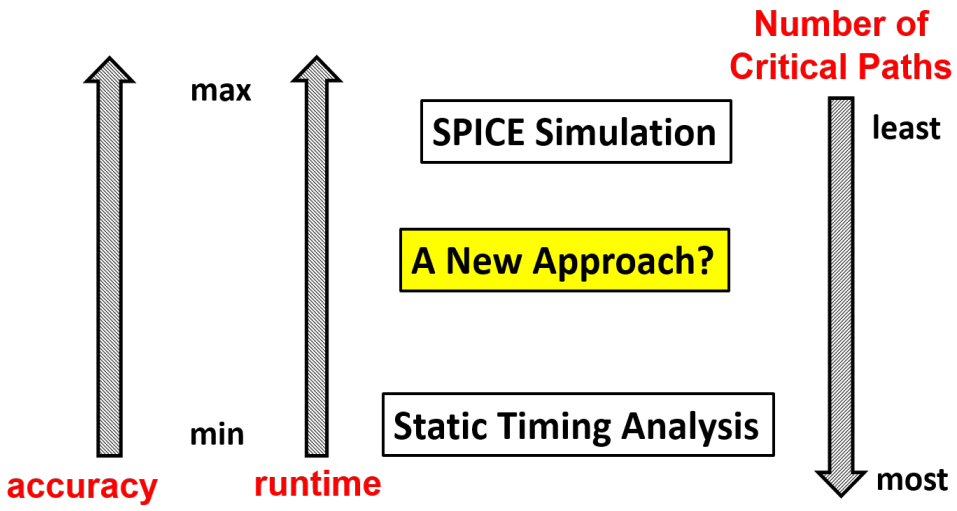


Figure 1.16: Defining the requirements of new timing estimation approach

additional timing-critical paths being added by STA. It is worth to mention that our new path delay estimation approach should not be used as standalone design sign-off timing analysis method.

1.9 Summary of The Thesis

This thesis, in general, addresses the shortcomings of conventional approaches to target SDDs. The thesis is broadly divided into two parts. First part discusses a new test methodology to cover SDDs more efficiently to reduce pattern volume and ATPG runtime as compared to traditional techniques. Our proposed test methodology for the first-time exploits path delay fault (PDF) test generation to not only generate the timing tests more efficiently, but the resulting TDF test sets are also more compact and perform better on commonly used delay test coverage metrics. This is because all TDF faults along a PDF targeted timing-critical path can be detected efficiently by generating a single PDF test. This efficiency is not explicitly exploited by node oriented TDF test generation even when the TDFs are targeted along the longest paths. We demonstrate the effectiveness of our methodology for a range of benchmark circuits by comparing the results from a commercially available TA-ATPG with our new approach that efficiently exploit PDF

tests wherever possible. Our proposed approach results in approximately 12.5% reduction in pattern volume, 35% reduction in ATPG runtime and also a 5% improvement in delay test coverage (DTC) when compared to existing TA-ATPG approaches. We also modified the conventional SDD metric definition to improve confidence of covering small timing defects through critical paths.

To uncover small timing defects of a circuit, it is important to identify a set of performance limiting paths which closely matches silicon behavior. In the second part of the thesis, we address the issues in identifying real timing critical paths that are the potential candidates for SDD testing. We discuss a new worst-case timing estimation algorithm that is reasonably accurate and substantially faster than commercially available SPICE level simulators. By using this new approach, we show that an approximately 50% of the unwanted additional paths added by pessimistic STA can be removed. Furthermore, nearly 94% accuracy in worst-case path delay estimation, as compared to dynamic simulations, of critical paths is observed on various benchmark circuits such as ITC'99, IWLS'2005 and ISCAS'89. We also validated the algorithmic complexity and its performance on various benchmark circuits and found that our approach is approximately 10-15 times faster than SPICE level simulations.

In order to explain the effectiveness of adopted test methodology which is used to generate a compact set of ATPG patterns in lesser time as compared to TA-ATPG, Fig. 1.17 is included in contrast to Fig. 1.13. It is mentioned in the previous Sub-section that the TA-ATPG on an average contributes 57.4% and 70% in overall pattern volume and ATPG runtime to cover 30% of the total number of nodes which are critical in various benchmark circuits. In earlier approach, all 30% of the total number of nodes are covered by TA-ATPG that seems to be very expensive in pattern volume and runtime. To improve the effectiveness and efficacy, we first aim to target a set of timing-critical nodes using path-based ATPG. As shown in Fig. 1.17, even if a modest 10% of the total number of critical nodes are covered by path delay fault based ATPG then this would result in 12.5% and 35% reduction in pattern volume and ATPG runtime respectively, which is further discussed in Chapter 3.

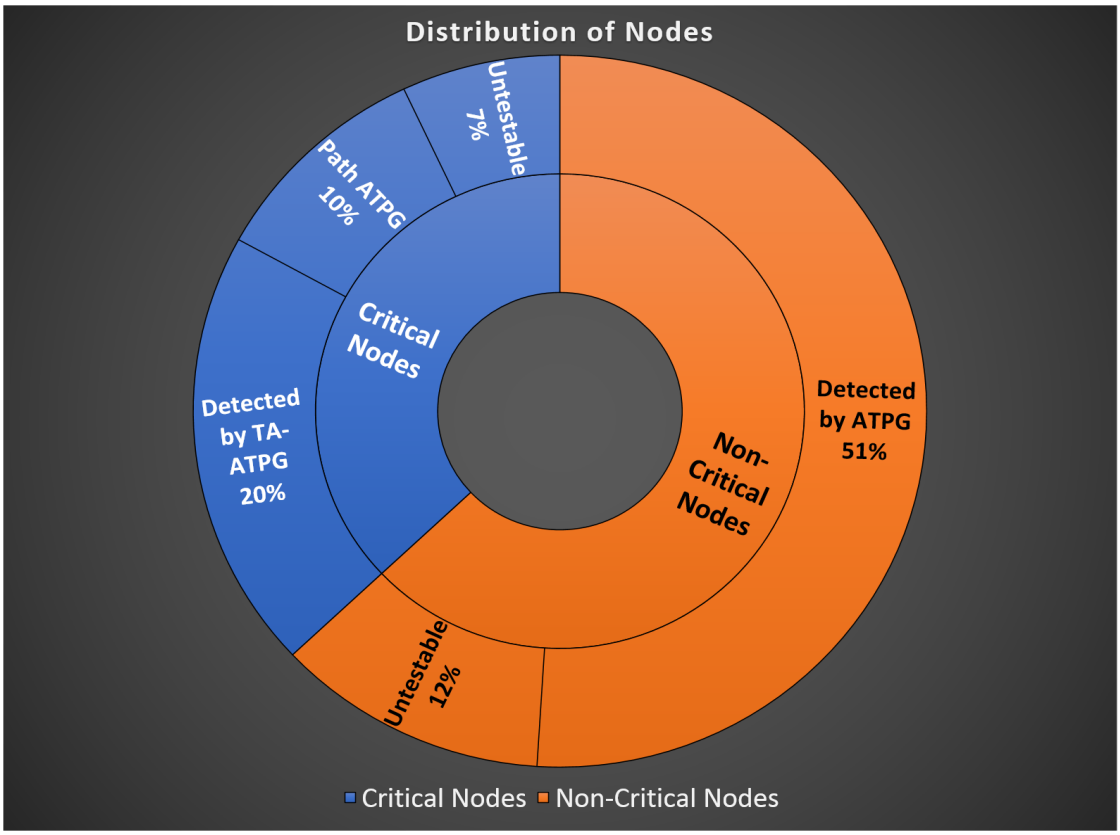


Figure 1.17: Percentage of nodes as covered by timing-aware ATPG, Path-based ATPG and conventional ATPG

Due to the pessimistic nature of STA, it adds many additional timing-critical paths while writing standard delay format (SDF) file which is used directly by TA-ATPG during pattern generation. In order to facilitate the understanding of core objective, we need to understand the distribution curve as shown in Fig. 1.18. Y-axis shows the fractional number of chips which include the same path fabricated on different dies whose arrival time (path timing delay) is shown on X-axis (typically follows Gaussian distribution). PD^{sta}_{min} and PD^{sta}_{max} are the best and worst timing delays as estimated by static timing analysis for a specified path. Similarly, PD^{sp}_{min} and PD^{sp}_{max} are the best and worst timing delays as estimated by SPICE level simulation under best PVT (PVT_{best}) and worst PVT (PVT_{worst}) for the same path as mentioned earlier. The difference between PD^{sta}_{max} and PD^{sp}_{max} shows the delay estimation gap between STA and dynamic timing

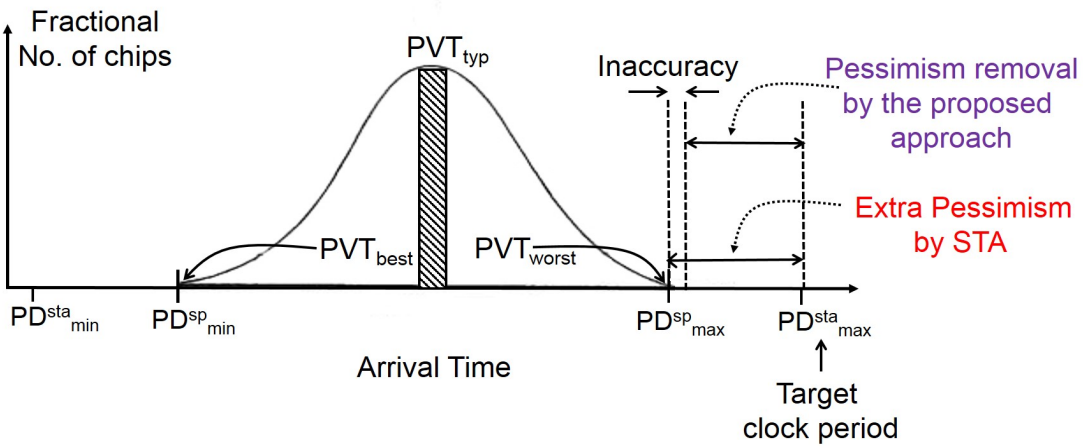


Figure 1.18: Path delay distribution over number of chips

simulation. Our yet another objective is to reduce the pessimism added by STA, provide a reasonable accuracy with respect to SPICE simulator, and must be substantially faster than dynamic timing simulation.

In this work, we explore an opportunity to use a new worst-case timing estimation approach to identify a set of realistic long paths that may become critical under spatial and temporal variations, with an aim to effectively and efficiently target the small timing defects in a circuit using a new test methodology. This thesis makes the following contributions,

1. Proposes a new test methodology [4] that for the first time exploits path delay fault based test generation for the timing-critical paths to generate timing tests for many of the targeted timing-aware transition delay faults. We demonstrate the effectiveness of our methodology for a range of benchmark circuits by comparing the results obtained from commercially available timing-aware ATPG [48, 49] with our new approach that efficiently exploit PDF tests wherever possible.

2. Adopting new test methodology resulted in approximately 12.5% reduction in pattern volume, 35% reduction in ATPG runtime and also a 5% improvement in

delay test coverage when compared to existing commercially available TA-ATPG approaches.

3. Proposes a new path based heuristic approach [3, 5, 59] that uses abstract level timing models to estimate worst-case delay of the target path under intra-die variations and aging effects. The path delays as estimated by the proposed algorithm have shown strong correlation with delays obtained using dynamic timing simulations on worst-case PVT corner.

4. The new path delay estimation algorithm is 10-15 times faster than the SPICE level simulation, with an approximately 94% accuracy in estimating timing delays at worst-case PVT corner.

5. Adopting the proposed path delay estimation technique reduces the number of extra timing critical paths, added due to pessimistic behavior of STA, by 50%.

1.10 Organization of The Thesis

It is worth to point-out that *our aim is not to replace industry standard multi-corner STA analysis to sign-off the circuit database for fabrication.* Instead we intend to identify a set of parameters that have major impact on the circuit performance and use them intelligently to identify a set of long paths that may become critical over time. These identified paths can be used for: *a)* running ad-hoc test to improve test quality after production, *b)* periodically monitoring the aging effects by functional testing of the paths, *c)* running dedicated scan-based test using Logic BIST every time the chip boots. The basis of using SPICE simulations as benchmark lies in the fact that the foundries provide SPICE models to accurately simulate the spatial and temporal unreliabilities in CMOS integrated circuits. Chapter 2 discusses the traditional fault detection approaches, their corresponding advantages and disadvantages respectively. The chapter also includes the closest prior test methodologies published in literature and subsequently discuses the enhancements required while targeting SDDs. Chapter 3 elaborates the procedure used

to develop the new test methodology to target small timing defects. It also discusses a step-by-step procedure to adopt our proposed methodology and further supported by experimental results obtained from a suite of benchmark circuits. Chapter 4 and Chapter 5 discuss the worst-case path delay estimation algorithm under spatial and temporal variations in CMOS circuit respectively. It also discusses a worst-case path delay estimation algorithm whose efficiency and performance is compared against the conventional approaches. Chapter 6 concludes the thesis with future work to improve the effectiveness and efficiency of the proposed test methodology.

— * — * —

Chapter 2

Previous Work

This chapter briefly discusses the origin of various automatic test pattern generators to cover fault models discussed in Section 1.4. Now the objective is to understand the basics of ATPG pattern generation, delay fault oriented testing steps and how it can be improved to get an optimal conditions for effective small delay defect coverage with reduced ATPG runtime and pattern volume. At the end of this chapter, we outline two closest test methodologies available in literature to compare with our approach in this thesis.

2.1 Traditional Fault Detection Approach

This section discusses the traditional delay defect detection techniques with their advantages and disadvantages. In structured test approach which is the broad area targeted in this thesis, circuit designers uses ATPG techniques to generate input test vector sequences to target classified faults in the circuit. Typically ATPG works in three main phases: a) excite the target fault, b) propagate the fault effect to the observation point through an identified path, c) justify the values of off-inputs without causing a contradiction. The goal of ATPG is to generate a set of patterns in order to achieve a desired test coverage. Test coverage can be defined as the percentage of total number of faults being detected using the generated pattern set out of initial target fault list. ATPG

usually targets the faults in two main steps: 1) generating fault-oriented test patterns and, 2) performing fault simulation to determine a list of faults being detected using the generated test patterns. The state-of-the-art ATPG tools automate these two steps into a single operation while pattern generation. The ATPG flow typically results in patterns, you can then save with added tester-specific formatting that enables a ATE to load the pattern data into a chip's scan cells. In order to achieve this, the normal flip-flops are replaced by D-MUX based scanable flip-flops (discussed in Sub-section 1.3.2) to provide 100% observability of data input and 100% controllability of data output from a flip-flop. Next, these flip-flops are stitched into a single chain or multiple chains based on the number of pads available on the device to perform test. The stuck-at fault model

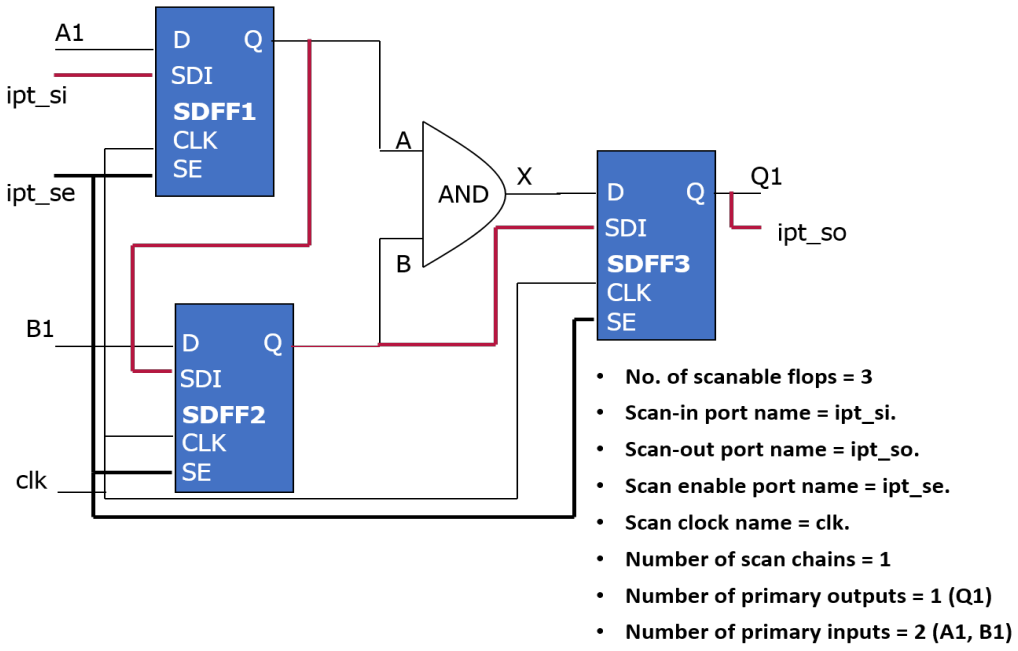


Figure 2.1: Scanable flip-flops in a chain

is, as discussed in Sub-section 1.4.1, assumed to be the most common fault model while performing fault simulation. It is being used because of its effectiveness in finding many common defect types. The stuck-at fault model captures the behavior that occurs if the terminals of a gate are either stuck at logic high (stuck-at-1) or logic low (stuck-at-0). Figure 1.5 and 1.6 of Sub-section 1.4.1 show the physical existence of stuck type defects

in silicon.

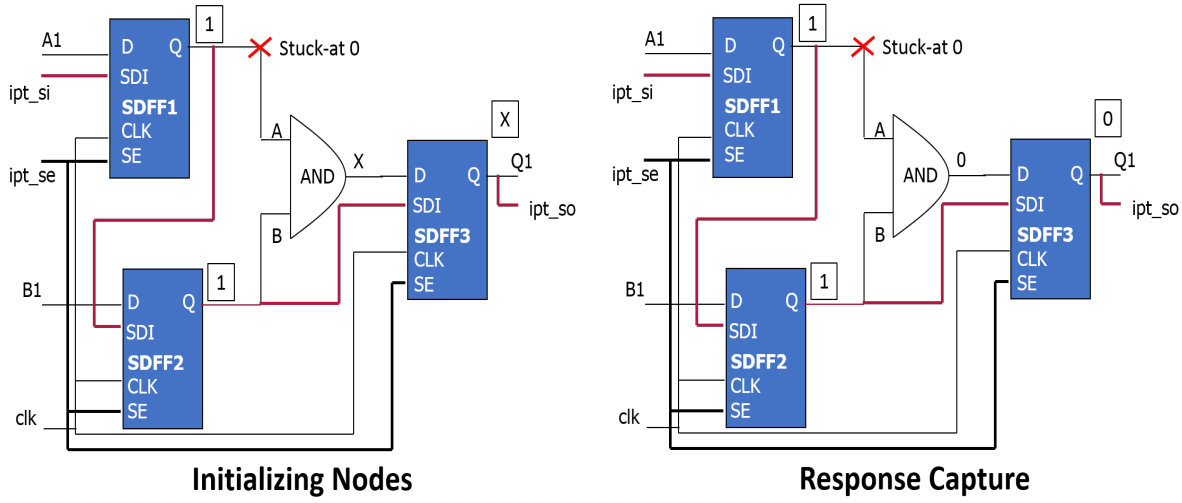


Figure 2.2: Circuit initialization during shift mode and response capture during functional mode

2.1.1 Stuck-at fault based ATPG

Assume a circuit shown in Fig. 2.1 which consists of three flip-flops, and one two input AND gate. The port name *ipt_si* and *ipt_so* correspond to scan input and output of the circuit which allow user-defined data of length 3 to be shifted whenever *ipt_se*, scan enable, is asserted. It can be seen that flip-flops are stitched into a single chain, shown in bold red connections, and assume that a stuck-at 0 defect exist after circuit fabrication. The fault site can be seen clearly in Fig. 2.2 and defect detection is divided into two parts, *Initializing Nodes* and *Response Capture*. Refer Fig. 2.3 to further understand the timing sequence which helps in explaining the circuit operation during scan mode. Once *SE* is asserted, fan-out from *ipt_se*, then all flip-flops would behave as serial-in serial-out register whose data shifting is controlled by pulsing shift clock *CLK*, fan-out from *clk*.

In order to initialize the stuck-at fault at A, a set of ATPG defined values are shifted-in so that they appear at output of flip-flops *SDFF1* and *SDFF2* whose values are shown inside rectangular boxes under *Initializing Nodes* in Fig. 2.2. Likewise, the shifted data

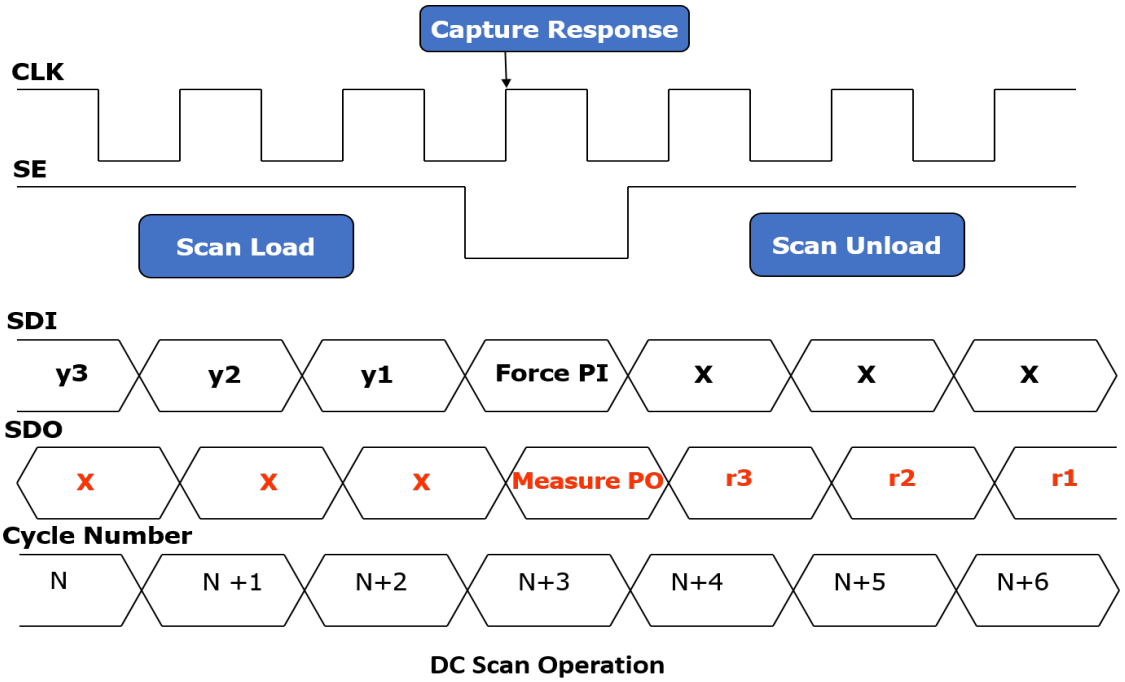


Figure 2.3: Scan timing sequence for stuck-at ATPG

y3, y2, y1 through the scan chain can be seen during *Scan Load* of Fig. 2.3. After data shifting, *SE* is de-asserted so that functional circuit response can be captured in the flip-flops after forcing primary inputs to some specified values. A capture pulse is issued from external clock driver to capture the response through data input of all flip-flops. After circuit responses are captured into the flip-flops during functional mode, scan enable *SE* is asserted again to shift the response data out which can be seen as R3, R2, R1 in Fig. 2.3. The cycle number can also be seen in timing sequence which clearly shows the number of external clocks issued by automatic test equipment. The stuck-at fault based ATPG is commercially known as DC Scan operation as it uses a single vector to statically initialize and propagate the fault effect to the observation point.

The main advantages of using stuck-at fault based ATPG are, 1) the number of faults increases linearly with increase in circuit size, 2) the state-of-the-art ATPG uses stuck-at fault test generation and simulation which can be easily modified for handling other at-speed delay defects, 3) diagnosis of defect locations in the circuit is reasonably well defined. However, as the circuit response capture is done statically, it is not capable of

targeting those defects which are timing sensitive.

2.1.2 Transition delay fault (TDF) based ATPG

In this sub-section, the focus would be on those defects which manifest themselves as delay defects and are sensitive to functional clock period of the circuit. The timing defects include gate drive strength variations due to random dopant fluctuations, resistive gate-oxide shorts, open or resistive vias, resistive bridges between the metal rails on same layer etc. The traditional TDF based ATPG [39] typically choses easy to control and easy to observe path with an assumption that extra delay caused due to a defect is large enough such that it can be observed at any of the circuit outputs. As it does not use circuit-level timing information while choosing the forward fault effect propagation path, therefore, it does not necessarily excite faults through timing-critical paths. As stuck-at fault CAD tools can be easily modified to target transition faults, it is the TDF based ATPG that is commonly used today to enhance stuck-at and transition fault tests in structural (scan based) testing of modern IC designs. In contrast to path delay fault based ATPG that covers distributed defects, which will be discussed in next sub-section, TDF tests target localized "lumped" delay faults at the circuit nodes as discussed in Fig. 1.8.

Defect that causes a node to become slow-to-rise or slow-to-fall are modeled as two separate and independently occurring faults by the TDF fault model. In order to achieve high coverage, TDF tests aim to cover all such faults in the circuit. Note that just as in stuck-at fault testing, it is the individual nodes in the circuit that are targeted by TDF tests. Therefore, the growth of TDF test set with increasing circuit size is similar to that for stuck-at tests. The size of the TDF test set is typically about 3-5 times the size of the stuck-at test sets. This is orders of magnitude smaller than PDF test set for a large circuit which makes TDF tests much more attractive. Also, distributed delays from random process variations, which are much better detected by PDF tests, are still mostly handled with timing margins. As TDF based ATPG conventionally choses short or intermediate paths for fault effect propagation, the generated ATPG patterns are

ineffective in targeting distributed small delay defects.

The TDF based ATPG also commercially known as AC scan which implies, testing gross-delay defects at functional speed. A transition delay test is being launched from primary input or data output from scanable flip-flop to sensitize delay defect and the fault effect being captured at primary output or data input of scanable flip-flop. The fundamental difference between the DC scan and AC scan is the addition of timing while launching and capturing the circuit response. Usually the transition delay tests are generated using two different methods: launch-on-capture (LOC) and launch-on-shift (LOS).

Launch-on-Capture (LOC) based delay test

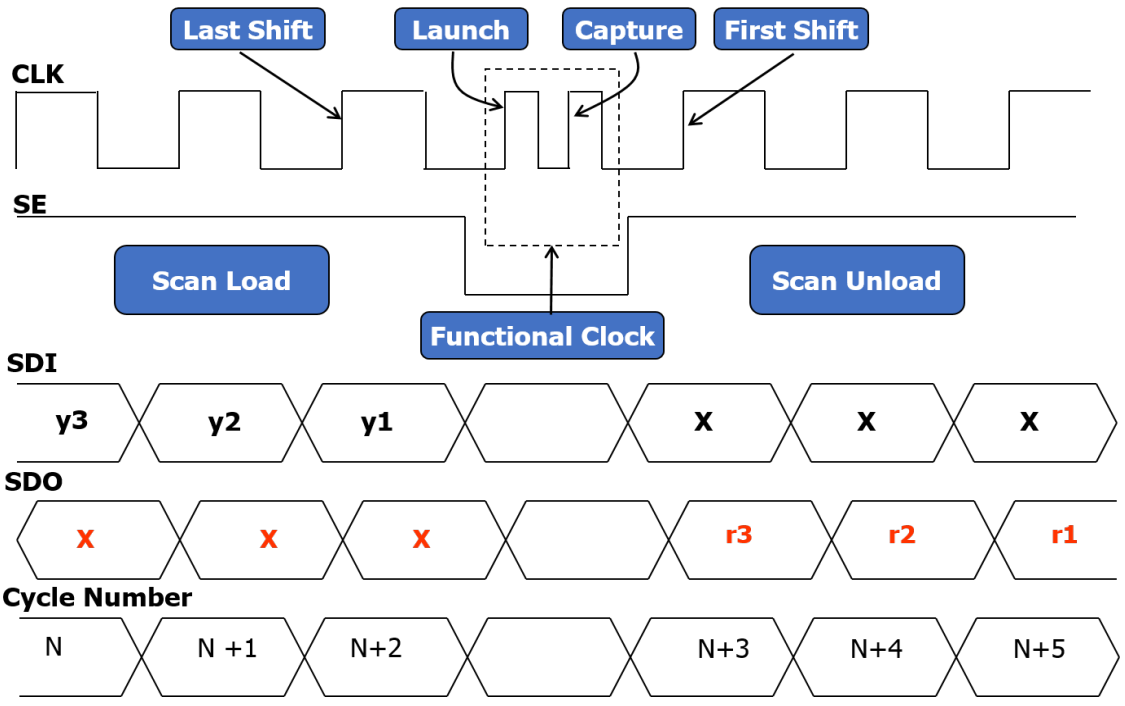


Figure 2.4: AC scan operation: Launch-on-capture test generation scheme

As we have discussed earlier, TDF based test requires a vector pair <V1,V2> where V1 is the vector which initializes the circuit and then V2 launches a transition that need to be captured at-speed. This implies that we eventually require functional at-speed clock cycles, one to launch the transition and second one to capture the circuit response.

The timing sequence of LOC can be explained with the help of Fig. 2.4 where it broadly sub-divided into *Scan load*, *Functional capture*, and *Scan Unload* respectively. Assuming a chain length of 3, the ATPG derived data would be scan-in in three cycles with data y3, y2, and y1 respectively. The last cycle *N+2* initializes the circuit and then, scan enable *SE* is de-asserted to allow data capture through functional paths. The first at-speed clock launches the transition whereas second clock captures the response at functional clock frequency. It must be observed that the launch value is derived from the preceding sequential depth associated combinational logic at that stage. It can also been seen in Fig. 2.4 that the cycle number of two capture cycles are not listed. This is being done to show that the capture pulses are not issued from an external ATE except in a case where shift frequency matches with circuit functional frequency.

Launch-on-Shift (LOS) based delay test

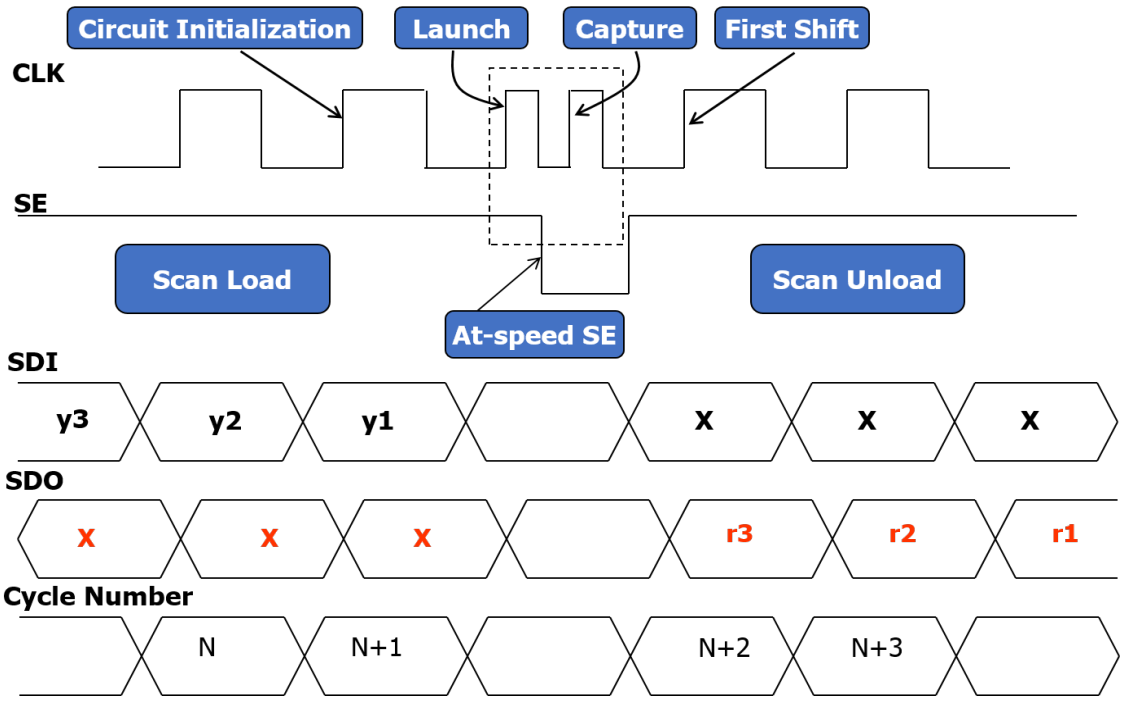


Figure 2.5: AC scan operation: Launch-on-shift test generation scheme

After discussing LOC based delay test generation, it is essential to take a look at launch-on-shift procedure which usually applied to fill TDF coverage gap in complex

circuits. Fig. 2.5 shows the timing sequence which is similar to Fig. 2.4 except that the scan enable *SE* is de-asserted at-speed between two functional clock cycles. The second last shift, which belongs to cycle number N+1, initializes the circuit and last shift launches a transition which needs to be captured at-speed on subsequent rated frequency clock pulse. The fundamental design issues of LOS in comparison to LOC are: a) scan enable must be treated as at-speed clock which certainly require more inverters and buffers to meet the timing requirements, b) more logic means more static as well as dynamic power dissipation during scan capture mode, c) high level of power consumption requires over-design of power rails. Now, lets take a look at ATPG outcome with respect to LOC and LOS test generation scheme. As the initialization and launch are done through second-last and last shift on flip-flop of scan chain, LOS is faster in runtime and easier for ATPG to generate compressed set of patterns to get maximum coverage. On the other side, LOC requires initialization through preceding sequential depth associated combinational logic. It generally takes longer time to generate timing tests and results in less pattern compression in the presence of on-chip compressor/decompressor logic. LOS is required to fulfill hazard-based detection conditions [60] that enhances the coverage of delay faults using the standard scan test application methods. In nutshell, LOC helps in generating quality patterns for delay testing at an expense of more runtime in comparison to LOS based testing. Whether we use LOC or LOS, the mainstream AC scan has an issues of not sensitizing and propagating the fault effect through longest paths. Until, the TDF based ATPG does not read the timing information of the circuit to choose the structurally longest path for fault effect propagation, it is not possible to target small delay defects.

2.1.3 Path-based ATPG

Instead of targeting gross-delay defects, path-based ATPG [61–63] targets predefined paths which usually consist of gates and interconnects, as discussed earlier in section 1.4.3 using Fig. 1.10. The generated ATPG patterns ensure that if timing defects, exist either on gates or interconnects, occur on the target path such that the arrival time

exceeds the clock period then it would result in erroneous circuit response. Tests for the path delay fault model can detect distributed timing defects caused by temporal and spatial variations in transistors and interconnects. However, there are several practical issues that have greatly limited the adoption of PDF testing [64–68]. Before the pros and cons of PDF based testing are further discussed, it is important to understand several classes of path delay fault model. A category of PDF based timing test is decided on how the path of a circuit is being sensitized: *robust, non-robust and functional sensitizable*. A detailed explanations of PDF categorization based on path sensitization criteria can be found in [37, 38, 69], however, this thesis defines an easiest way to classify path delay faults at circuit level. The next three sub-sections would help in isolating the PDF timing tests into designated category.

Robust test

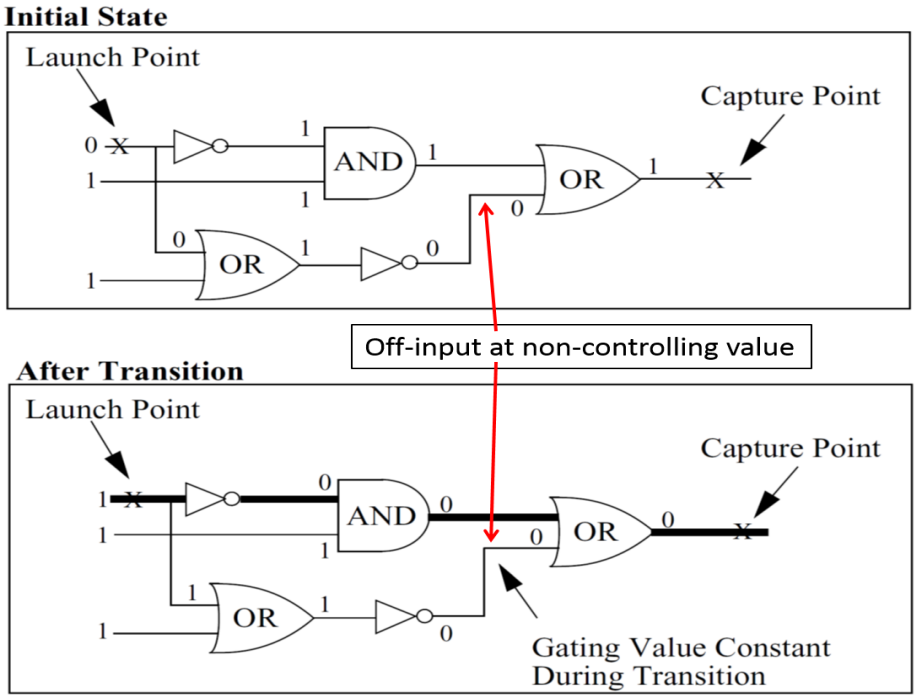


Figure 2.6: Robust test pattern generation

Now assume that the path shown in bold line is to be target by ATPG for slow-to-rise and slow-to-fall transition. Robust detection keeps the gating of the path constant during fault detection and thus, does not affect the path timing. Because it avoids any possible re-convergent timing effects, it is the most desirable type of PDF detection. For this reason, ATPG algorithm always first tries to generate robust patterns. However, ATPG cannot use robust detection on many paths because of its restrictive nature and once it is unable to create a robust test, will automatically try to create a non-robust test which is described in next sub-section. In nutshell, if the off-input paths remain at non-controlling value either during circuit initialization or during transition propagation then such kind of pattern can be categorized under robust test. This can be clearly seen in Fig. 2.6 that off-input of OR gate on target path remains at non-controlling value.

Non-robust test

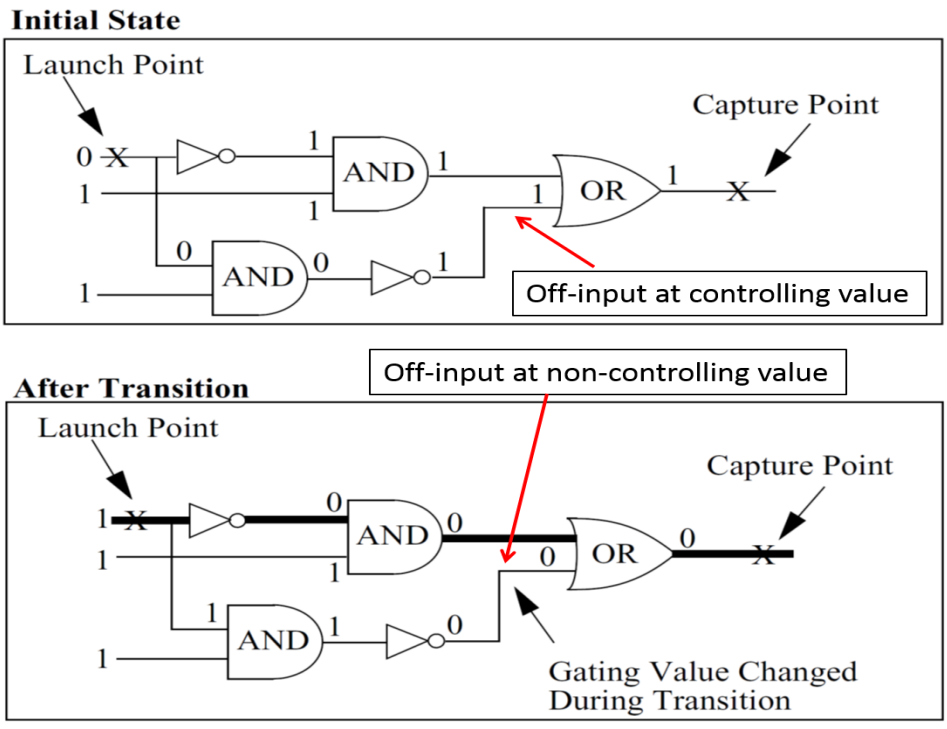


Figure 2.7: Non-robust test pattern generation

Non-robust detection does not require constant values on the gating inputs used to

sensitize the path. It only requires the proper gating values at the time of the capture event. Notice that due to the circuitry, in Fig. 2.7, the gating value on the OR gate changed from 1 to 0 during transition propagation. Thus, the proper gating value is required only at the OR gate during capture event. In short, if the off-input paths are at non-controlling values during signal transition (fault propagation) even the off-inputs earlier driving controlling values during initial state, then the pattern can be categorized under non-robust test.

Functional sensitizable test

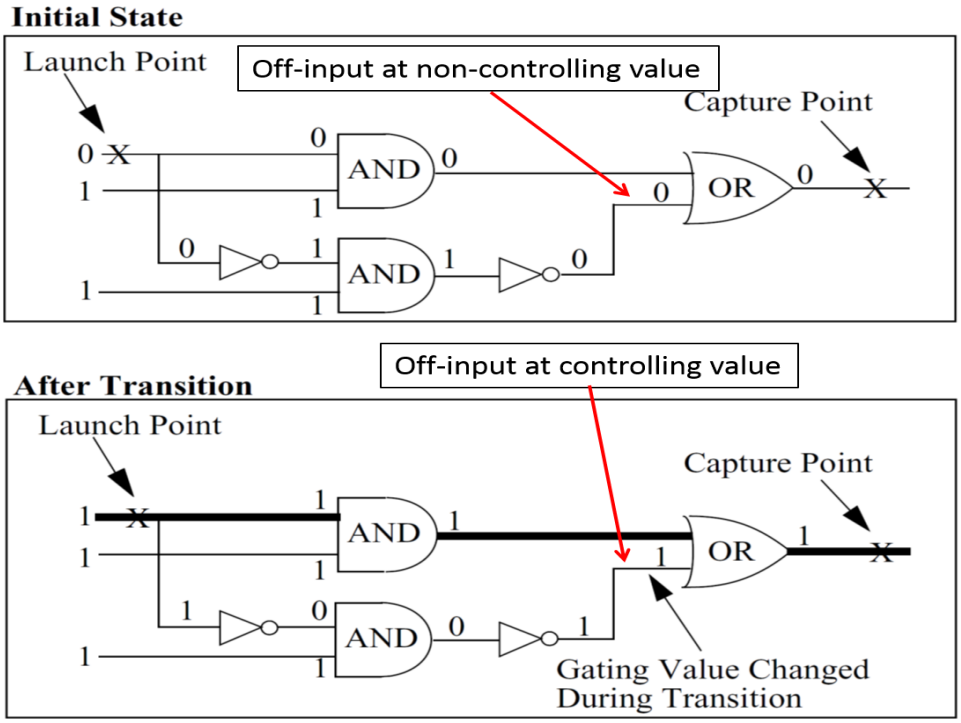


Figure 2.8: Functional sensitizable test pattern generation

Functional detection further relaxes the requirements on the gating inputs used to sensitize the target path. The gating of the path does not have to be stable as in robust detection, nor it have to be non-controlling at the capture event, as required by non-robust detection. Functional detection requires only that the gating inputs does not block propagation of a transition along the path. Fig. 2.8 gives an example of functional

detection for a rising-edge transition within a sample path. Notice that, due to the circuitry, the gating (off-path) value on the OR gate is neither stable, nor sensitizing at the time of the capture event. However, the path input transition still propagates to the path output.

The key problem with PDF based ATPG is the size of the test set, and resulting test application time, which grows dramatically faster with increase in circuit size compared to stuck-at tests. Also, if numerous random delays from process variability in a circuit are to be reliably tested then the PDF tests must be robust [70]. This means that the PDF test for a target path that is faulty (unacceptably slow) should not be potentially invalidated because of additional delays occurring in some other path that converges with the target path. It gives rise to an invalidating hazard at the observed output on the capture clock edge. Robust PDF tests ensure a reliable test for the target path irrespective of other delays in the circuit. Unfortunately, it has long been known that no robust PDF tests exist for many paths in a large circuit. Therefore, both robust [40] and non-robust [70] PDF test generation are widely supported by commercial ATPG tools [48, 49]. Due to above stated issues, path delay tests have been adopted in practice only to a very limited extent.

2.1.4 Segment delay fault based ATPG

It has been observed that most of the paths in a circuit are sensitizable neither through robust nor non-robust test generation criteria. A segment delay fault model tries to bridge the fundamental issues of testing a complete path by structurally choosing a segment of a path such that it can be covered by relaxing the robust or non-robust sensitization criteria. Segment delay fault model [41] is discussed earlier in Section 1.4.4 which clearly represents a better option than TDF and PDF fault models. Now assume a segment *e-f-g-h* as shown in Fig. 2.9 where the input cone and output cone represents the fan-in and fan-out cone respectively. The test pattern generation is divided into three phases: transition launching in **A**, fault excitation in **B** and fault effect propagation in **C** as shown in above figure. Transition launching requires the first node *e* of segment e-f-g-h

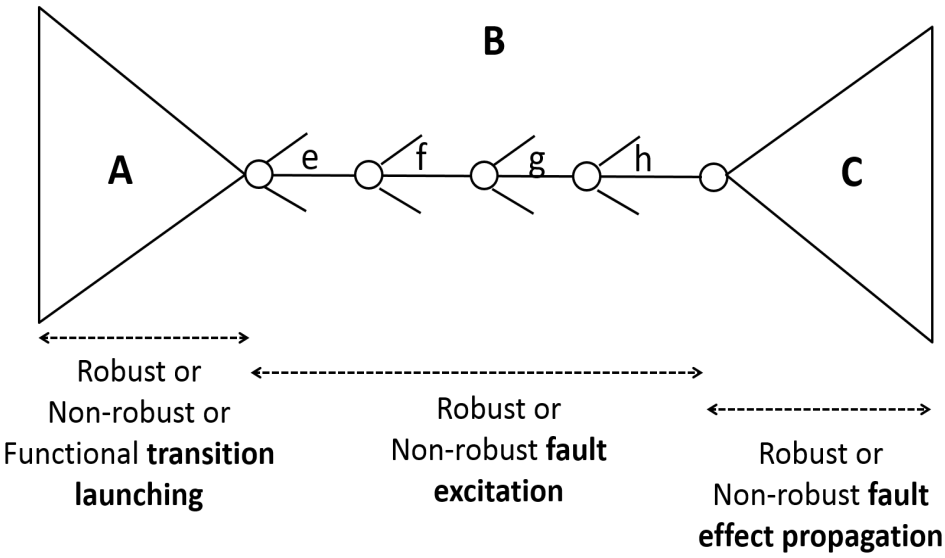


Figure 2.9: Test pattern generation for segment delay fault

to observe a transition with or without a hazard. This allows ATPG to choose robust or non-robust or functional sensitizable path from PI to e of part **A**. Fault excitation requires robust or non-robust transition propagation along the segment e-f-g-h while fault effect propagation chooses the same sensitization criteria as used to sensitize the target segment. ATPG can choose any path from h to PO of part **C** such that the propagation conditions are more relaxed than single sensitization criteria used for path delay fault based testing.

Choosing segment delay fault is potentially a good trade-off between TDF and PDF models. Instead of targeting gross-delay defects, it is much better way to club a number of gross-delay faults to make a segment and use various sensitization criteria similar to TDF based testing. In comparison to PDF based ATPG where path coverage is significantly low in most of the complex circuits, identifying various testable segments with varying length would potentially help in covering distributed timing defects. However, covering randomly occurring defects, of small sizes which fall on timing-critical paths, using segment based approach may miss many such small delay defects. The another problem is, segment based ATPG does not aware of the longest paths during transition launching and fault effect propagation while targeting segment fault. Hence, reasonable

number of SDDs can not be covered efficiently using segment based ATPG.

2.1.5 As late as possible transition fault (ALAPTF) based ATPG

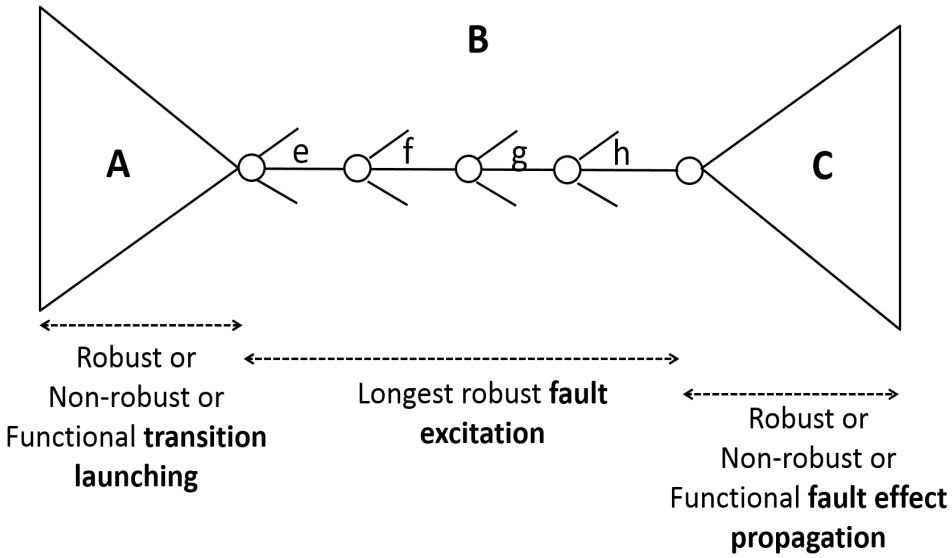


Figure 2.10: Test pattern generation for ALAPTF fault

In order to target segments through longest path during transition launching and fault effect propagation, a new *as late as possible transition fault* (ALAPTF) [71] is being introduced. Fig. 2.10 is similar to Fig. 2.9 except few changes are proposed to cover most of the distributed defects more effectively. It can be seen that transition observed by node *e* of segment *e-f-g-h* now can be sensitized using robust or non-robust or functional criteria. However, the path chosen by ATPG for transition launching must be longest so that if any small-timing defects exist then it must be covered by ALAPTF. In short, the objective is to delay the transition seen by first node of a target segment such that distributed timing defects accumulate over the path to show erroneous response. In this approach, instead of using robust or non-robust test criteria, only robust criteria is used for fault excitation. While the fault effect propagation uses all sensitization criteria to allow transition to be propagated along the longest path starting from fault site and

to be captured at PO.

ALAPTF model addresses the shortcomings of segment delay fault model by sensitizing longest path for transition launching at the fault site and chooses longest robustly testable segment to target small delay defects more effectively. In order to identify the longest path, ALAPTF approach uses unit level gate delays to structurally decide the longest path and that is potentially a reason that it is more effective than segment based approach. However, for nanometer technology nodes, identifying longest paths based of unit delay model is not justified as interconnect delays are getting dominant over gate delays for few paths. Hence, it can be concluded that ALAPTF may be better approach than using segment based delay testing, however, it has a major shortcoming of not sensitizing the real long paths for fault detection.

2.1.6 N-detect transition fault based ATPG

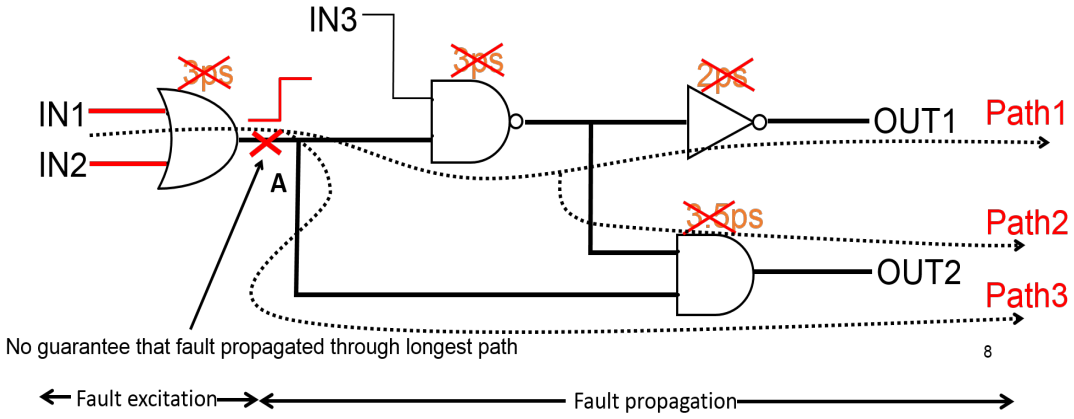


Figure 2.11: N-detect transition fault based ATPG: logical view

In conventional TDF based ATPG, if a fault is detected through a test pattern then it is necessarily dropped from the target fault list so that ATPG makes effort in targeting rest of the faults. However, in N-detect ATPG [72, 73], the fault is not dropped from the target fault list until it is being detected N times through different ATPG test sequences. In order to explain this at circuit level, we have used Fig. 2.11. The circuit consists of four gates, three primary inputs (IN1 to IN3) and two primary outputs (OUT1 and

OUT2). Now assume that node A has to be tested for slow-to-rise transition fault and ATPG is programed not to drop the fault until it is being detected 3 times. The fault at node A can be detected through three different paths namely Path1, Path2 and Path3 respectively. Even though the longest timing path is Path2, however, it is not guaranteed that ATPG would choose the longest timing path, out of three chances, to excite and propagate the fault effect. This is due to the fact that it does not uses circuit-level timing information while propagating the transition along the target path.

N-detect ATPG can be static in nature as studied in [72] where authors have a unique chip fallout of approximate 3.8% over basic stuck-at, bridge and functional tests on Intel Pentium 4 processors. However, static N-detect ATPG can not detect at-speed timing defects as transition launch and capture must be done at functional frequency. Later, authors of [73, 74] have discussed the effectiveness of N-detect transition delay fault based ATPG. Several major shortcomings of using N-detect ATPG are: a) its effort in detecting same fault N times which makes in expensive in ATPG runtime and pattern volume, b) no guarantee that the fault is sensitized through the longest testable path as it does not uses circuit-level timing information, and c) no small-delay defect (SDD) coverage metric to claim defect free circuit. In short, if a set of distributed small timing defects are need to be covered then it is essentially required to first identify real timing-critical paths. This can be done either using enhanced ATPG algorithm which can read circuit-level timing information or use same timing analysis approach to identify a set of long paths that can potentially fail under spatial or temporal variations on chip.

2.1.7 Timing-aware ATPG

To ensure detection of small delay defects, TDF test generation can be made "timing-aware" such that each target TDF fault is possibly propagated along the longest sensitizable path of a circuit. We have used the same circuit, of Fig. 2.11, to explain the test generation steps of timing-aware ATPG which now uses gate delays as shown in Fig. 2.12. For the sake of clarity, interconnect delays are ignored while path delay estimation. Assume that a node A needs to be tested for slow-to-rise fault and there are three paths

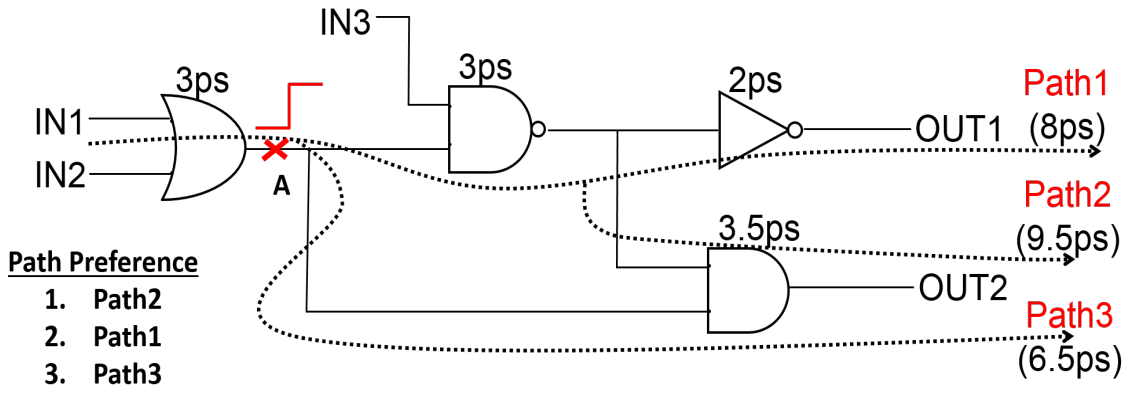


Figure 2.12: Timing Aware (TA) ATPG: logical view

available to ATPG for fault propagation namely Path1, Path2 and Path3 respectively. It can be seen that Path2 is longest with absolute delay of 9.5ps while rest of two paths Path1 and Path3 have delays of 8ps and 6.5ps respectively. TA-ATPG would first try to sensitize the fault through Path2 and in case it is unsuccessful, would then try to sensitize the fault through Path1 and so on until a user-defined fault dropping criteria is not reached. TA-ATPG maximizes the SDD detection chances, however, it significantly increases the test set size and test generation time because of ATPG's ability to use circuit-level timing information while sensitizing the longest path. The increase in test cost has been a limiting factor to the adoption of timing-aware TDF tests [47], although this is expected to change as more number of SDDs are encountered in FinFET technology.

So far we learn that timing-aware ATPG reads timing information from a Standard Delay Format (SDF) file and tries to generate patterns that detect transition faults using the longest sensitizable path. Now lets discuss few important terms that would influence the TA-ATPG's outcome. Slack is equal to the setup margin between the path delay or path arrival time and functional clock period. Slack of a target path represents the smallest delay defect that can be detected by applying timing-aware ATPG generated pattern. Fig. 2.13 illustrates slack calculations of various paths which are classified as *longest, medium and shortest*. Assume there are three paths that can detect a same

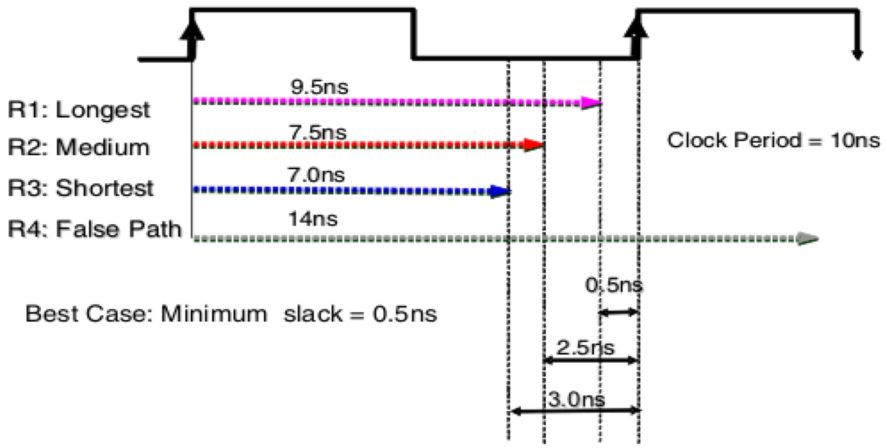


Figure 2.13: Timing slack of various paths (example only, not to scale at specified technology node

target fault. The paths are having 9.5ns, 7.5ns and 7.0ns delay, respectively. The clock period is 10ns and slacks for the paths are calculated as 0.5ns, 2.5ns and 3ns, respectively. For the longest path, which has a 0.5ns slack, the smallest delay defect can be detected is 0.5ns. Similarly for the path with 2.5ns, the smallest detectable delay defect is 2.5ns. You can detect smaller delay defects by using a path with least slack. Any path that is longer than the clock period is a false path which functionally does not require any timing constraints.

Delay test coverage

ATPG effectiveness is generally measured using defect coverage metric called Delay Test Coverage (DTC). It determines the quality of the test patterns in terms of ATPG's ability in sensitizing longer or longest path while fault detection. The delay test coverage is automatically included in the ATPG statistics report when timing-aware ATPG mode is enabled. DTC (in percentage) for a specified fault can be defined as,

$$Delay\ Test\ Coverage = \frac{Longest\ Testable\ Path\ Delay}{Structural\ Longest\ Path\ Delay} \times 100 \qquad (2.1)$$

Using the path example as discussed in Fig. 2.13, if ATPG uses path R2 then the delay test coverage is calculated as: 7.5ns / 9.5ns x 100 = 79%. If ATPG used the longest path (R1), then the delay test coverage would be 100%. Undetected faults have a delay test coverage of 0%. DI faults (Detected by Implication) have a delay test coverage of 100%. *Chip-level delay test coverage is calculated by averaging the delay test coverage values for all faults which are timing critical.*

Timing-critical transition faults

Timing critical faults are those transition faults which fall along timing critical paths or long paths. Unlike PDF based testing, a transition fault can tested along a longer path if not testable through longest path. This gives TA-ATPG an opportunity to find multiple long paths, based on the circuit-level timing information, to sensitize the target critical fault. As the timing information is available to ATPG, it can isolate a set of transition faults into two category, one is timing critical and another is non-timing critical based on the following condition,

$$\frac{T_{func} - PD_f^s}{T_{func}} < \lambda \tag{2.2}$$

PD_f^s is the longest path delay through a transition fault f and T_{func} is the functional clock period at which fault f need to be targeted at-speed. The user-defined parameter λ is a real number between 0 and 1. If any fault follows the condition mentioned in Eqn. 2.2 then it must be categorized as timing critical transition fault which is suppose to be targeted by timing-aware ATPG while rest of the faults would be covered by conventional ATPG. For example, assume there are two faults (F1 and F2) with static longest path delay of 8.5ns and 9.5ns respectively. The clock period is 10 ns, and if we specify $\lambda = 0.1$ then F2 would be selected (with calculated value of 0.05) and F1 would not (calculated value of 0.15). In this thesis, all TA-ATPG oriented experiments use a value of $\lambda = 0.2$, however, it is not hard-coded and user can pick any value based on the test pattern volume or test quality.

Fault dropping criterion

Once a set of timing-critical transition faults exists then TA-ATPG can be invoked to generate timing tests, however, it would not be straight forward. Now assume a case where a timing critical transition fault lies along four paths with path delay of 9.5ns, 8.5ns, 7.5ns and 6.5ns respectively. As we discussed earlier, TA-ATPG would first try to sensitize the fault through 9.5ns path and then rest of the paths sequentially. This would start limiting the TA-ATPG performance in terms of algorithm runtime and certainly we need to limit the TA-ATPG to drop the fault from the list if it goes beyond a user-defined fault dropping criteria. In dropping based on slack margin (DSM), a fault is being dropped and categorized as undetected if it satisfies the below condition:

$$\frac{T_a - T_{ms}}{T_a} > \delta \tag{2.3}$$

where T_a is the slack used by the ATPG for fault sensitization and T_{ms} is the minimum slack as estimated structurally with respect to clock period. δ is a user-defined real number between 0 and 1. For TA-ATPG, $\delta = 1$ is equivalent to traditional fault dropping criteria while $\delta = 0$ invoke deterministic test generator which covers faults sensitized only through longest paths. In this thesis, all TA-ATPG oriented experiments use a value of $\delta = 0.5$ which is suppose to be optimum as published and explained in [47, 75].

Timing-aware ATPG versus Traditional ATPG

In recent years, many work have been published [47, 75–77] which compares TA-ATPG and conventional ATPG results. In this thesis, an industrial design is chosen to compare the results in three dimensions: pattern volume, delay test coverage, and ATPG runtime. The circuit has following characteristics:

- Deign size: 2.67 million gates
- Number of flip-flops: 75,243
- Functional clock frequency: 1.8GHz

Table 2.1: Testcase to compare TA-ATPG and Traditional ATPG results

ATPG Runs	Pattern Count	Transition Coverage	Delay Test Coverage	CPU Runtime	Memory Requirement
Trad. ATPG (1 detect)	12079	98.17	55.67	9889	20GB
Trad. ATPG (7 detect)	24987	98.17	60.34	27013	20GB
TA ATPG ($\delta = 1$)	14508	98.16	75.34	37345	80GB
TA ATPG ($\delta = 0.5$)	35415	98.16	89.67	129735	110GB
TA ATPG ($\delta = 0$)	95767	98.17	92.31	220123	140GB

- Number of clock domains: 2 [both asynchronous clocks are generated using on-chip PLL, one is locking at 1.8GHz and another is locking at 600MHz]
- 402 seconds to read the SDF file that has 12,678,213 lines, generated at worst PVT corner

Table 2.1 compares the timing-aware ATPG and conventional ATPG results on the circuit whose characteristics mentioned above. We choose to run five different ATPG runs which are shown in column 1 of the above table. The first two runs invoke traditional ATPG with 1 and 7 transition fault detection mode respectively. The last three runs invoke timing-aware ATPG with DSM criteria, δ, of 1, 0.5 and 0 respectively with timing-critical transition delay fault parameter $\lambda = 0.2$. For ease of use, lets call row 2 to row 6 of column 1 as ATPG1 to ATPG5 respectively. In the experiment, we ensure that whether we use traditional or TA-ATPG approach, the transition fault detection coverage should remain the same which can be seen in column 3. The delay test coverage is also reported for traditional ATPG approaches and can be observe that even the conventional technique covers the timing-critical transition faults through mid-size paths. The N-detect ATPG (ATPG2) additionally improves the DTC by 4.7% by generating approximate 2 times more patterns and consuming 2.7 times the ATPG runtime of conventional ATPG. The ATPG3 which uses conventional fault dropping criteria increases the DTC by 19.7% with merely 1.2 times the pattern volume of ATPG1, however, runtime blows to 3.7

times which is expensive. Similarly, ATPG4 and ATPG5 get even more expensive in pattern volume (which is 3 times and 8 times more than ATPG1) and ATPG runtime with reasonably good SDD coverage. In short, *timing-aware ATPG is expensive even if a limited set of transition faults are targeted through it.* Below is the list of timing-aware ATPG limitations,

1. Circuit-level timing information is parsed through SDF [57] files which is typically generated using static timing analyzer [52, 55, 58]. Due to the pessimistic behavior of STA while estimating worst-case path delays, it inherits into ATPG analysis as well. The additional timing-critical paths added due to pessimistic behavior severely impacts the ATPG analysis in terms of pattern volume, runtime, and sometime SDD coverage as well.

2. The transition test coverage of TA-ATPG may be lower than traditional ATPG. This is because timing-aware ATPG tries to detect a fault with a longer path, and it is more likely to hit the abort limit before being detected.

3. ATPG run time for timing-aware is about eight times slower [48, 49] than the normal transition fault ATPG. However, it depends on the circuit complexity which may gets even worse.

4. A large combinational loop may slow down the analysis to calculate the static slack. It also makes the actual delay analysis less accurate.

5. False and multi-cycle paths are defined by synopsis design constraint (SDC) files. SDC usually adds more constraints to ATPG while pattern generation which start impacting ATPG runtime. The slack for false and multi-cycle paths is rounded to zero, and the delay test coverage is 0% as well.

6. If the compressor and decompressor hardware logic are used on chip then TA-ATPG slows down the pattern re-ordering which is optimized for test coverage.

2.2 Small Delay Defect Detection Methodology

So far we have discussed various standalone approaches of targeting small delay defects in the circuit. This section includes test techniques which are close to our proposed test methodology and need to be examined for pros and cons. Once the prior published methodologies [1, 2] are covered in next two sub-sections, we will conclude this section to re-define the objective of SDD detection more effectively with enhanced efficiency in pattern generation. The problem of targeting SDDs can be solved by either using standalone circuit-level timing information first and then use the extracted intermediate information to guide traditional ATPG for pattern generation or using timing-aware ATPG techniques which are fully supported by the majority of CAD tool vendors.

2.2.1 Prior Test Methodology 1 [1]

Methodology 1 is proposed in 2006 by Ahmed et al. [1] which presented a technique of pre-processing slack information associated with each flip-flop to generate cell constraints. By doing this, it can guide existing timing-unaware ATPG tool to behave as slack-based ATPG tool. A complete flow diagram of *methodology 1* is shown in Fig. 2.14 which broadly covered in five steps, namely step1 to step5. It begins with using commercially available STA tool which helps in identifying top timing-critical paths and also identifies a set of flip-flops which act as an observation point or end-points for these critical paths. Obviously, there would be a list of overlapping flops which can also behave as end-points for intermediate or short paths as well. In order to avoid intermediate paths and short paths under *step1*, *Tool 1* is utilized to generate static cell constraints during shift in such a way that ATPG tool never excite the fault through intermediate or short paths. *Tool 1* not only helps in generating ATPG cell constraints, however, it also identifies the list of timing-critical transition faults and corresponding path delay faults (equivalent to long paths identified by STA).

In *step 2*, the PDF faults are targeted by path-based ATPG and it is worth to remind that it would not result in significant path coverage as discussed earlier in Sub-sections

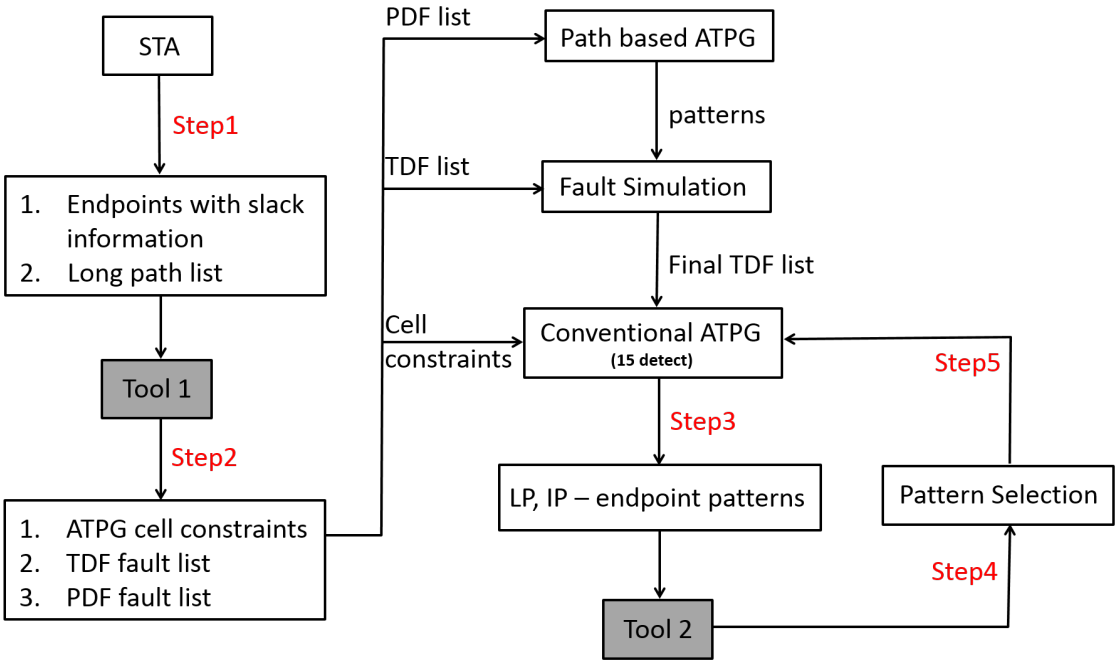


Figure 2.14: Automation flow of methodology 1 [1]

1.4.3 and 2.1.3. The generated path-based ATPG patterns are fault simulated with TDF faults as isolated by *Tool 1*. A final list of undetected TDF faults are exposed to timing-unaware ATPG enabled with 15 detect to generate patterns. It must be remembered, as discussed earlier in Sub-section 2.1.6, that 15-detect ATPG usually results in 3-4 times more pattern volume as compared to traditional ATPG approach. It also results in few more paths being exercised through intermediate paths which must be screened out. *Step 3* calls for an another tool, *Tool 2* which can perform the path delay distribution analysis, followed by masking end-points which sensitize the intermediate paths. *Tool 2* helps in pattern re-ordering to maximize fault coverage and fault grade with a set of patterns which maximizes the SDD coverage. Below is the list of major shortcomings that can be considered to make it more effective and efficient,

1. Generating least slack information per scanable flip-flop is overhead for STA tool, additionally, generating static cell constraints during shift can only be achieved effectively if the circuit size is small.

2. Adding cell constraints to ATPG in order to avoid short or intermediate paths usually results in increased ATPG runtime and pattern volume with lower test coverage.

3. Pattern re-ordering technique for smaller circuits which do not use on-chip compressor/decompressor logic usually works well. However, with many cell constraints in the presence of on-chip compressor/decompressor logic results in lower test coverage. This certainly results in increased overall runtime during pattern re-ordering or fault-grading.

2.2.2 Prior Test Methodology 2 [2]

Methodology 2 is proposed in 2009 by Goel et al. [2] for generating timing-aware transition fault patterns that target small delay defects. It uses timing-aware ATPG to cover a subset of transition faults while rest of them would be targeted using traditional ATPG approach. Fig. 2.15 shows an outline of *Approach 1* and *Approach 2* dedicated to improve SDD coverage and reduce pattern volume respectively. In *Approach 1*, the conventional transition fault based ATPG patterns are generated and then ATPG mode is changed to "timing-aware fault simulation mode". As discussed in Sub-section 2.1.7, user defined parameters (λ and δ) are suitably defined before using fault grading technique with existing set of traditional ATPG patterns. After fault simulation, undetected timing-critical transition faults are isolated and TA-ATPG is invoked to generate high quality test patterns by using DSM criteria, δ, close to 0. Now, the top-off SDD patterns must be used along with traditional TDF pattern sets to claim TDF coverage and delay test coverage. This approach typically results in improved DTC coverage at an expense of additional patterns added by TA-ATPG which is categorized as *top-off SDD patterns*. One of the problem with this approach is, a specified timing critical transition fault may be multiply detected by conventional ATPG pattern through shorter path first and then the TA-ATPG generated SDD pattern through longer(or longest) path. This results in additional patterns which are really not required and generated SDD patterns can be

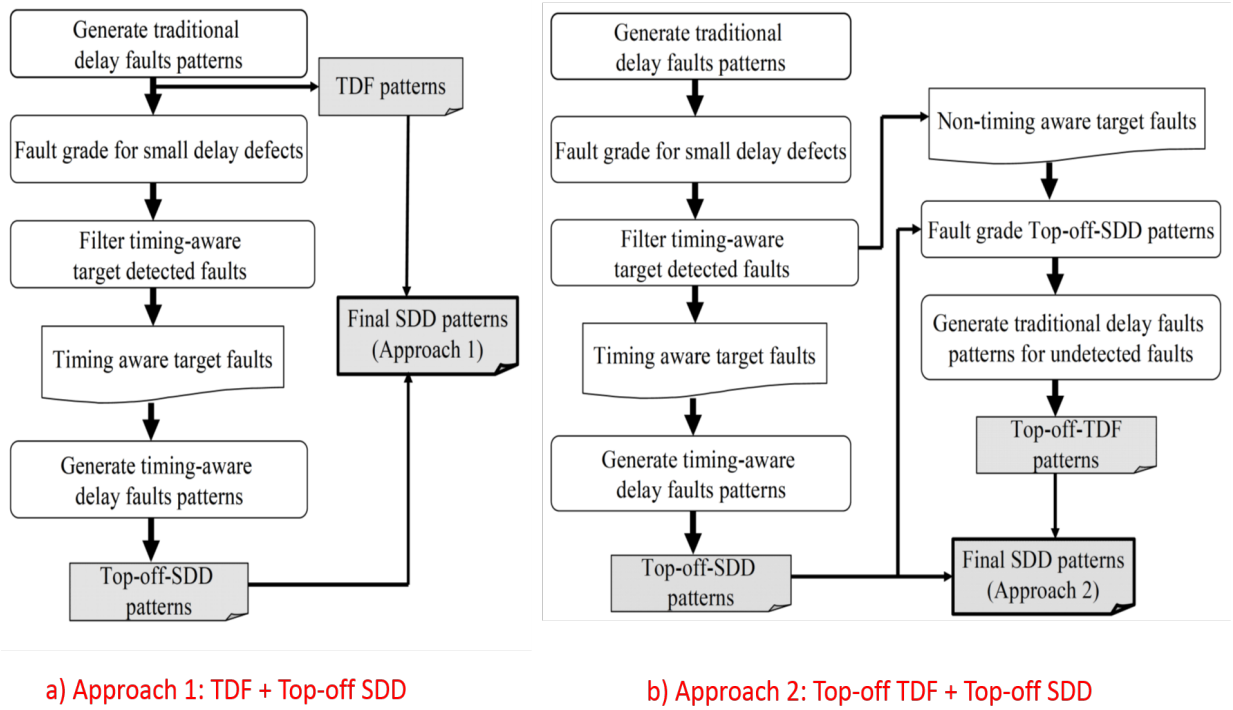


Figure 2.15: Automation flow of methodology 2 [2]

fault simulated to cover many traditional TDFs.

The *Approach 2* is similar to *Approach 1* except a fact that timing-critical and non-timing critical faults are isolated earlier in the flow diagram shown in Fig. 2.15b. It can be seen that top-off SDD patterns are fault graded with non-timing critical faults first and then conventional ATPG is used to generate final set of TDF-based ATPG patterns which is shown as "Top-off TDF patterns". In short, the *Methodology 2* outlines a standard approach of using TA-ATPG much more effectively, however, there is a list of short-comings which must be discussed and treated as possible enhancements:

1. *Methodology 2* must explore the opportunity to use path-based ATPG, as adopted in *Methodology 1*, before opting for TA-ATPG for targeting timing-critical transition faults. It is worth to remember that TA-ATPG is very expensive in runtime and pattern volume.

2. It is important to measure the correlation between circuit-level timing information provided by static timing analyzer (in the form of SDF files) with respect to more accurate dynamic timing simulation results. In general, STA is pessimistic in estimating worst-case path delays and it is expected that it would add additional paths as "timing-critical" which would eventually add more number of timing-critical transition faults. This would certainly over-burden the TA-ATPG for pattern generation as well as claiming SDD coverage.

This chapter has discussed the merits and demerits of two closest prior published test methodologies to target distributed delay defects. Along with inefficiency of conventional test approaches in pattern generation and runtime. In order to minimize the shortcomings of traditional test approaches, next chapter introduces **a test methodology** [4] **to cover SDDs more efficiently to reduce pattern volume and ATPG runtime**. Next chapter also discusses the modification in conventional SDD metric definition to improve confidence in covering small timing defects.

— * — * —

Chapter 3

A New Test Methodology to Target Small Delay Defects

3.1 Introduction

Timing tests that screen ICs for small delay defects (SDDs) have been of interest for over a decade [1, 2, 43, 44]. SDDs are assumed to add a small localized delay to a switching transition at the output of the target gate due to the defect. The primary sources of SDDs in earlier technologies were believed to be resistive vias and interconnects, and also gate oxide breakdown. However, the introduction of FinFET transistors at the 22nm node by Intel, and most other foundries at 16/14nm, has given rise to a new type of physical defect in ICs - the broken fin defect. In a FinFET, the conducting channel is in the thin vertical fin which makes the device three dimensional (3D). Increasing the drive strength of the transistor (equivalent of increasing device width W of a traditional planar transistor) requires increasing the height (H) of the fin. However, this is not very practical from a manufacturing perspective.

For structural strength, FinFET transistors are fabricated with a fixed and relatively small fin height [45]. Multiple FinFETs of this fixed size are connected in parallel to achieve the desired drive strength of a gate [46]. Thus, a single broken fin does not generally result in catastrophic failure of the circuit. Only the gate drive strength is degraded,

causing an increase in gate delay. For example, the loss of a single FinFET out of three parallel FinFETs driving the output of an inverter would increase the driving resistance and gate delay by approximately 50%. This would manifest as a small delay defect (SDD). Logic circuits designed in emerging gate all around (GAA) transistor technology, for 7nm and beyond, are also expected to display similar SDDs from transistor failures. This is a key reason for the increased focus on timing-aware TDF [47] test generation in industry.

The Transition Delay Fault (TDF) model targets slow-to-rise and slow-to-fall delay faults at a gate output. The traditional TDF model [39] is timing-unaware, and therefore only guarantees detection of a target fault if the extra delay due to the defect is "gross" - large enough to overcome any circuit timing slack. To ensure detection of small delay defects, TDF test generation can be made "timing-aware" such that each target TDF fault is detected using the longest sensitizable path to a circuit output. While this maximizes SDD detection, it also increases test set size, and significantly increases test generation cost because of the need for ATPG to work with circuit timing information. The increase in test cost has been a limiting factor to the adoption of timing-aware TDF tests [47] so far, although this is expected to change if, as expected, more SDDs are encountered as FinFET technology is widely adopted.

In first part of the thesis, we present a new test generation methodology for timing-aware TDF tests that for the first-time exploits Path Delay Fault (PDF) test generation to obtain tests for the targeted faults more efficiently, thereby reducing test generation effort. Additionally, the resulting timing-aware TDF test set is more compact and performs better over commonly used delay test coverage metrics. The results presented in this paper using industrial commercial tools on benchmark circuits show that the tests generated by the proposed approach achieves, on average, 12.5% and 35% reduction in pattern volume and runtime. It also gives 5% higher delay weighted test coverage in comparison to traditional TA-ATPG. The rest of this chapter is organized as follows. Section 3.2 discusses the basics of path delay and transition timing tests to develop the background and motivation. Section 3.3 presents an overview of the adopted new

approach. Section 3.4 describes the proposed methodology in detail and Section 3.5 validates the effectiveness of the proposed methodology on IWLS, ITC'99 and ISCAS'89 benchmark circuits. The last section concludes the chapter.

3.2 Background and Prior Work

Timing tests are two pattern tests [78, 79] that propagate switching transitions through the circuit and check if all outputs stabilize within the required clock period. Arguably, ideal timing tests are PDF tests that propagate transitions along individually targeted circuit paths to evaluate the timing of the complete path from an input to output. Such tests can detect delay defects at individual circuit nodes, as well as timing failures caused by an accumulation of extra delays that may be distributed among the gates and interconnections along the path. The distributed timing delays, from random process variations, are becoming an increasing concern at advanced technology nodes and it ideally require the use of PDF tests.

There are several practical issues that have greatly limited the adoption of PDF testing [64–68]. A key problem is the size of the test set, and resulting test application time, which grows dramatically faster with increase in circuit size compared to stuck-at tests. Also, if numerous random delays from process variability in a circuit are to be reliably tested, the PDF tests must be robust [70]. This means that the PDF test for a target path that is faulty (unacceptably slow) should not be potentially invalidated because of additional delays occurring in some other path that converges with the target path. It gives rise to an invalidating hazard at the observed output on the capture clock edge. Robust PDF tests ensure a reliable test for the target path irrespective of other delays in the circuit. Unfortunately, it has long been known that no robust PDF tests exist for many paths in a large circuit. Therefore, both robust [40] and non-robust [70] PDF test generation are widely supported by commercial ATPG tools [80]. Due to above stated issues, path delay tests have been adopted in practice only to a very limited extent.

It is the transition delay fault (TDF) that is commonly used today to enhance stuck-at tests in structural (scan based) testing of modern designs. In contrast to path delay tests, TDF tests target localized "lumped" delay faults at the circuit nodes. Faults that cause a node to become slow-to-rise or slow-to-fall are modeled as two separate and independently occurring faults by the TDF fault model. High coverage TDF tests aim to cover all such faults in the circuit. Note that just as in stuck-at fault testing, it is the individual nodes in the circuit that are targeted by TDF tests. Therefore, the growth in the TDF test set with increasing circuit size is similar to that for stuck-at tests - the size of the TDF test set is typically about 3-5 times the size of the stuck-at test set. This is orders of magnitude smaller than PDF test set for a large circuit which makes TDF tests much more attractive. Also, distributed delays from random process variations, which are much better detected by PDF tests, are still mostly handled with timing margins.

The primary target of timing tests are manufacturing defects which typically cause isolated and localized delays at some circuit nodes. These can be effectively targeted by TDF tests. However, a serious drawback of traditional TDF tests is that they are generated based only on the logical analysis of the circuit netlist, without any consideration of circuit timing. Consequently, such TDF tests will only detect a targeted delay fault in practice if the extra delay from the defect exceeds the timing slack (margin) in the path along which the transition is propagated to a circuit output. Smaller delay defects are absorbed in the timing slack and remain undetectable at the output on the capture clock edge. This problem is accentuated by the fact that for ease of test generation, ATPG programs often tend to use short and intermediate paths with relatively large slacks to detect the target TDFs. As a result, many small or even medium delay defects at a node can remain undetected in practice, even though the node is theoretically covered by the TDF test.

To maximize detection of such small delay defects (SDDs), "timing-aware" TDF test generation has been introduced in the past decade [75, 81, 82]. Timing aware tests target each TDF fault using the longest sensitizable path to a circuit output. This minimizes timing slack and causes smaller delay defects to exceed the clock period and

thus be detected. However, timing-aware tests require the ATPG to work with gate and interconnect timing information which greatly increases test generation time and cost. Given the increasing concern over gate drive strength degradation in FinFET designs, there is strong motivation for developing methods that more efficiently generate timing aware TDF tests.

In this thesis, we present such a methodology that for the first time exploits PDF test generation for the long paths to generate tests for many of the targeted timing-aware TDF tests. This not only saves test generation time, but because multiple TDFs along a path are detected by a single two-pattern test, also yields a more compact timing-aware TDF test set that can save test application time. Additionally, some of the target TDFs are detected through longer paths with lesser slack which enable the detection of smaller delay defects. However, not all transition delay faults can be covered by PDF tests along long paths. These remaining TDFs, not covered by the proposed PDF based test generation approach, are targeted using conventional TDF test generation to maximize test coverage.

3.3 Overview of the Proposed Approach

Any physical path that consists of gates and interconnects can be defined as an ordered set of intermediate nodes $\{g_0, g_1, g_2, ... g_{L-1}\}$ on a path p with length L. Let p:$a_0 \rightarrow a_0^{'}$ denotes the path delay fault that corresponds to the structural path p sensitized by applying a transition initialized to $a_0 = 0$ (rising transition at input) or 1 (falling transition). To cover distributed delay defects along a target path, a transition $a_0 \rightarrow a_0^{'}$ at g_0 (launch event) should be propagated along the path to the last node g_{L-1} and must be captured at a scan cell or primary output. The transition seen at any node g_i, where $0 < i < L$, with respect to g_0 depends on the inverting gates present in the sub-path between g_0 and g_i. Different types of PDF tests require the transition to propagate along the identified target path using *robust [40, 70] and non-robust test [70] criteria*. The path delay tests, in particular robust PDF tests, are theoretically the most effective circuit

timing tests, whereas their application to complex circuits is not very practical. Consequently transition delay fault tests (including timing aware TDF tests) are widely used in practice. TDF test generation programs are node oriented rather than path oriented, since they target slow transitions at individual circuit nodes.

In this work, we show that PDF tests can be used efficiently to detect all the transition delay faults that fall along the same path with a single two pattern set. Thus PDF tests can be exploited to improve timing-aware TDF test generation resulting in a more compact test set with better coverage for small-delay defects. This section explains how the test for a path delay fault covers a set of transition delay faults and with appropriate timing-aware detection coverage. A complete description of our test generation methodology is presented in the next section. Consider the path shown in Fig. 3.1 which may be the part of a bigger circuit. The circuit nodes are marked as *a, b, c, d, e, f, g, h, i, j, k, l, m*. Each node is considered to be affected by gross-delay defect which is associated with two transition faults: a slow-to-rise and a slow-to-fall fault. A set of nodes *a-c-f-g-j-k-m* in Fig. 3.1 constitute a path *p1*. Nodes *a* and *b* in this example are equivalent as they belong to primary inputs of the first gate of path *p1*. The target path is highlighted in

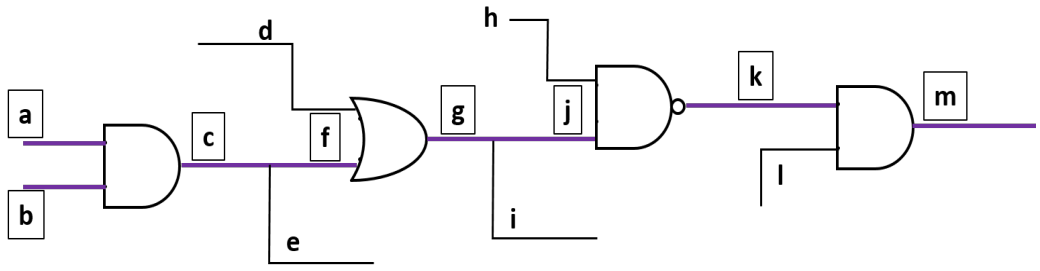


Figure 3.1: Transition delay faults covered by PDF based test

Fig. 3.1 which traverses through the set of nodes as shown inside the rectangular square boxes. Now assume that there exist scan-based tests through path *p1* to target respective path delay faults. For rising and falling transitions, PDFs of path *p1* can be written as p1:0→1 and p1:1→0 respectively. Let T0(n) and T1(n) denote the transition faults for slow-to-rise and slow-to-fall at a node *n*. By implication, PDF test for rising transition inherently covers a set of TDFs T0(a), T0(c), T0(f), T0(g), T0(j), T1(k), T1(m) and for

falling transition, it covers T1(a), T1(c), T1(f), T1(g), T1(j), T0(k), T0(m). It can be observed that gross-delay defects (TDFs) associated with rising and falling transition at any node that fall along the path *p1* can be detected by path delay tests i.e. p1:0→1 and p1:1→0. *The PDF tests, generated for a set of identified long paths, can be used to detect an ordered set of TDFs simultaneously along the specified long path in a timing aware manner*. Note that a single PDF test can only target half of the TDFs at intermediate nodes that fall on the target path.

Definition 1: *Speed-limiting transition delay fault (S-TDF)* is defined as an ordered set of transition faults that lie along an identified long path.

In this work, long paths are associated with a set of high timing variability paths [3] that may become speed-limiting on silicon under statistical process variations and aging. The core objective is to generalize the target list of S-TDFs beyond the traditionally targeted timing-aware TDFs. It is important to note that the circuit timing commonly used by the timing-aware ATPG is generated at a specified process-voltage-temperature (PVT) condition using static timing analysis (STA). Usually, the chosen PVT condition is the worst case corner as it is expected to cover all the long paths that may become timing-critical on silicon. However, process variations and aging can degrade additional paths to criticality as studied in [3, 5]. Apart from S-TDFs, the remaining non-timing critical TDFs must be classified as non S-TDFs.

For any circuit C, assume that total number of TDFs, S-TDFs and non S-TDFs are denoted as TDF(C), STDF(C), nonSTDF(C) respectively. Then, we can easily write STDF(C) $\subset$ TDF(C) and nonSTDF(C) $\subset$ TDF(C) such that,

$$TDF(C) = STDF(C) + nonSTDF(C) \tag{3.1}$$

Broadly, the total number of transition faults can be sub-divided into S-TDFs and non S-TDFs. For S-TDF classification in the example circuit shown in Fig. 3.1, assume that path *p1* that traverses through nodes *a-c-f-g-j-k-m* is timing-critical. If the number of nodes in example circuit is 13 then the total number of TDFs in this circuit is 2 x 13,

whereas the number of S-TDFs is 2 x 8 (that lie on the path *p1* where node a and b are equivalent) and number of non S-TDFs is 2 x 5. *The novelty of the proposed approach is that it first targets the path delay faults which are associated with an ordered set of identified long paths. Thereafter, it translates detected and undetected PDFs into detected and undetected S-TDFs along the target path respectively.*

It is important to define the conditions under which, an S-TDF can be classified *detected* along the longest paths as required by PDF based test generation. Let ST0(g_i) and ST1(g_i) denote slow-to-rise and slow-to-fall speed-limiting transition delay faults at a node g_i which belongs to an ordered set of nodes $\{g_0, g_1, g_2, ... g_{L-1}\}$ on a path p with length L, where $0 \leq i < L$. There exists another path q which consists of an ordered set of nodes $\{h_0, h_1,...h_{i-1}, g_i, h_{i+1},.....,h_{M-1}\}$ with length M, where $0 \leq i < M$. Lets assume that node g_i is common for paths p and q such that $0 \leq i < \text{Max}\{L, M\}$. ST0($g_i$) and ST1($g_i$) are considered to be detected using the proposed approach if they fulfill any of the following criterion (**S-TDF detection criterion**):

1. If even number of inverting gates exist on the sub-path between g_0 and g_i then ST0(g_i) is detected if the PDF based test p:0→1 exists for the target path p. Similarly, ST1(g_i) is detected if the PDF based test p:1→0 exists for the path p. The conditions would be inverted if odd number of inverting gates exist on the sub-path between g_0 and g_i that lies on the path p.

2. Assume that even number of inverting gates exist on the sub-path between h_0 and g_i such that ST0(g_i) is undetected through the target path p. If there exists a PDF based test q:0→1 such that the node g_i falls over it and common node for both the path p and q, then ST0(g_i) is detected. Under the same conditions, ST1(g_i) is detected if the PDF based test q:1→0 exists for the target path q. The conditions would be inverted if odd number of inverting gates exist on the sub-path between h_0 and g_i that lies on the path q.

In order to generate high quality PDF tests, we have used functional broadside [81] or launch-on-capture (LoC) transition test to avoid overtesting that may occur due to

unrestricted scan-based pattern generation. Usually, scanable flops allow the circuit to initialize in arbitrary state that may not be reachable during the functional operation. Any transition test consists of two vectors <v1p1,v2p2> where v1p1 is used for initializing the circuit and v2p2 is to launch a transition to target a list of faults. For LoC, v1 is shifted-in through scanable flip-flops while primary input vectors p1 and p2 are applied in functional mode. The state v2 is circuit response (if it exists) by applying primary and pseudo primary inputs p1 and v1 respectively. If state v1 is functionally reachable then state v2 is guaranteed to be reachable.

We are aware of the fact that identifying functional valid states in complex designs is NP hard problem, hence it is fair to start with practical and optimistic assumption that the state v1 is always reachable. With this assumption, LoC can be used to target an optimistic set of functional paths which may limit the circuit performance. This also helps in avoiding test pattern generation for those paths where v2 is not reachable even with an assumption that v1 is reachable. This finally results in obtaining high quality path delay fault based test patterns.

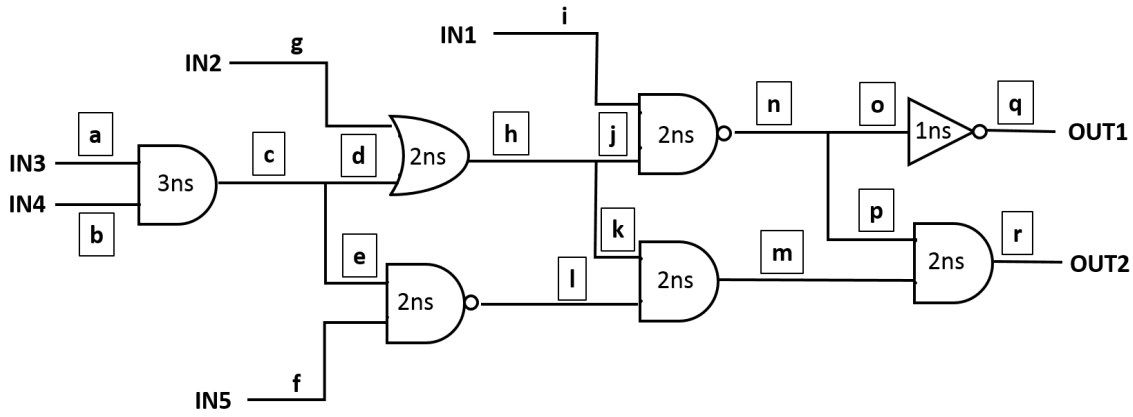


Figure 3.2: Example circuit synthesized to operate at functional clock period of 10ns

Now consider a circuit as shown in Fig. 3.2. The circuit has five primary inputs *IN1* to *IN5* and two primary outputs as *OUT1* and *OUT2* respectively. The gate delays are shown in *ns* and all intermediate nodes are marked from *a* to *r*. The circuit (under discussion) is synthesized to operate at functional clock period T_{sys} of 10ns. Lets assume that any path whose setup slack is less than or equal to 2ns can be categorized as timing-critical. The identified critical paths can be seen in Fig. 3.2 from *p1* to *p4*. As discussed earlier, total number of TDF in the above circuit is 18 and by using *S-TDF definition*, the number of transition faults that lie along the paths *p1* to *p4* is 15 which are associated with nodes *a, b, c, d, e, h, j, k, l, m, n, o, p, q, and r* respectively. However, path *p4* does not have robust or non-robust PDF test and by using *S-TDF detection criterion*, only node *p* falls under undetected S-TDF category. Whereas, TDFs associated with nodes *f, g, and i* are non S-TDF as they did not fall along the identified long paths and can be targeted using traditional ATPG.

3.4 The proposed methodology

This section outlines the proposed procedure for efficiently generating a compact and effective test set for timing faults. The complete methodology is broadly divided into three phases which are named as *Pass1, Pass2 and Pass3* respectively. *Pass1* is defined to exploit PDF based test generation to not only generate timing tests for identified paths, but it also helps in targeting a set of speed-limiting TDFs that can potentially results in more compact tests and perform better on commonly used SDD coverage metric. *Pass2* uses TA-ATPG to target undetected S-TDFs of Pass1 while *Pass3* targets all remaining faults using traditional timing-unaware ATPG.

3.4.1 PDF Based Test Generation to target S-TDFs (*Pass1*)

Fig. 3.3 shows the flow chart that needs to be adopted for *Pass1*. It starts with identifying N_c number of long paths of a circuit C by using a user-defined parameter for setup slack margin. In this work, we have used the procedure presented in [3, 5] to identify a set of

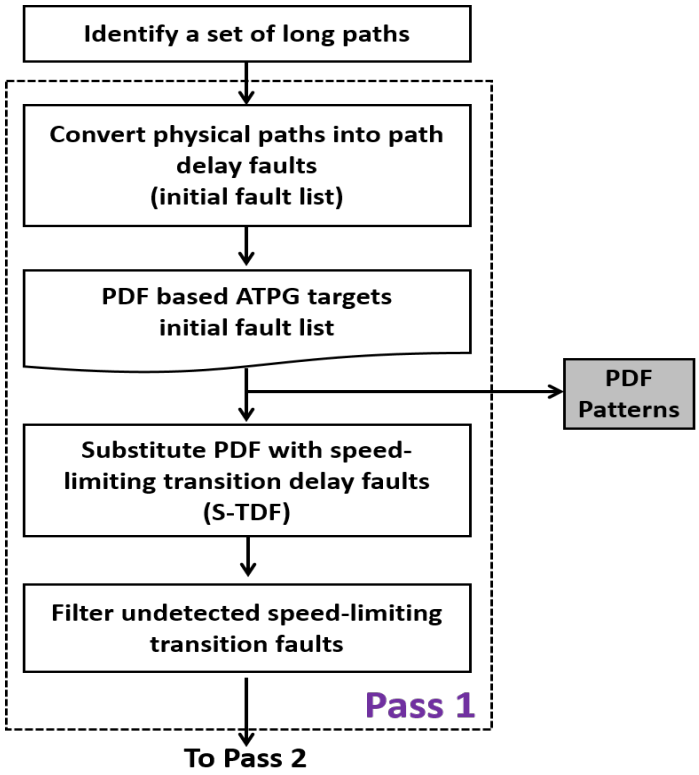


Figure 3.3: Flow chart of the proposed methodology: Pass1

high variability near timing-critical paths under statistical process variations and aging which would be discussed further in Chapter 4 and 5 respectively. However, the proposed methodology is not limited to some specific approach for identifying long paths. User can choose any method to identify such paths which also include procedures that involve statistical static timing analysis (SSTA) and SPICE simulations. We use the following criterion based on the user-defined parameter λ ($0 < \lambda \leq 1$)[47]. *A path is considered to be near timing critical if* $(T_{sys}$ - $T_p)/T_{sys} \leq \lambda$. T_{sys} and T_p are the functional clock period and longest path delay based on structural or statistical timing analysis respectively.

For example, assume that there are two paths (*p1 and p2*) with longest timing delays of 7.5ns and 8.5ns respectively. The clock period is 10ns and λ assigned a user-defined value of 0.2 then path *p2* is selected and path *p1* is dropped. After identifying long paths (whose number is N_c), convert them into path delay faults whose count is 2xN_c for rising and falling transitions. The fault list consists of all PDFs associated with

identified long paths. These faults are targeted by path-based ATPG using sensitization criteria: robust [40, 70] and non-robust [70]. The faults which are excited under these criterion can affect the performance of the circuit and collectively named as functional irredundant faults.

In order to avoid over-testing due to unrestricted scan-based tests, we are strictly using launch-on-capture (LoC) test scheme with an optimism that initializing vector (of two vector set) state is always functionally reachable for a circuit. Once path-based ATPG is finished, translate each PDF with an equivalent set of S-TDFs and classify them under detected or undetected category based on the ***S-TDF detection criterion*** as discussed in the previous section. It must be noticed that, the at-speed patterns are generated to target path delay faults not the associated S-TDFs. We use TCL scripts to filter detected and undetected S-TDFs using PDF detection information available at the end of ATPG run. The output of *Pass1* are the PDF based timing tests for detected path delay faults and a filtered set of S-TDFs which are associated with undetected PDFs.

3.4.2 Timing-aware ATPG to Target S-TDFs (*Pass2*)

Fig. 3.4 shows the flow chart that must be followed to cover remaining S-TDFs through the longest testable paths. *Pass2* starts with a set of undetected faults (S-TDFs) of *Pass1*. PDF based test patterns generated from *Pass1* are fault graded for the S-TDFs (which are undetected in *Pass1*) by setting ATPG in timing-unaware fault simulation mode. Fault grading will help in reducing the final pattern count and in proposed procedure, it is used without much penalty on tool runtime. Now, filter the undetected S-TDFs at the end of fault grading and the updated fault list of S-TDFs is to be targeted by TA-ATPG. The only reason to use TA-ATPG is to offset the huge runtime of path-based ATPG in testing a larger set of path delay faults in the circuit and lower test coverage. A better approach is to use path-based ATPG for an extremely limited set of identified PDFs and remaining can be targeted by TA-ATPG.

Next, TA-ATPG reads circuit timing generated at worse PVT corner and uses structural timing information to guide the excited faults through the longest testable paths.

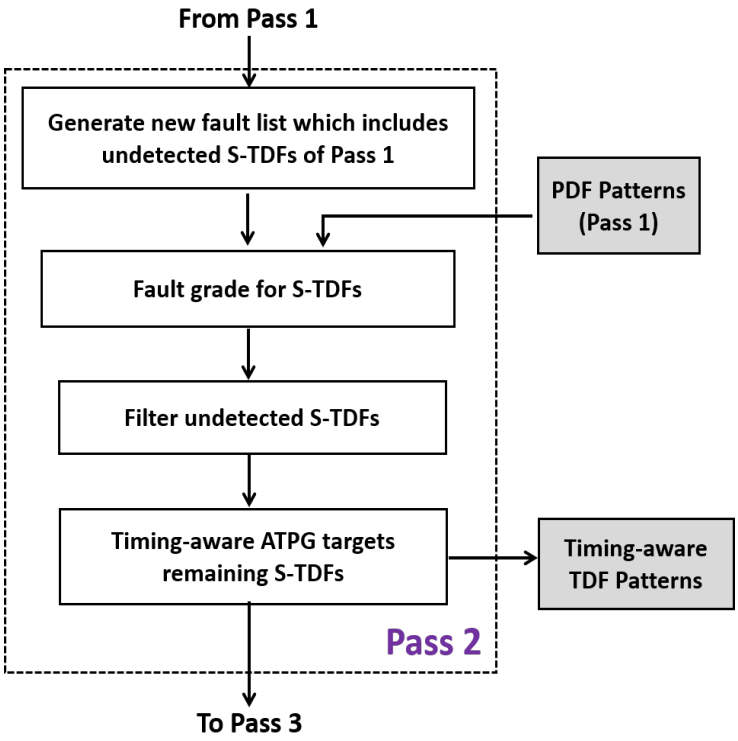


Figure 3.4: Flow chart of the proposed methodology: Pass2

However, using detection as the only criterion to drop the faults from fault list does not meet the requirement of near timing-critical path definition. Usually, commercial TA-ATPG [80] uses a fault dropping criteria, named as *Dropping based on Slack Margin* (DSM) [47], to trade-off between test generation effort and test pattern quality. We use the following fault dropping criterion based on the user-defined parameter δ ($0 \leq \delta \leq 1$)[47]. A fault is dropped from the fault list if $(T_a - T_{ms})/T_a < \delta$. T_a is the setup slack used by TA-ATPG for fault detection while T_{ms} is the minimum setup slack calculated using structural timing analysis with respect to functional clock period.

For example, assume that there are two faults f_1 and f_2 that must be excited through longest path p_1 and p_2, each having a minimum setup slack of 1ns (T_{ms}). However, TA-ATPG chooses path $p_1^{'}$ and $p_2^{'}$ to detect above faults with a slack (T_a) of 2.1ns and 1.9ns respectively. If δ is assigned a user-defined value of 0.5 then fault f_2 is detected and fault f_1 is dropped. It is worth to point out that $\delta = 1$ tends to match traditional fault

dropping criteria. At the end of *Pass2*, TA-ATPG is able to detect a set of S-TDFs through longest possible path and contribute significantly to improve SDD coverage. In the experimental section, the reported DTC accounts for S-TDFs only and for transition delay faults (which may include S-TDFs as well), traditional TDF coverage is reported.

3.4.3 Traditional ATPG to Target remaining Transition Faults (*Pass3*)

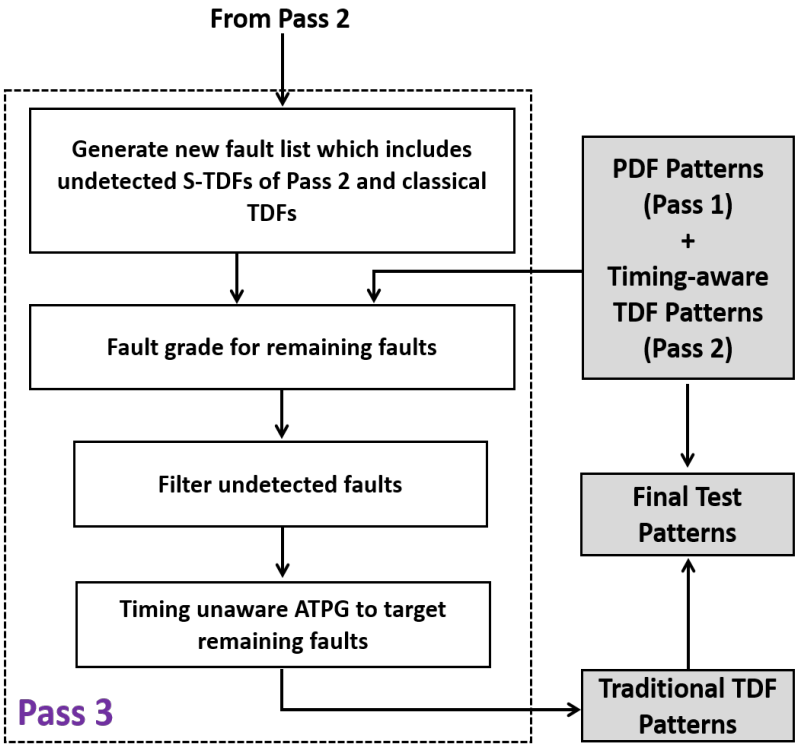


Figure 3.5: Flow chart of the proposed methodology: Pass3

Fig. 3.5 shows the last step to target undetected S-TDFs (from *Pass1* and *Pass2*) and non S-TDFs using traditional timing-unaware ATPG through any testable path. This helps in relying on industry's most prevalent fault model to target gross-delay defects. The procedure starts with generating a fault list which consists of undetected S-TDFs of *Pass1* and *Pass2* plus non S-TDFs (non-critical transition delay faults) of the circuit. The PDF patterns (obtained in *Pass1*) and timing-aware TDF patterns (obtained in

Pass2) are fault graded for remaining faults by setting the ATPG in timing-unaware fault simulation mode. Filter the undetected faults at the end of fault grading procedure that must be targeted by classical TDF based ATPG for pattern generation. Traditional timing-unaware ATPG ensures that no fault has been escaped untested during the proposed multi-step procedure. The final pattern count consists of PDF based timing tests from *Pass1*, timing-aware TDF patterns from *Pass2* and traditional TDF patterns from *Pass3*. In next section, we demonstrate the validity of our method for a range of benchmark circuits and compare the delay test coverage, pattern volume, and runtime of timing-aware ATPG with our new approach that effectively and efficiently exploits PDF tests wherever possible.

3.5 Experimental Results and Discussion

This section demonstrates the effectiveness of our proposed methodology by validating the experimental results on ITC'99, IWLS'2005 and ISCAS'89 benchmark circuits. The twelve circuits shown in Table 3.1 are selected on the basis of varying path length distribution and complexity. The benchmark circuits are synthesized using 28nm library to target functional clock period (T_{sys}) of 1ns and highly optimized for area using Cadence RTL compiler [83]. Total gate and flip-flop count of each benchmark circuit are shown in column 2 and 3 of Table 3.1. The total number of uncollapsed set of transition delay faults are shown in column 4. We have used a commercial ATPG tool FastScan [80] to target classical TDFs and the corresponding test coverage is reported under column 5.

The benchmark netlists did not include any compression or decompression logic and having an optimal chain length of 120 (scanable flip-flops per chain). User can choose any chain length, however this may results in increased test time. A Launch-on-Capture (LOC) based transition test is used during ATPG pattern generation. The pattern volume and ATPG tool runtime (in sec) are shown in column 6 and 7 respectively. Table 3.1 is included to verify the number of detected TDFs (which is a super set of S-TDFs and non S-TDFs) and corresponding traditional test coverage, against the proposed

Table 3.1: Characteristics of Benchmark Circuits and Traditional TDF Coverage

			Traditional ATPG			
Ckt.	#Gates	#Flops	#TDF	Cov.	#Pat	RTime
b15_1	3581	449	33618	91.3	877	15.5
b17_1	10942	1415	103366	94.96	963	41.4
b18_1	29963	3326	271908	93.68	996	184
b19_1	56919	6654	523474	95.39	1035	344.5
dma	9100	2200	106802	89	867	17.6
usb_funct	6382	1766	68118	91.91	276	5.8
ethernet	21379	10545	305462	97.57	1915	76.5
mem_ctrl	3286	1144	39414	91.07	643	10.1
vga_lcd	33377	17102	488376	93.76	3211	187.4
s35932	3268	1728	45390	80.02	57	2.6
s38417	3886	1564	49604	92.77	224	4.6
s38584	5504	1451	60112	88.63	306	5.2

methodology. Even though, timing-aware ATPG results are the actual candidates for comparison, we want to ensure that classical TDF coverage is not impacted while adopting the new procedure. All ATPG runs are executed on a host system using a pool of state-of-the-art servers with at least two processors available at all times with 8GB memory on Linux platform.

Table 3.2 shows the results of commercial TA-ATPG [80] where a set of *timing-critical transition faults* [75] (TCF), delay test coverage (TrdDTC), traditional TDF coverage (Cov.), pattern count (Pat) and tool runtime (RTime) are listed in respective columns. During pattern generation using TA-ATPG, if any transition delay fault f which can be propagated along a longest structural path p such that $(T_{sys}$ - $T_p)/T_{sys}$ is less than a user-defined value λ then it can be categorized as timing-critical transition delay fault (TCF). T_{sys} is the minimum functional clock period, T_p is the longest static path delay

Table 3.2: Test Patterns and DTC Results for Timing-Aware ATPG

Ckt.	Timing-aware ATPG (λ=0.2,δ=0.5)				
	#TCF	TrdDTC	Cov.	#Pat	RTime
b15_1	14133	83.93	91.3	1188	155
b17_1	44900	86.82	94.9	1329	409
b18_1	133108	84.15	93.65	1469	1721
b19_1	248333	85.44	95.36	1574	3082
dma	38107	82.82	89	1320	212
usb_funct	6301	89.61	91.9	360	34
ethernet	47957	89.44	97.57	2339	979
mem_ctrl	8236	84.63	91.07	837	110
vga_lcd	185701	91.51	93.74	4908	2694
s35932	23111	79.67	80.02	1334	254
s38417	9201	87.77	92.77	287	33
s38584	3711	83.57	88.63	398	36

through fault f and λ is a user-defined real number (as explained in Sub-section 3.4.2) between 0 to 1. The timing-critical TDF coverage is estimated using delay test coverage (DTC) metric proposed in [47] and defined below:

$$DTC = \sum_{i=1}^{N_f} \left(\frac{L_t}{L_{max}}\right) \times \frac{1}{N_f} \times 100 \tag{3.2}$$

where L_t is the longest tested path used by the ATPG and L_{max} is the longest testable path through any fault f. N_f is the total number of faults considered while estimating DTC. We used fault dropping based on slack margin (DSM) criteria while performing experiments with different values of δ, however the most optimum results are obtained using a value of 0.5. An optimal value of DSM is referred from [75] and a detailed example is discussed in Sub-section 3.4.2 to understand the concept of fault dropping.

If user sets a DSM value of 0 then ATPG starts targeting the faults through longest testable path else it drops without exploring any other less critical alternate path. While value of 1 allows ATPG to target faults through paths starting from timing-critical to non timing-critical, which is equivalent to traditional fault dropping criteria.

In order to minimize the test pattern count, TA-ATPG usually targets a subset of TDFs while minimizing the impact on delay test quality. To further reduce the pattern count and run-time along with improving commonly used delay test coverage metric, we proposed an approach that can be referred in previous section. *The complete procedure consists of sequential steps namely Pass1, Pass2 and Pass3 which inherently uses path-based ATPG, TA-ATPG and traditional ATPG respectively.* Table 3.3 shows the results

Table 3.3: PDF based Test Generation to Target Speed-Limiting Transition delay faults

	PDF based ATPG (Pass1)				
				#Det. STDF	
Ckt.	N_c	#PDF	#STDF	$STDF_{p1}^{det}$	CovPDF
b15_1	337	674	13903	2302	21.7
b17_1	1107	2214	43557	8933	23.6
b18_1	2365	4730	132219	13942	12
b19_1	4579	9158	247169	45527	17
dma	1646	3292	37622	11232	38
usb_funct	333	666	6207	1652	28.4
ethernet	967	1934	47782	7291	21.5
mem_ctrl	527	1054	8154	1114	19.54
vga_lcd	15560	31120	184326	42067	19.1
s35932	1918	3836	22028	20391	92.6
s38417	207	414	9080	2660	29.7
s38584	179	358	3449	1778	55.9

obtained by targeting a limited set of PDFs of various benchmark circuits listed in column 1. N_c is the number of identified longs paths, using the procedure presented in [3, 5], that may become speed-limiting under statistical process variations and aging. However, user can choose any approach for long path identification. The number of path delay faults are shown under column 3 and the count is twice the number of long paths (N_c) in a circuit to target both transitions (rise and fall). In Table 3.3, column 4 shows the total number of S-TDFs translated from PDFs (as discussed in section 3.3) which is approximately 25% of the total number of TDFs. The PDF coverage (*CovPDF*) of each benchmark circuit is shown under column 6. The benchmark circuit *s35932* shows exceptionally high PDF coverage (which is 92.6%) while *b18_1* shows the modest 12% PDF coverage.

We will show later in this chapter that high PDF coverage likely to improve the pattern volume saving, overall run-time and a relatively better DTC in comparison to TA-ATPG. At the end of *Pass1*, the detected PDF information generated by ATPG is used to filter out equivalent detected S-TDFs and is shown for each circuit in column 5 as $STDF_{p1}^{det}$. It is worth to point out that *initial fault list consists of only PDFs and once ATPG run is completed, detected and undetected PDFs are translated into detected and undetected S-TDFs respectively.* The generated pattern count and tool run-time are discussed later in this chapter.

Table 3.4 shows the effectiveness of TA-ATPG in targeting undetected S-TDFs of *Pass1* using circuit timing at worse PVT corner. Before using TA-ATPG to generate patterns, we must choose to perform fault simulation using generated path-based ATPG patterns of *Pass1*. *Fault grading is recommended if the non-robust fault sensitization criterion is used during PDF based test generation.* After fault grading, the number of detected S-TDFs can be seen in column 2 of Table 3.4 as $STDF_{p2}^{fg}$. The fault simulation helps in detecting an additional approx 1.4% of total undetected S-TDFs of *Pass1*, before being targeted by TA-ATPG. For example, lowest S-TDF detection (only 0.3%) is reported for circuit *b17_1* where 101 S-TDFs are detected out of 34624 (43557 - 8933) faults.

Table 3.4: Timing-aware ATPG to target undetected S-TDFs of Pass1

Circuit	TA-ATPG with $\delta = 0.5$ (Pass2)			
	#Det. STDF (Fault Grade)	#Target STDF	#Det. STDF	CovDTC
	$STDF_{p2}^{fg}$	$STDF_{p2}^{fl}$	$STDF_{p2}^{det}$	
b15_1	48	11553	10753	87.6
b17_1	101	34523	33291	86.83
b18_1	641	117636	114040	86.6
b19_1	776	200866	196831	84.9
dma	89	26301	26296	87.14
usb_funct	71	4484	4484	88.8
ethernet	482	40009	39960	91.5
mem_ctrl	107	6933	6820	84.91
vga_lcd	617	141642	141637	92
s35932	101	1536	1536	97.5
s38417	57	6363	6363	87.04
s38584	55	1616	1611	78.89

The remaining faults that must be targeted by TA-ATPG is shown as $STDF_{p2}^{fl}$ under column 3 of Table 3.4. As the target fault list already exists, we do not need to set any criterion (user-defined value λ) to identify timing-critical transition delay faults [75]. However, a DSM criteria must be set for TA-ATPG to avoid considerable runtime and high pattern volume. After running TA-ATPG, an approx. 98.5% S-TDFs are detected by choosing a DSM criteria $\delta = 0.5$ which is a most optimal value as suggested in [75] and further validated through our set of experiments. The number of detected S-TDFs is shown as $STDF_{p2}^{det}$ in column 4. TA-ATPG able to find long paths for 100% S-TDFs of circuits *usb_funct*, *s35932* and *s38417* with an average delay test coverage (DTC) of approx 91%. DTC of each circuit in listed as *CovDTC* in the last column of the

Table 3.4. The number of patterns generated using TA-ATPG and tool runtime will be discussed separately later in this paper.

Table 3.5: Traditional ATPG to target remaining faults

Circuit	Traditional ATPG (Pass3)				
	#Det TDF (Fault Grade)	#Target TDF	#Det TDF	CovTDF	Final TDF
	TDF_{p3}^{fg}	TDF_{p3}^{fl}	TDF_{p3}^{det}		
b15_1	1031	19484	16559	85	91.3
b17_1	2101	58940	53730	91.2	94.96
b18_1	15007	128278	111093	86.6	93.68
b19_1	14109	266231	242099	90.9	95.39
dma	1399	67786	56038	82.7	89
usb_funct	4039	57872	52361	90.5	91.91
ethernet	22306	235423	228000	96.8	97.57
mem_ctrl	1205	30168	26648	88.3	91.07
vga_lcd	11001	293054	262579	89.6	93.76
s35932	1998	21364	12295	57.6	80.02
s38417	2089	38435	34849	90.7	92.77
s38584	1955	54713	47878	87.5	88.63

The patterns generated in *Pass1* and *Pass2* are now fault simulated for all undetected faults (of *Pass1* and *Pass2*) and non-critical TDFs. The number of detected faults after fault grading is shown as TDF_{p3}^{fg} in column 2 of Table 3.5. An approx. 5.5% of the total number of classical TDFs are detected through fault simulation and remaining undetected faults are classified under a new fault list whose count can be seen in column 3 as TDF_{p3}^{fl}. In our ATPG runs, *fault grading and pattern generation are not separate steps* as we have used in-built TCL scripts to handle fault filtering and updating the fault list on the fly. Once the ATPG pattern generation is complete, the number of

detected faults and TDF coverage with respect to target faults (TDF_{p3}^{fl}) are shown as TDF_{p3}^{det} and CovTDF in column 4 and 5 respectively. Now, the overall TDF test coverage ($FinalTDF$) is calculated as,

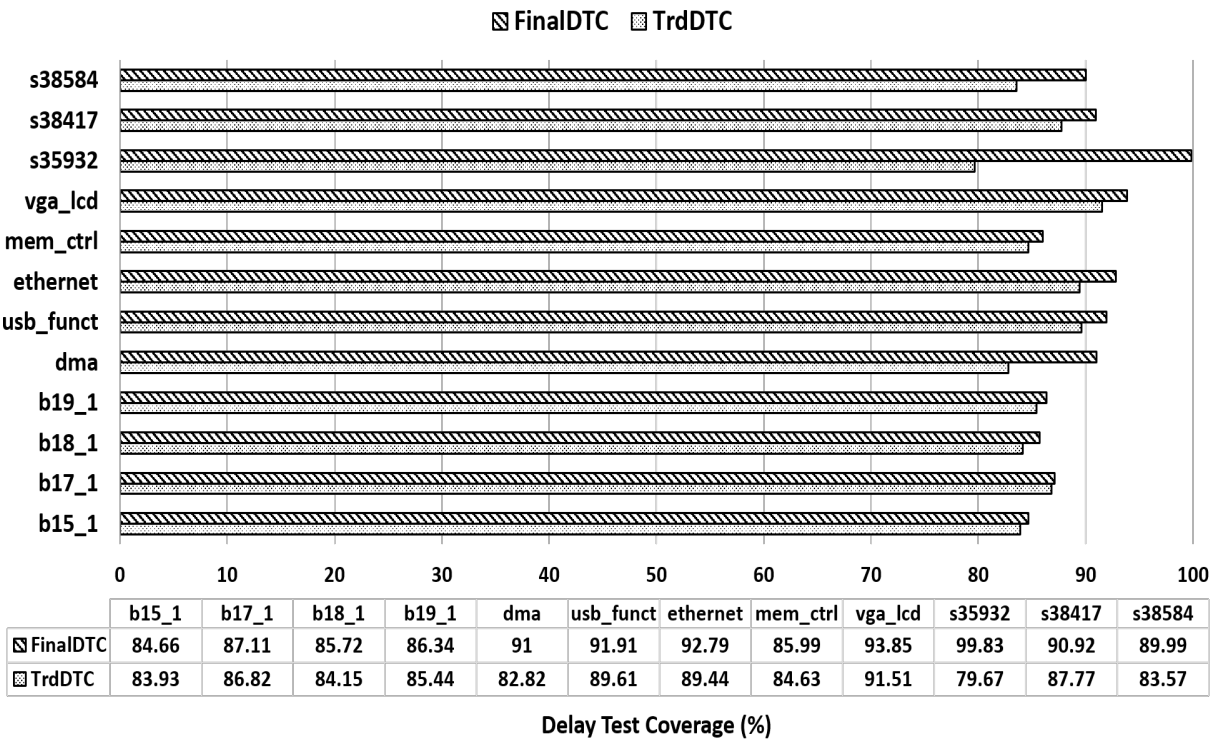
$$\frac{(STDF_{p1}^{det} + STDF_{p2}^{fg} + STDF_{p2}^{det} + TDF_{p3}^{fg} + TDF_{p3}^{det}}{\#Total\ number\ of\ TDFs} \tag{3.3}$$

where $STDF_{p1}^{det}$, $STDF_{p2}^{det}$ and TDF_{p3}^{det} are the number of faults detected by path-based ATPG, TA-ATPG and traditional ATPG in *Pass1*, *Pass2* and *Pass3* respectively. $STDF_{p2}^{fg}$ and TDF_{p3}^{fg} are those faults which are detected via fault-simulations using scan-based patterns of *Pass1* and *Pass2*. Estimating the TDF coverage at the end of our three step procedure is an important step to confirm that *no transition fault is escaped during the whole process*. We compared the obtained TDF coverage (of each circuit) shown in column 6 of Table 3.5 against the coverage reported by traditional ATPG in column 5 of Table 3.1. Next, the overall delay test coverage of S-TDFs is calculated as,

$$\frac{(STDF_{p1}^{det} \times 1) + (STDF_{p2}^{fg} \times 1) + (STDF_{p2}^{det} \times CovDTC)}{\#Total\ number\ of\ STDFs} \times 100 \tag{3.4}$$

where $CovDTC$ is the delay test coverage of faults $STDF_{p2}^{det}$. A weight of 1 is assigned to a set of faults which is excited and propagated through long paths using path-based ATPG. In DTC metric, weight assigned to a fault is the ratio of propagation delay of the longest tested path and the delay of structurally longest testable path. In case, the path-based ATPG chooses to propagate the fault effect through a longer path or if the tool knows that the chosen path belongs to a set of near timing-critical paths, then the weight assigned to that specific fault must be 1. The novelty of our approach is that, *we choose to propagate the fault effect at a node through one of the identified long path using path-based ATPG (or during fault simulation using PDF tests) and assigns a weight of 1 during DTC analysis*.

Fig. 3.6 shows the comparison between the proposed approach estimated overall DTC ($FinalDTC$) using Eqn. 3.4 and the traditional DTC ($TrdDTC$) reported by commercial TA-ATPG [80]. The percentage DTC improvement over the traditional TA-ATPG is approx 5% whereas a maximum improvement of 25.3% is seen on circuit *s35932*. On



	b15_1	b17_1	b18_1	b19_1	dma	usb_funct	ethernet	mem_ctrl	vga_lcd	s35932	s38417	s38584
FinalDTC	84.66	87.11	85.72	86.34	91	91.91	92.79	85.99	93.85	99.83	90.92	89.99
TrdDTC	83.93	86.82	84.15	85.44	82.82	89.61	89.44	84.63	91.51	79.67	87.77	83.57

Figure 3.6: DTC comparison between the proposed and traditional timing-aware ATPG

an average 3.1% DTC improvement has been obtained even if we exclude the utmost contribution by circuit *s35932*. We performed an in-depth analysis on each benchmark circuit to find out the probable reasons to enhance the DTC improvement which are as follows,

1. Effectiveness of ATPG while targeting PDFs using various sensitization criterion. A higher PDF coverage is likely to improve the overall DTC coverage.

2. The percentage of uniquely detected S-TDFs that are covered by path-based ATPG. There might be a case where PDF coverage is high, however, the number of detected PDFs translate to a lower count of S-TDFs that results in modest improvements.

3. Effectiveness of TA-ATPG in propagating the fault effects through longest testable paths which can help in increasing the weight assigned to each fault during DTC estimation.

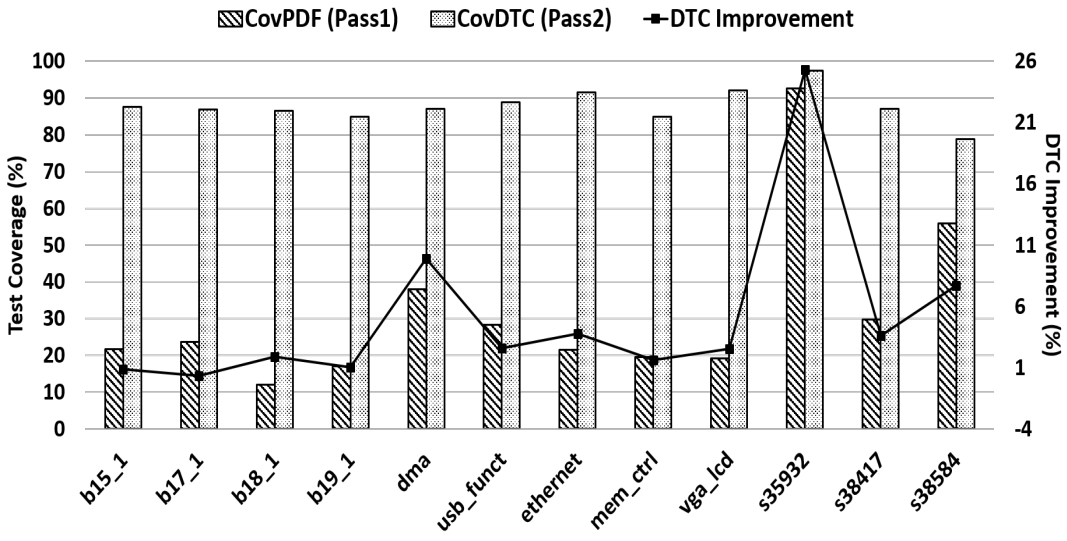


Figure 3.7: DTC improvement for each benchmark circuit

Fig. 3.7 shows the percentage DTC improvement of the proposed approach over commercial TA-ATPG for each benchmark circuit. The DTC coverage (CovDTC) obtained for each benchmark circuit during *Pass2* is greater than 80% and almost flat as can be seen in the above figure. Hence in most of the cases, the circuits having higher PDF coverage (*CovPDF*) in *Pass1* translate to improved DTC coverage as well. For example, *dma, usb_funct, s35932, s38417* and *s38584* show more than or nearly equal to 30% PDF coverage.

Using the new approach, it is assumed that these circuits get better DTC improvements in comparison to others. However, the circuits *usb_funct* and *s38417* achieve approx only 3% improvement which is lower than expected. After investigation, we found that successfully detected PDFs translate to a lower than expected uniquely detected S-TDF count. For example, 29% of total number of S-TDFs of circuit *s38417* are detected through path-based ATPG while rest of them are detected through TA-ATPG. In conclusion, *the maximum DTC improvement (for the proposed methodology) can only be achieved if PDF coverage is high and maximum number of unique S-TDFs are detected by PDF tests.*

Fig. 3.8 helps in explaining the reasons behind the pattern volume reduction, where

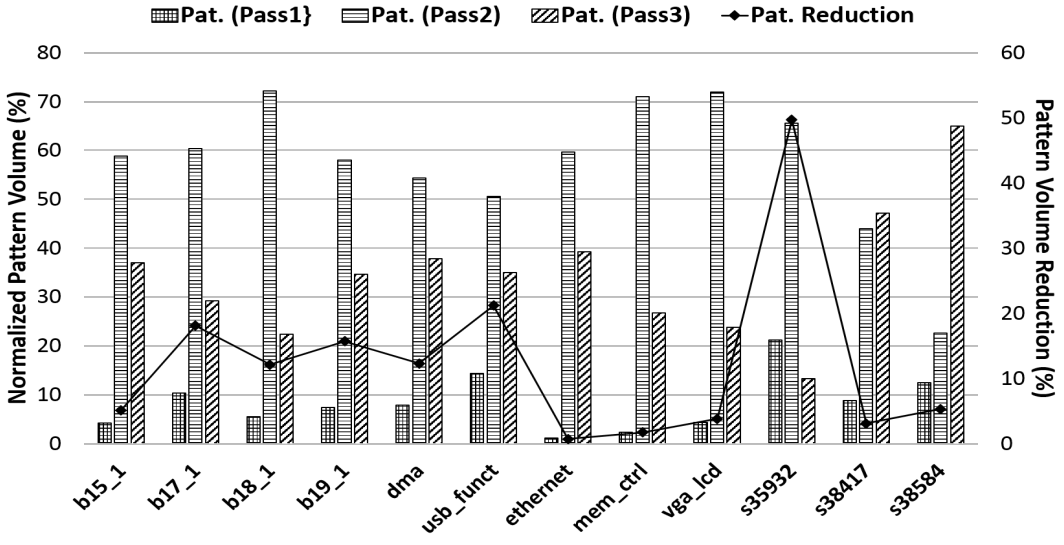


Figure 3.8: Pattern volume reduction (%) using the proposed methodology

we have normalized the pattern volume obtained in each step by the total pattern count for each benchmark circuit. The primary Y-axis shows the percentage of pattern contributed in each pass which is primarily associated with path-based ATPG, TA-ATPG and traditional ATPG respectively. The secondary Y-axis denotes the absolute percentage change in pattern volume in comparison to TA-ATPG generated patterns. For circuit *b15_1* in Fig. 3.8, the pattern volume contributed by *Pass1, Pass2* and *Pass3* are 4.2%, 58.8% and 37% whereas the reduction in pattern volume is approx 5%. It must be noticed that the patterns contributed by TA-ATPG dominates the overall pattern volume by 57.4% while the patterns generated by path-based ATPG and traditional ATPG contribute to 8.3% and 34.3% respectively.

Our one of the objective is to generate PDF tests as much as possible under a condition that it does not exceed TA-ATPG runtime in targeting a same set of distributed defects. Typically, an increase in PDF test pattern count translate to a better pattern volume reduction. However, it is dependent more on path length distribution and circuit topology which finally decide the pattern volume and ATPG runtime. For example, circuits *s38417* and *s38584* show a lower than expected pattern volume reduction even the PDF based pattern volume is significant. It is due to the fact that pattern volume is not

dominated by the TA-ATPG patterns obtained in *Pass2*, instead the pattern generated in *Pass3* limits the improvement in Fig. 3.8. The most ideal condition can be seen in circuit *s35932* where *Pass1* patterns successfully target most of the S-TDF which help in reducing the pattern volume of *Pass2* and further supported by a modest pattern sets from *Pass3*. Overall an average 12.5% pattern volume reduction is obtained and after excluding utmost pattern reduction contributed by *s35932*, an approx 9% reduction is easily achievable for rest of the benchmark circuits.

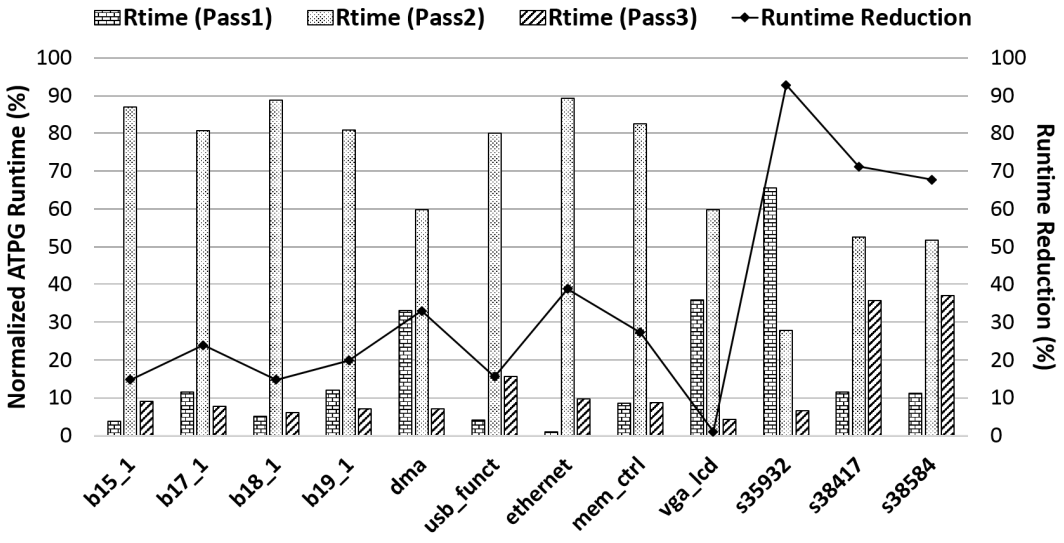


Figure 3.9: Run time reduction (%) using the proposed methodology

After discussing the ability to curb pattern volume using the proposed approach, it is important to evaluate the overall runtime in comparison to TA-ATPG. It is worth to point out again that we may be successful in reducing the runtime, only if path-based ATPG is being used to target a set of faults in effective and efficient manner. One of the major limitations of path-based ATPG is that the number of long paths in complex design increases dramatically even with the modest change in user-defined setup slack margin. A better solution is to add minimal path delay faults in the target list so that path-based ATPG outperforms to TA-ATPG in terms of runtime. Fig. 3.9 is similar to Fig. 3.8 in a sense that normalized pattern count is now replaced with normalized ATPG and fault grading runtime respectively while absolute percentage change in pattern volume

is replaced with respective runtime reduction.

The overall runtime is approximately 70% dominated by TA-ATPG (during *Pass2*), which is evident from Fig. 3.9. The runtime reduction observed for circuit *s35932* is 92.8% which is exceptionally good. It is due to the fact that the circuit has high PDF test coverage (that may be correlated to ATPG runtime) which ensures a lower runtime of TA-ATPG or traditional ATPG as they have to target lesser number of remaining faults. However, the circuit *vga_lcd* shows an opposite behavior with modest runtime reduction of 1.1%. After investigation, we found that path-based ATPG consumed 36% of total runtime and resulted in merely 19.1% PDF test coverage. As discussed earlier, *higher the number of uniquely detected S-TDFs associated with detected PDFs, better would be the results in terms of SDD coverage, pattern reduction and runtime*. Overall, an approx 35% runtime reduction is observed for all circuits and without *s35932*, it is approx 30%.

3.6 Summary

Tests to detect small delay defects (SDDs) in digital circuits have been developed and applied, to a limited extent, for over a decade. With more widespread deployment of FinFET technology, the need for SDD tests is growing. This is because logic gates commonly employ multiple identically sized FinFET structures in parallel to mimic a single transistor with sufficient drive strength; a broken fin in such a multi-fin transistor, which is a commonly observed failure mode, results in degraded performance. Similar concerns exist for the parallel structures in emerging gate all around (GAA) technologies targeting the 7nm node and beyond. Currently, SDDs are typically targeted by timing-aware TDF tests that propagate the fault effect along the longest sensitizeable path to minimize timing slack that can absorb and hide SDDs. However, timing aware test generation greatly increases test generation time, test set size, and at times, does not provide the desired coverage. In this thesis, we have presented a more efficient timing aware TDF test generation approach that exploits path delay (PDF) test generation to

generate many of the targeted TDF tests. TDF tests are node oriented, and consequently timing aware TDF test generation targets one node at a time. PDF tests for a long path, on the other hand, can simultaneously target and detect many TDFs on the nodes along the path in a timing aware manner. Exploiting PDF test generation thus helps in obtaining a more compact TDF test sets, reduces ATPG runtime and even performs better on commonly used delay test coverage metrics. The experiments presented in this paper show that the new approach resulted in approximately 12.5% reduction in pattern volume, 35% reduction in ATPG runtime and also a 5% improvement in delay test coverage (DTC) on various benchmark circuits when compared to existing commercial TA-ATPG approaches.

Chapter 4

Process-variation Aware Path Delay Estimation

4.1 Introduction

Integrated circuit in deep-submicron technology usually suffers from subtle additional delays due to process variations on both gates and interconnects [6, 9, 11]. The interconnect process variations have a greater impact on circuit delay as compared to variations on transistor [11]. Interconnect process variations are due to various effects like dummy fill insertion to minimize chemical mechanical planarization (CMP) thickness variability [8], dishing and erosion [8], structural and morphological, changes in electrical parameters [10] and cross-talk effects [9]. Multi-step CMP alone can change the total capacitance by a factor of over 10% while dishing and erosion phenomena can cause changes in the total resistance of interconnects by a factor of over 30% [7, 8].

The delay for many short interconnects in integrated circuits are generally small compared to gate delays. Thus, even as large as 30% variation in resistance of short interconnects usually have minimal impact on overall path delays. However, a significant impact on path delays can be observed on very long interconnects whose resistance can be as high as a Kohm or even greater. More importantly, such long interconnects when driven by high drive strength gates offer a much higher timing variability as compared

to when they are driven by gates with medium or low drive strengths. We develop a theoretical explanation for this using the *Elmore delay model* in Section 4.2. We demonstrate the impact of process variations [7, 8] on the overall timing delay of segments using Monte-Carlo simulations. In extreme process corners, the nominal delays of long interconnects, often comprising a significant fraction of the path delay, can exceed the timing slack and cause timing failures. If such potentially slow paths are not properly tested then, parts from slow process corners can escape to cause field failure and contribute to increase DPPM.

Most of the previous works [1, 6] aim at finding the longest paths in a circuit to detect delay defects caused by resistive shorts, resistive opens, resistive bridges and process variations in the gates. Timing aware ATPG [47] has been proposed to improve the test pattern quality at the expense of relatively high pattern count. To improve on test pattern count without sacrificing much on delay test coverage (DTC), timing-critical transition delay fault testing procedure [75] has also been proposed. Various path selection procedures [84, 85] based on statistical timing analysis (SSTA) are introduced to improve DTC by targeting a set of critical paths. *The key objective of this chapter is to identify high delay variability paths using delay prediction models (in the form of look-up tables) which are developed using SPICE level simulations and formulate a heuristic based path selection algorithm in Section 4.4 which identifies a candidate set of critical paths under spatial statistical variations.* We also propose "three pass" method with the aim of developing delay timing tests to improve the overall DTC metric of complex designs. Section 4.3 elaborates inaccuracy in estimating path delay measured using timing libraries of standard cells versus actual silicon. The effectiveness of the proposed algorithm is demonstrated in Section 4.5 on IWLS, ITC'99 and ISCAS'89 benchmark circuits and Section 4.6 concludes the chapter.

4.2 Key innovation of the proposed methodology

4.2.1 Segment Definition

Definition 1: A Segment is defined as the combination of a driver gate and an interconnect.

Segment is the smallest entity while defining a target path. Referring to Fig. 4.1, let a target path be $\{g_1, w_1, g_2, ...w_{n-1}, g_n, w_n\}$, where w_i denotes the on-input interconnect or wire driven by gate g_i. Dash arrows show either fan-in or fan-out to a specified node or gate. For any target path p, the following definitions are used to calculate path delay under aging:

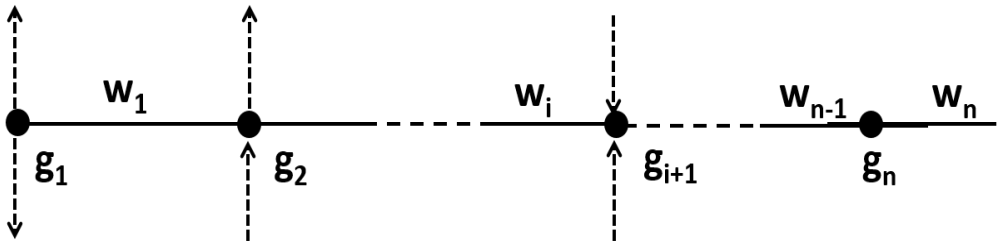


Figure 4.1: Defining a path using segment

- The maximum and typical absolute gate delay estimated by dynamic timing simulation (running SPICE level simulation) at worst and typical PVT corners are GD^{sp}_{max} and GD^{sp}_{avg} respectively.
- Similarly, the maximum and typical absolute wire delay estimated by SPICE simulation are WD^{sp}_{max} and WD^{sp}_{avg} respectively.
- By Definition 1, a segment comprises of a gate and an interconnect ($g_i + w_i$). The worst delay of a segment s_i is defined as:

$$SD^{sp}_{max}(i) = GD^{sp}_{max}(i) + WD^{sp}_{max}(i) \tag{4.1}$$

- for any path p_i, the maximum path delay is expressed as:

$$PD^{sp}_{max} = \sum_{i=1}^{n} SD^{sp}_{max}(i) \tag{4.2}$$

  $1 \leq i \leq n$, where n is the set of segments or gates in a path.

- The maximum path delay of any path p_i under variability can be expressed as:

$$PD^{sp}_{max} = \sum_{i=1}^{n} (GD^{sp}_{max}(i) + WD^{sp}_{max}(i)) \tag{4.3}$$

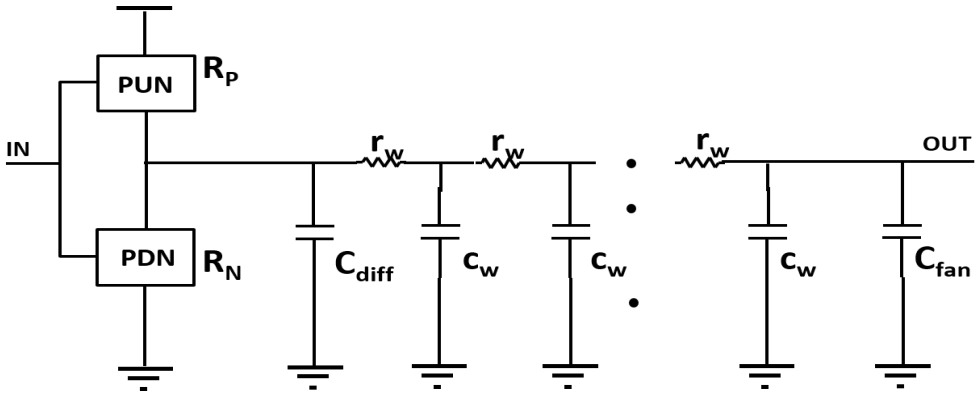


Figure 4.2: CMOS driver gate and distributed RC load as Segment

4.2.2 Segment Delay Model

A segment consists of a driver gate (CMOS logic) and wire load which is modelled as distributed RC network (T model) as shown in Fig. 4.2. PUN and PDN are the pull-up and pull-down network with on-resistance R_p and R_n respectively. C_{diff} is diffusion capacitance of driver gate and C_{fan} is input capacitance of fanout gates. Wire length L, with lumped resistance (R_w) and capacitance (C_w), is partitioned into N identical segments with r and c as wire resistance and capacitance per unit length. The per unit length r_w and c_w are given by rL/N and cL/N respectively.

In this work, driver gate of Fig. 4.2 has fixed fanout of 3 two-input NAND gates and wire load has N = 30 intermediate sections for distributed RC network. An extensive

work by Bowman *et al.* [86] and Lopez *et al.* [87] have been done on determining the maximum delay of critical path identified in a microprocessor. Critical path can be modeled as a chain of two-input NAND gates with each gate having a fan-out of 3 [86, 87]. It is worth to point out that the segment model presented in Fig. 4.2 can only be used during LUT generation as discussed in Sub-section 4.4.2. When the wire time constant (R_wC_w) is comparable to driver time constant (R_pC_{diff}) then the process variation effect on wires would have greater impact on overall timing delay. The resistance of short interconnects typically have at most a few tens of ohms and are usually driven by standard drive transistors with effective source (channel) resistance of one (or a few) Kohm(s). Thus the impact of RC delay variability on timing paths is typically unnoticeable in terms of percentage change in path delay variation due to interconnects. On the other hand, very long interconnects whose resistance is in the range of a Kohm and typically driven by high drive strength transistors (resistance in the range of 100 ohms) result in greater variability on timing paths due to process related variations.

4.2.3 Experimental Justification

In order to observe the timing behavior of interconnects when driven through different drive strength gates, SPICE simulation is used on 28nm library to acquire detailed delay information of all basic to complex gate types, including INV, NAND, NOR, MUX and AOI. For experiments, we used the following typical (PVT_{typ}) and worst (PVT_{worst}) corners to measure the absolute percentage delay variations across the segments.

PVT_{typ}:

- Process: TT
- Voltage: V_{nom}
- Temperature: $+25^0$C
- Threshold voltage change (ΔV): 0%
- Wire resistance change (ΔR_w): 0%

- Wire capacitance change (ΔC_w): 0%

$\underline{PVT_{worst}}$:

- Process: SS
- Voltage: V_{nom} - 10%
- Temperature: $+125^0$C
- Threshold voltage change (ΔV): +30%
- Wire resistance change (ΔR_w): +30%
- Wire capacitance change (ΔC_w): +10%

The simulations are carried out for the following realistic wire loads: L1(50Ω,2fF), L2(250Ω,10fF), L3(500Ω,20fF), L4(750Ω,30fF), and L5(1000Ω,40fF) as shown in Fig. 4.3. The naming convention used for the gate is <gate type, # inputs, drive strength>, e.g. NOR2_X8 is a two input nor gate with drive strength 8. We calculated the percent-

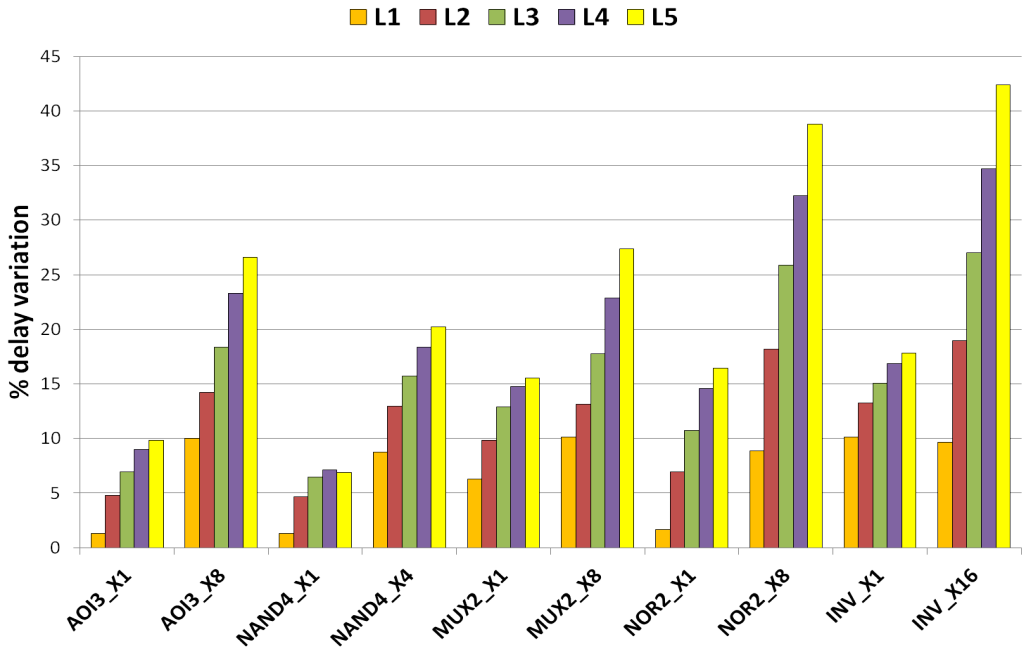


Figure 4.3: Effect of delay variation on segments due to interconnect geometry

age change in delay by comparing the absolute timing delay measured across a segment without process variations (at PVT_{typ}) with the measured absolute timing delay across the same specified segment under process variations (at PVT_{worst}). The percentage change in delay is calculated using below Equation,

$$\frac{SD^{sp}_{max} - SD^{sp}_{avg}}{SD^{sp}_{avg}} \tag{4.4}$$

We run SPICE simulation at typical and worst process corners to perform timing delay measurement. It can be observed in Fig. 4.3 that wire loads driven from high drive strength gates display a relatively higher delay variation in comparison to low drive strength gates. For same wire load L5, INV_X16 have showed a delay variation of 42.3% as compared to 17% delay variation with INV_X1. We have not considered cross talk and other aging effects.

4.3 Silicon Data analysis

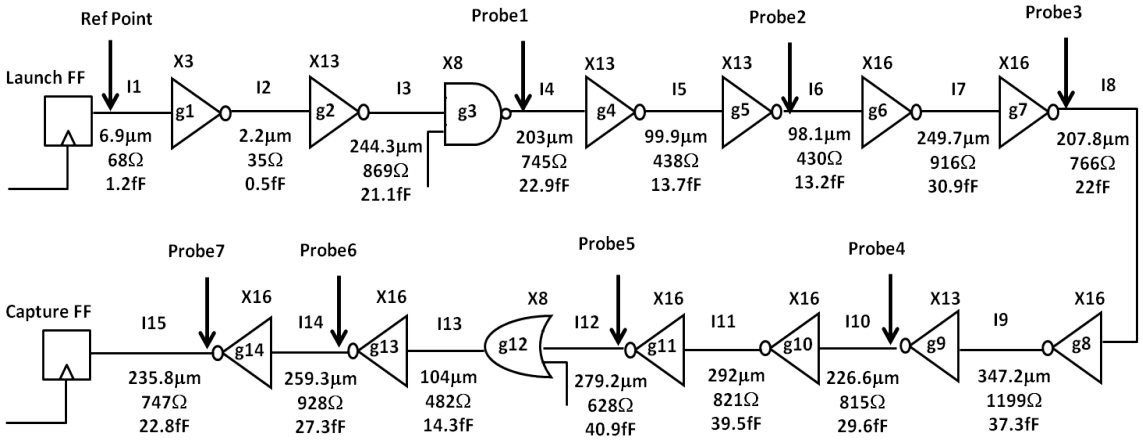


Figure 4.4: Speed limiting path on silicon

This section demonstrates the unexpected process-related delay variability observed in significant numbers of manufactured parts which resulted in lower frequency bins. During the initial characterization of silicon wafers in a 28nm technology, it is noticed

that a small number of paths have poor performance and behaved slower than expected. One of the speed-limiting or critical path which consists of gates *g1* to *g14* is shown in Fig. 4.4. The interconnect physical parameters (physical length, resistance, and capacitance) are mentioned in Fig. 4.4 for interconnects *I1* to *I15*. In order to measure the delay

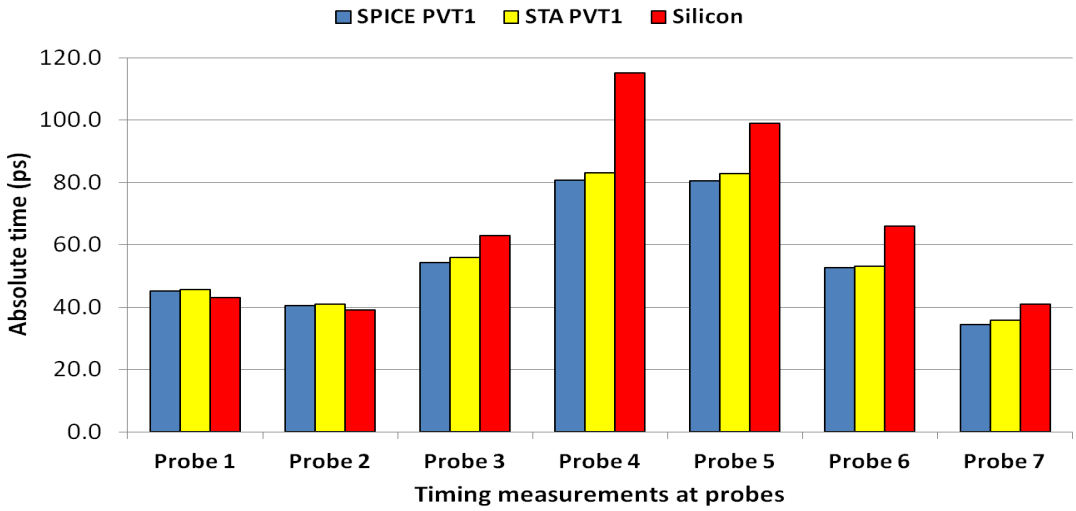


Figure 4.5: Timing comparison at various probes

of identified path on silicon, several contact-less probe points (*Probe 1* to *Probe 7*) are identified using Laser Voltage Probe (LVP) technique [88]. To find the root-cause of the failure, delay is measured (see Fig. 4.4) from *Ref Point* to *Probe 1*, from *Probe 1* to *Probe 2*, and so on at a temperature of $+105^{o}$C with 1V power supply. There is no voltage drop, ambiguous clock skew or other unacceptable variations are reported during Failure Mode Analysis (FMA). SPICE and STA estimated delays are compared with silicon results as shown in Fig. 4.5. PVT1 is the worse process corner which corresponds to slow-slow, 1V, $+105^{o}$C.

Each probe points (on the Y-axis of Fig. 4.5) measures the absolute timing delay from previous probe point. For example, *Probe 4* measures the timing delay of two inverters (*g8* and *g9*) and two interconnects (*I8* and *I9*) with respect to *Probe 3*. It can be observed that STA estimated delays (*STA PVT1*) are always pessimistic as compared to SPICE results (*SPICE PVT1*) at all probes. The STA analysis is able to predict the

timing delays for segments related to *Probe 1* and *Probe 2*, however it is not true in case of long interconnect dominant segments which corresponds to *Probe 3* to *Probe 7*. *Probe 4* offers the maximum delay variation of 38.5% between STA predicted delay and real silicon results. The percentage variation is in-line with our predicted variation as shown in Fig. 4.3 for basic inverter. The analysis demonstrates that discrete-valued timing models may become invalid for estimating delays in few extreme process related variations in deep sub-micron technology.

4.4 Path Selection Procedure

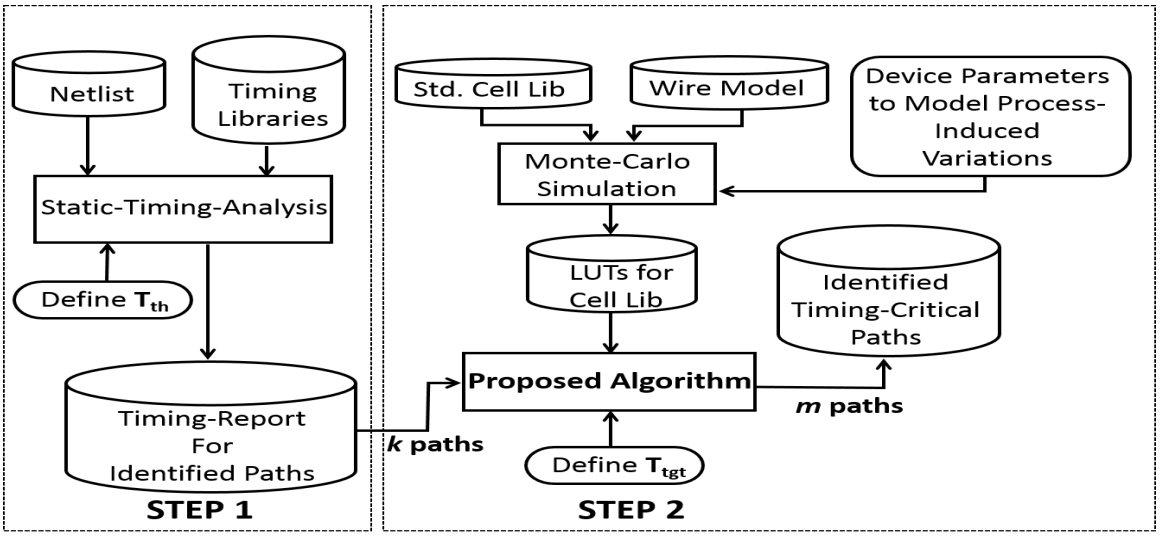


Figure 4.6: Flow diagram of the proposed methodology

4.4.1 Flow diagram of the proposed methodology

Before we start discussing a systematic flow to generate look-up-tables (LUTs), first we introduce terminologies that will be used in rest of the paper. T_{sys} is defined as functional clock period used to test the circuits at-speed. T_{tgt} is the absolute minimum clock period for the circuits, assuming zero variations due to spatial device parameter fluctuation. T_{th} is the long and intermediate path delay threshold limit that guides STA

tool to identify probable set of paths that may show higher timing variability. In the proposed procedure, T_{sys} is fixed for a synthesized circuit while T_{th} and T_{tgt} are user-defined values. A simple relation among the variables are described earlier in [5] as $T_{th} < T_{tgt} < T_{sys}$. A flow diagram of the proposed methodology is shown in Fig. 4.6 where two step procedure is adopted.

STEP 1: In order to handle process variations, STA provides multi-corner analysis to predict worse-case circuit delays. The key objective of this step is to identify a set of intermediate and long paths whose delay exceed T_{th} (a user defined value). We have used commercial STA tool (Cadence encounter timing system [83]) constrained with $T_{th} = 0.5T_{sys}$ in this step. Note that, the proposed procedure is not limited to some specific static timing analyzer. It reads physical netlist, timing libraries, timing constraints and physical data like placement, floorplan, routing etc. The PVT condition at which static timing analysis has been done is an important factor to influence the results of the proposed methodology. We will revisit this factor later in this chapter. The final outcome from *STEP 1* is path-based timing reports of all paths (lets say this number is k) whose delays are greater than or equal to T_{th}.

4.4.2 Look-up Table (LUT) Generation for Standard Cells

Before we discuss *STEP 2* in detail; it is important to understand, *how the absolute delay variations due to process variations are captured into LUTs.* We first identify different wire loads from intermediate metal layers using physical design database. These identified wire loads range from smaller to longer interconnect size, e.g., W1(20Ω, 0.8fF), W2(40Ω, 1.6fF), W3(80Ω, 3.2fF) with lumped resistance and capacitance progressively increasing up-to W30(1160Ω, 46.4fF). By using the segment model explained in Sub-section 4.2.2, each driver gate from 28nm library drives a specified wire load from W1 to W30 at a time, that finally translates to 30 different cases. The following procedure is used to generate LUT corresponds to a standard cell,

- For each case of specified driver gate driving different wire loads (W1-W30), no device parameter variations are applied. For every simulation at PVT_{typ}, delay

measured for gate, interconnect, and segment using SPICE simulator are GD^{sp}_{avg}, WD^{sp}_{avg}, and SD^{sp}_{avg} respectively.

- Now apply extreme parameter variations on transistors as mentioned in Sub-section 4.2.3. For a specified driver gate and using the same simulation setup, SPICE simulation is performed at PVT_{worst} with each wire load (W1-W30). The maximum delay measured across the segment is SD^{sp}_{max}.

- Find the multiplication factor by using below Equation,

$$1 + \frac{SD^{sp}_{max} - SD^{sp}_{avg}}{SD^{sp}_{avg}} \tag{4.5}$$

The absolute delay variation which is defined above, is termed as *delay multiplication factor due to process variations* (M^p_f). While generating LUT for a specified standard cell, we store WD^{sp}_{avg} and corresponding delay multiplication factor M^p_f. For example, a generated LUT for two-input NOR gate is shown in Table 4.1 with drive strength X1 and X8 respectively. The above table stores a delay variation factor M^p_f which is a *delay*

Table 4.1: Example Look-up Table for two input NOR gate

NOR2_X1		NOR2_X8	
WD^{sp}_{avg} (ps)	M^p_f	WD^{sp}_{avg} (ps)	M^p_f
1.4	1.016	0.7	1.089
7.2	1.069	4.8	1.182
16.1	1.115	12.2	1.273
27.4	1.146	21.5	1.322
41.3	1.165	32.7	1.353

multiplication factor that accounts for with-in die process related variations. In Table 4.1, WD^{sp}_{avg} and M^p_f are shown in column 1, 3 and 2, 4 respectively. WD^{sp}_{avg} is used as *input search data to LUTs* to obtain M^p_f. If certain value of WD^{sp}_{avg} is not found in the LUT then an approximate values of M^p_f is calculated using *linear interpolation method.*

It can be seen in Table 4.1 that an interconnect with wire delay (WD^{sp}_{avg}) of 4.8ps when driven by a NOR2_X8 gate results in $M^{p}_{f}[5]$ as 1.182. This translates to 1.8% absolute delay variation across the segment under extreme process-variations. While LUT generation procedure, the PVT conditions under which the dynamic simulations are carried out is a crucial factor.

4.4.3 Path Delay Prediction

STEP 2 outlines an algorithm that uses path-based static timing reports of k *paths* identified from *STEP 1*. The path-based STA report consists of gate delays (GD^{sp}_{avg}) and wire delays (WD^{sp}_{avg}) that build the target path. In our case, it is worth to mention that multi-corner STA and multi-corner SPICE results are well correlated at PVT_{typ} and thus, we use them interchangeably at this corner. For complete description and implementation of the algorithm, refer next sub-section. In this step, inputs to the algorithm are: *a)* generated LUTs for standard cell library as described in Sub-section 4.4.2, *b)* user-defined value T_{tgt}, *c)* path-based static timing reports. The proposed algorithm uses the following equation to estimate worst-case path delay,

$$PD_{algo} = \sum_{i=1}^{n} (SD^{sp}_{avg}(i) * [M^{p}_{f}(i)]) \tag{4.6}$$

$0 \leq i \leq n$, where n is the number of segments in a path. SD^{sp}_{avg} is the segment delay estimated by SPICE or STA at PVT_{typ} corner. Each segment of a target path has its own absolute delay variation. The proposed algorithm multiplies STA estimated absolute delay (which does not include any variations) of each segment by delay multiplication factor (obtained from LUTs) and finally, adds individual segment delay to estimate worst-case path delay. At the end, any path whose delay is greater than or equal to T_{tgt} would be categorized under m timing-critical paths (see Fig. 4.6).

4.4.4 Implementation of Algorithm for Path Selection

This subsection outlines the pseudo-code for the proposed algorithm (***Algorithm 1***) which overcomes the problem of estimating pessimistic delays using STA timing models under extreme process variations.

Algorithm 1 ***Path_Selection_Procedure***

Procedure: **Enumerate all paths whose delay is greater than T_{th}**

```
1.  PD_algo = 0;
2.    while all paths are not covered {
3.     while all segments are not covered on selected path {
4.          for selected segment {
5.              getdelayscalefactor(gate_type,inter_delay,drive_strength);
6.                  PD_algo += segment_delay * M^P_f ;
7.          end for
8.      end while
9.   if PDalgo ≥ T_tgt {
10.      return(SUCCESS);
11.        if ((PD_algo - PD^sp_avg)/ PD^sp_avg) ≥ β {
12.          return(HIGH_VARIABILITY_PATH);
13.           reset_all_delays;}
14.        else {reset_all_delays;}}
15.  else { return(FAILURE);
16.           reset_all_delays; }
17.  end if
18. end while
```

subroutine getdelayscalefactor **(x,y,z)**

```
1.  input x – gate type;
2.  input y – interconnect delay;
3.  input z – gate drive strength;
4.   Return delay multiplication factor (M^P_f), based on type of gate, its
     drive strength and wire load, from pre-defined lookup table prepared
     for each standard cells present in technology library
```

In order to achieve target yield, timing margins are built by the designers to account for process variation and other random effects to ensure functional silicon. The exact value of timing margin is dependent on chosen technology node and manufacturing foundry. Any path whose timing delay exceeds a predefined threshold value T_{th} is

considered for delay variability computation by the proposed algorithm. The relation among the variables is already discussed. For each segment in a path, the absolute value of M_f^p is calculated through subroutine ***getdelayscalefactor***.

If the computed path delay (PD_{algo}) is greater than or equal to T_{tgt} then the path can be categorized as critical or speed-limiting. While delay computation, if the percentage delay variation in a path exceeds a user-defined threshold value β then the algorithm stores the specific path in a separate table. The list of high variability paths can be considered for possible re-design, but Engineering Change Order (ECO) may be expensive late in the design cycle. If the number of identified paths in the first pass is k and the average number of segments in a path is n, then the time-complexity of the algorithm is *O(kn)*. As the calculation of path delay is independent of each other, it can be parallelized to reduce the time-complexity to *O(kn/T)* where T is number of threads.

4.5 Experimental Results and Discussion

In this section, we present an experimental results for IWLS, ITC'99, and ISCAS'89 benchmark circuits to demonstrate the effectiveness of our proposed methodology.

4.5.1 Experimental Set-up

In order to make a relatively complex design, we used ten copies of each benchmark circuit listed in Table 4.3. A commercial synthesis and physical design tool (Encounter RTL compiler [83]) is used to optimize the design for aggressive area-optimization. Synthesis is done at a target frequency (T_{tgt}) of 1.25GHz to achieve highest bin frequency (T_{sys}) of 1GHz after including process variation margin of 20%. We utilized six metal layers for wire routing where M5 and M6 are considered as global routes while all other layers are used for local routing. In final layout, the global routing dominates over local routing for the following circuits:

In order to generate lookup tables for 920 standard cells available in 28nm library, we carried out 110400 SPICE simulations (approx 22 hours) on a host system using a pool

Table 4.2: Distribution of Local and Global Wires

	Local Routing	Global Routing
vga_lcd	42.2%	57.8%
des_perf	44.7%	55.3%
b19	45.5%	54.5%
s38584	47.3%	52.7%

Table 4.3: Three-pass test generation results for benchmark circuits

Circuit	Timing-Aware λ=0.2 δ=0.5			Pass 1 Path Delay			Pass 2 Timing-Aware δ=0.5			Pass 3 Trad. TDF		Final TDF	Final DTC	Final Pat.
	TCov.(TA)	DTC(TA)	Pat.(TA)	Paths	PCov.	Pat.	TCov.	DTC	Pat.	TCov.	Pat.	Cov.		
s13207	90.27	82.84	420	102	70.1	21	99.5	81.36	94	89.6	303	90.26	91.02	418
s35932	79.79	98.4	68	5	100	10	-	-	-	79.9	56	79.78	100	66
s38584	88.64	84.8	386	809	74.17	117	99.85	84.26	156	86.94	326	88.63	93.82	599
s38417	92.8	90.45	277	747	79.18	101	99.96	87.58	85	91.67	216	92.8	96.31	402
b15	91.27	84.2	1118	423	42.67	74	97.27	81.62	317	89.98	816	91.26	88.15	1207
b17	95.15	85.44	1262	1213	50.16	237	95.59	81.69	329	94.8	942	95.15	89.73	1508
b18	93.68	83.83	1457	2857	37.12	282	96.36	81.2	412	93	949	93.68	87.69	1643
b19	95.66	83.63	1560	5777	44.93	280	97.28	80.71	462	95.23	983	95.65	87.58	1725
ac97_ctrl	97.4	97.73	179	807	98.14	52	92	79.5	15	97.25	169	97.4	99.01	236
pci	95.74	88.91	677	2587	63.7	173	99.87	82.56	345	95.11	465	95.74	93.46	983
des_perf	97.05	97.19	231	1536	100	86	-	-	-	96.66	224	97.04	100	310
vga_lcd	93.73	87.58	5166	16547	19.43	234	99.93	90.21	4689	92.67	3165	93.72	91.42	8088

of state-of-the-art servers with at least fourteen processors available at all times with 8GB of memory and running Linux. For generating lookup tables, each standard cell requires 120 simulations to cover 60 cases. We used a commercial ATPG tool (FastScan [80]) to generate timing tests using launch-on-capture (LOC) and launch-on-shift (LOS). However, our proposed methodology is not limited to a specific ATPG for pattern generation.

4.5.2 Results

We present experimental results for the twelve benchmark circuits. The objective of the proposed methodology is not only limited to identify the high variability delay paths,

instead it is further extended to improve delay test coverage (DTC)[47] of complex designs. We used path delay fault (PDF)[40], timing-critical transition fault (TCF)[75] and traditional transition delay fault (TDF) based ATPG in three incremental modes to show the DTC improvement with high quality top-up patterns. The results obtained from the proposed methodology is finally compared with TCF based ATPG approach [75].

In our experiments, δ[75] is a fault dropping criterion named as Dropping based on Slack Margin (DSM) while λ[47] is a predefined limit whose value helps in determining timing-critical paths. Any path whose static timing delay is greater than T_{th}, in our case $0.7T_{sys}$, is considered as the input data set for the proposed algorithm. The test generation results are shown in Table 4.3. The results of timing-aware ATPG are shown in column *Timing-Aware (λ=0.2, δ=0.5)* with TDF coverage, DTC, and pattern count in sub-column *TCov.(TA)*, *DTC(TA)* and *Pat.(TA)* respectively. The target clock period $T_{tgt} = 0.8T_{sys}$ corresponds to $\lambda = 0.2$. *DTC is reported for only timing-critical transition faults as all faults are not sensitive to timing.* We performed the following experiments using **three pass method** for pattern generation:

Pass 1: The proposed algorithm is used to find all paths whose delay under process variation exceeds T_{tgt} ($0.8T_{sys}$). We used path delay fault (PDF) based ATPG to generate high quality robust tests for the identified paths. The timing-critical faults which lie on the paths covered by PDF based ATPG, accounts for 100% TDF and 100% DTC coverage. The results of PDF based testing is shown in column *Pass 1* of Table 4.3 presenting the total number of paths, path coverage and pattern count in sub-columns *Paths*, *PCov.* and *Pat.* respectively.

Pass 2: Unlike path delay testing, a timing-critical transition fault (TCF) may be testable even if the critical path on which it lies is untestable. In this incremental mode, a timing-aware ATPG is used to target TCF faults which are left uncovered in Pass 1. As lower DSM value can invoke deterministic test generator which may result in high pattern count, we used DSM criterion $\delta = 0.5$. The results of TCF based testing is shown in column *Pass 2* of Table 4.3 with sub-columns TDF coverage (*TCov.*), delay

Table 4.4: Increase in DTC and corresponding top-up patterns by applying different DSM criterion

	δ=0		δ=0.1		δ=0.5	
Benchmark	ΔDTC	ΔPat	ΔDTC	ΔPat	ΔDTC	ΔPat
s13207	8.7	-4	8.54	-3	8.18	-2
s35932	1.8	-2	1.8	-2	1.6	-2
s38584	8.86	208	8.77	205	9.02	213
s38417	5.81	119	5.55	128	5.86	125
b15	4.22	76	3.99	79	3.95	89
b17	4.4	229	4.43	230	4.29	246
b18	4.1	203	3.9	160	3.86	186
b19	3.97	125	3.93	153	3.95	165
ac97_ctrl	1.09	58	0.63	58	1.28	57
pci_bridge	4.75	284	4.92	304	4.55	306
des_perf	1.03	74	1.27	76	2.81	79
vga_lcd	2.81	3908	3.27	4476	3.84	2922

test coverage (*DTC*) and pattern count (*Pat.*).

Pass 3: The last incremental mode is targeting all transition faults which are not covered through *Pass 1* and *Pass 2*. The aim of this last mode (*Pass 3*) is to create traditional TDF patterns and match the TDF coverage reported by timing-aware ATPG (λ=0.2, δ=0.5). The final TDF coverage, delay test coverage, and pattern count are reported in column *Final TDF Cov.*, *Final DTC* and *Final Pat.* respectively. It can be seen that s35932 and des_perf shows 100% DTC as all identified paths are covered by PDF based ATPG. The increase in delay test coverage (ΔDTC) and pattern count (ΔPat) are computed as

- ΔDTC = *Final DTC - DTC(TA)*
- ΔPat = *Final Pat. - Pat.(TA)*

The ΔDTC and ΔPat using different DSM criteria are shown under the columns *δ=0, δ=0.1, and δ=0.5* of Table 4.4. It is observed that the proposed **three pass method** performs better while using DSM criterion δ=0.5. It offers an average increase in DTC and pattern count by 5.11% and 26.29% respectively. However, the corresponding percentage increase in DTC and pattern count for DSM criterion δ=0 are 4.96% and 26.71% respectively. This is expected as lower DSM criterion drops too many timing-critical transition faults before targeting them explicitly by ATPG [75]. The circuits s13207 and s35932 show an improvement of 8.18 and 1.6 in DTC at δ=0.5 without an increase in pattern count.

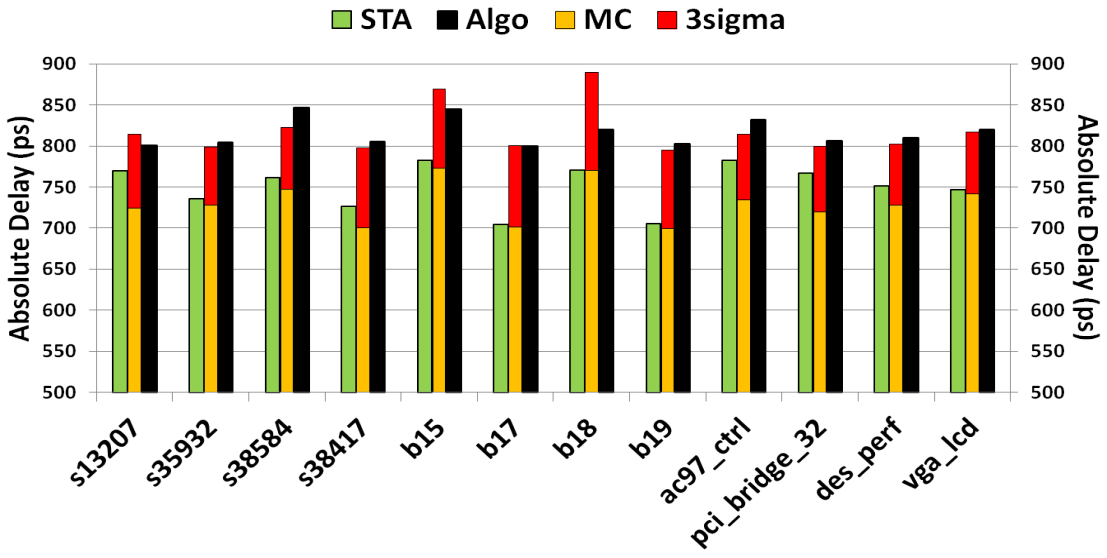


Figure 4.7: Timing delay predicted by STA, MC simulation and proposed algorithm on paths which offers worst delay variation

To investigate the effectiveness of the proposed algorithm in predicting timing delay under process variations, we performed experiments to compare our results with Monte-Carlo simulation and STA analysis as shown in Fig. 4.7. We selected a path from each circuit which showed the highest delay variability as predicted by ***Algorithm 1*** in subsection 4.4.2. The proposed algorithm (*Algo*) predicted timing delay of individual paths are compared with MC simulation (*MC + 3sigma*) and STA analysis (*STA*). *MC* (in Fig. 4.7) is the average value of 1000 Monte-Carlo simulations of each path

using *3sigma* as standard deviation. The algorithm predicted highest percentage delay variation is observed for circuit b18 (15.62%) while a lowest is observed for circuit s35932 (9.76%). The error in delay prediction by the proposed algorithm is in the range of ±10%. The only exception is circuit b18 whose prediction error with respect to MC simulation is 7.87%. This is due to the interconnect cross-talk effect which is not considered in the proposed work.

4.6 Summary

We have demonstrated in this chapter that long interconnections are more susceptible to significant process variations that can lead to timing failure. Such delays are not always reliably detected by traditional TDF timing tests. We presented a heuristic based path selection algorithm which uses the empirically developed models to predict 3σ variations and identifies the high variability delay paths. A "three pass" method is proposed which uses different ATPG techniques to improve delay test coverage and generates high quality top-patterns for identified critical paths. The experimental results showed that the proposed methodology detects a significant number of timing critical faults through longer paths which would have escaped using traditional TDF test.

— * — * —

Chapter 5

Aging Aware Path Delay Estimation

5.1 Introduction

Aging is a phenomenon of gradual circuit degradation over time which may results in performance limitation [3, 89, 90] when the circuit is operating under certain workload at specified environmental condition. Fortunately, the devices which fail early in life or during stable operating life follow a failure curve that is typically known as the *bathtub curve* [20]. The failure curve helps in predicting the life cycle of such devices by broadly categorizing it into three periods: early, stable, and wear-out. The bathtub curve can also be similarly categorized in three periods. It defines the allowed failure rate of acceptable product by defining the allowed failure rate over a defined period of life. Typically, one requirement is for the failure rate during *Early Life Failure Rate* (ELFR) period [21] and the other requirement is for the failure rate over the longer period of life known as *High Temperature Operating Life* (HTOL) period [22].

Before we review the other aspects of integrated-circuit aging mechanism and its modeling, it is necessary to discuss the most dominant reliability issues, such as Negative-Bias Temperature Instability (NBTI) and Hot Carrier Injection (HCI), observed in sub-32nm CMOS technologies. NBTI is the key reliability issue in PMOS-FETs when negative-bias

voltage is applied across the gate, whereas HCI is an issue in NMOS-FETs when gate voltage is comparable to drain voltage [16, 23, 24] at elevated temperature. Nevertheless, they remain a major reliability concern as DC supply voltage scaling is slowing down for sub-32nm technologies [16]. A geometrical unification of the theories of NBTI and HCI are well studied and elaborated for planar MOSFETs, as well as multi-gate field-effect transistors, e.g., MuFETs and FinFETs [25, 26].

Temporal reliability issues in the CMOS circuits do not manifest immediately after the chip production, whereas, spatial variations can be tested just after the fabrication. To guarantee the device operation over a defined period, usually the burn-in stress tests are utilized to age the circuits in shortest duration of time without waiting for years to see final aging effects. Identifying a set of long paths, that potentially can fail or degrade the circuit performance during HTOL period, in stipulated design cycle time is a real challenge for system-on-chip (SoC) designers. The temporal reliability issues in CMOS circuit has attracted many researchers and engineers to develop more accurate models of BTI and HCI to predict failure rate of various circuits operating under certain workload. Most of the previous work [23, 25, 28, 29, 91] either aims at developing an accurate model to fix key reliability issues or discusses the impact of aging on circuit level. The following procedures are available in the literature that may help in identifying long paths:

1. Static Timing Analysis (STA) [52] identifies top critical paths under specified user-defined setup slack condition for path criticality with respect to target clock period.

2. Dynamic timing simulation (SPICE level simulator) [50, 51] accurately estimates timing delays of various paths and a user-defined timing critical condition helps in isolating a set of speed-limiting paths.

3. Statistical Static Timing Analysis (SSTA) [53] identifies the speed-limiting paths based on the target yield. SSTA represents the delay in terms of path distribution function and invariably uses SPICE models to generate industry's first open standard effective current source model (ECSM [92]).

However, all of the above procedures have limitation either in terms of path delay estimation accuracy or overall runtime. STA is faster than any of the timing-analysis methods available in the literature. The only limitation is the pessimism in estimating the timing delay due to which, it is not easy to identify a realistic set of speed-limiting paths. The pessimism in delay calculation is due to the fact that it uses abstract level models which provide pessimistic delays of gates and interconnects [83]. In order to time circuit at worst corner, *STA uses the extreme values of all physical parameters related to MOS transistors and interconnects along with worst environmental conditions which are unlikely to happen at the same time.* That is why, when the parametric testing is performed, it is not rare to observe that most of the devices perform better than expected in terms of operating frequency. SPICE simulations are reasonably accurate in estimating path delay with respect to delay measured on actual silicon. The Only drawback is the simulation runtime of all paths in a circuit. It may work for smaller circuit, however, simulation run time grows exponentially with design size.

This chapter first *evaluates the applicability of NBTI and HCI geometry-oriented unified theory, that uses reaction-diffusion (R-D) model [93], in estimating threshold voltage (V_{th}) degradation at specified technology node and correlates the theoretically obtained V_{th} degradation with real silicon data of 28nm planar CMOS technology.* Secondly, we obtain a *realistic set of physical and non-physical parameters that influence the device performance at system level.* The identified parameters are used to generate the two dimensional (2D) look-up table (LUTs) for each standard cell of the library. Lastly, we explore an opportunity to use the proposed algorithm (that uses LUT of each standard cell) in estimating realistic worst-case path delay under 10 years of aging. In experimental results, we have shown that the proposed methodology outperforms SPICE simulation and STA in terms of time complexity and timing-delay estimation accuracy respectively. *The critical paths identified using the proposed approach can be used to develop the functional or structural-scan based test patterns to improve small-delay-defect (SDD[77]) coverage or it can be periodically monitored for aging analysis under real workload.*

Section 5.2 discusses the prior work published in the literature that broadly covers the

impact of aging at device and system level. It is followed by the summary of the work that covers the novelty of our approach. Section 5.3 discusses the framework which accepts the unified model of NBTI and HCI presented by Kufluoglu et al. [26] to intelligently choose a set of parameters that influence the circuit performance under aging. Additionally, our single PVT corner based methodology makes valuable contributions when the number of process corners (at which STA has been signed-off for design tape-out) grows exponentially with the introduction of very deep sub-micron technologies. Section 5.4 formally defines a segment and its modeling that help in generating 2-D look-up tables (LUTs) for each standard cell present in the library. Section 5.5 outlines an efficient, simple and faster algorithm that estimates the timing delay of paths in close proximity of SPICE simulated results within an acceptable delay estimation inaccuracy of 5.5%. Section 5.6 measures the time efficiency of the proposed algorithm and compares the timing delay estimation accuracy with SPICE simulated results. Section 5.7 concludes the paper with future work.

5.2 Prior Work and Key Contributions

It is desired to fabricate the integrated circuits such that they continue to meet system level specifications over time, and fairly important to address the temporal reliability issues [16] in deep sub-micron (DSM) CMOS technologies. In the past 4 to 5 decades, many researchers and reliability engineers have made efforts to understand and solve the integrated circuit aging phenomenon. A slower voltage scaling in CMOS technology nodes has emerged as a main source of aging-effects in planar and 3-D FinFETs [16]. A comparatively higher operating voltage and shorter gate length give rise to hot carrier effects such as HCI, which remain a major reliability concern for FinFET devices [25]. NBTI and HCI are fundamentally different from each other, however, both involve in breaking Si-H bond at oxide interface followed by H_2 diffusion into oxide [26, 28]. Both these reliability issues can be modeled using reaction-diffusion (R-D) [26] and lucky electron model [16]. R-D model uses threshold voltage shift to account for BTI and HCI

effects while lucky-electron model uses peak substrate current to measure the device degradation.

In a recent study [94], it is found that NBTI degradation in planar transistor is more severe in comparison to multi-gate FinFET when both transistors are built on 16nm technology node. In an another work [95], FinFET devices exhibit worst NBTI than planar devices on wafers with standard orientation, due to the higher availability of Si ends at the (110) oriented fin sidewalls. *NBTI will remain a key reliability issue due to higher hole concentration at the interface between the nitride gate oxide and the Si-Fin surface.* A broad investigation on reliability characteristics, in NMOS and PMOS FinFETs, is performed in [23] and evaluated on 3-D FinFET fabricated on traditional ULSI logic platform. An excellent PMOS NBTI lifetime has been predicted for the device with wider fin width, and degradation is expected by scaling down the fin width. Whereas interface state generation in an ultrathin nitrided gate oxide ($14\mathring{A}$), self-heating effect, and temperature dependent bandgap energy, play a significant role in hot-carrier degradation.

In a recent work [3], an approach is proposed to identify a set of long paths that may become speed-limiting on silicon over time. The authors have reported an inaccuracy of approx 10%, in estimating worst-case path delays as compared to SPICE estimated delays under temporal variations. For complex designs, a 10% inaccuracy in estimating timing delays may qualify thousands of more paths to be considered for delay estimation. Due to this reason, the authors of [3] concluded that their approach is useful for small to mid-sized circuits only. The current work addresses the drawbacks of [3] by reducing the inaccuracies in estimating path delays under 10 years of operational aging and at the same time, must be scalable for complex designs fabricated at different technology nodes. The following changes are proposed to improve the path delay prediction accuracy and improving the overall methodology of [3] to address complex designs,

(a) Revisit segment model presented in [3, 5] by replacing the worst-case distributed RC T-network with performance driven balance RC π-network [52, 96], which can be optimized for standard fan-outs. This is a faster high-level approach for estimating

path delays in a circuit and has very good accuracy.

(b) Adopts a unified aging model [25, 28] of NBTI and HCI degradation in terms of threshold voltage (V_{th}) shift over device lifetime. The V_{th} shift and empirically chosen realistic worst-case PVT corner avoids multi-corner delay estimation requirement.

The authors of [26] modified the 30 year old R-D classical models of NBTI and HCI by re-evaluating and establishing a new geometrical relation between them. They explained NBTI and HCI using 1-D and 2-D diffusion equations. Their research outcome has enabled the authors of [25] to re-establish the equations for 3-D FinFET structures. However, the authors of [25, 26] are not alone who explained the performance degradation (in terms of V_{th} shift) due to NBTI and HCI. As digital circuits may fail either due to spatial unreliability (related to process variations) or due to temporal unreliability (aging, electromagnetic induction etc), the authors of [29] have proposed a variation-aware aging analysis to estimate the overall performance degradation. Their proposed model is not straightforward and difficult to use at system level. Apart from this, a work by Ma et al. [97] is introduced in 2013 that focused on developing a compact reliability model for channel hot electron (CHE) and NBTI aging effects. The only drawback of their standard solution is that it is not scalable to 3-D FinFET structures.

Our primary objective is to explore the opportunity to use the proposed methodology, instead of using pessimistic STA or time-consuming SPICE for delay calculations at worst PVT corner, to identify a set of realistic long paths that may become critical during device life-time. This chapter makes the following contributions,

1. Predicts the maximum V_{th} degradation, due to NBTI and HCI, that can be observed at specified process-voltage-temperature (PVT) corner. Theoretically estimated V_{th} degradation based on reaction-diffusion (R-D) model, presented in [28], is empirically compared with the results obtained from 1000 industrial production parts fabricated on 28nm planar CMOS technology node.

2. Identifies the worst-case PVT corner that shows maximum temporal variations due

to predicted V_{th} degradation.

3. Proposes an algorithm which is substantially faster than SPICE simulators [50, 51] with an approx. 94% accuracy in estimating path delays at specified PVT corner.

4. Develops a methodology to identify a set of long paths that may fail or degrade the circuit performance over time. The presented methodology uses the proposed algorithm in isolating such speed-limiting paths that can be used to improve SDD coverage. Alternatively, it can be used to periodically monitor the aging impact.

It is worth to point-out that *our aim is neither to replace industry standard multi-corner pessimistic STA analysis to sign-off circuit database for fabrication nor to develop the standalone aging-aware timing library.* As the outcome of the proposed methodology and its sustainability are dependent on delay estimation by SPICE and STA based approaches, it cannot be adopted as independent delay estimation technique. We intend to offset the pessimism of STA in estimating timing delays by adopting the proposed algorithm (discussed in Section 5.5) that uses reasonably accurate look-up-tables (LUTs) as discussed in Section 5.4. We intelligently identify a set of parameters that has major impact on the circuit performance and use them radically to identify a set of long paths that may become critical over time. The identified paths can be used for *a)* periodically monitoring the change in timing delay, using functional tests, to measure the impact of aging over time, *b)* running dedicated scan-based tests using Logic BIST every time the chip boots, and *c)* running ad-hoc test to improve test quality after production.

Furthermore, the proposed methodology can be used in identifying fast-aging paths and generation of path delay distributions using clock sweeping techniques [98]. In [99–101], the path delay distributions of fast-aging paths are used as fingerprints to distinguish recovered integrated circuits (ICs which are used parts and being sold as new) from a batch of fresh integrated circuits. Usually, the aging effect (due to BTI or HCI) translates into threshold voltage V_{th} shift that varies with different technology nodes. The path-delay fingerprinting technique [99–101] has a drawback of assuming

the static V_{th} degradation of 5%, which is non-realistic and non-scalable to deep sub-micron technology nodes. In this work, we have introduced a scalable methodology to estimate V_{th} shift on a specified technology node under *weibull statistical analysis* [27] based workload. The proposed work empirically correlates the estimated frequency degradation (due to V_{th} shift) with statistical data obtained from 1000 industrial ICs. In the experimental section, the basis of using SPICE simulated result as benchmark lies in the fact that the foundry provides SPICE models to measure the impact of spatial and temporal variations in path delay estimation.

5.3 Parameters that Impact Circuit Performance under Aging

This section discusses the physical and non-physical circuit parameters that must be chosen rationally such that it result in worst aging effects on circuit performance. This section also explores the viability of establishing a relationship between theoretically estimated and experimentally obtained V_{th} shift. In an earlier work [3], the parameters that influence the circuit performance under aging are defined completely on the basis of empirical data obtained on specified CMOS technology node. This makes the methodology, presented in [3], less adaptive in estimating V_{th} degradation at different CMOS technology nodes. To address one of the drawbacks of [3], it is required that unified-aging model of NBTI and HCI [28] to be used in a reusable framework to estimate performance degradation of circuits in terms of threshold voltage V_{th} shift.

5.3.1 Theoretical framework of V_{th} degradation under Aging

Before we further explain the temporal reliability issues at device level, it is important to mention that the research outcome of [28] is used which seems to be computationally efficient in estimating the V_{th} shift with an acceptable accuracy in comparison to [29, 97]. As mentioned earlier, NBTI and HCI are involved in Si-H bond breaking in oxide with

holes and electrons displace hydrogen away from Si-oxide interface respectively. In NBTI,

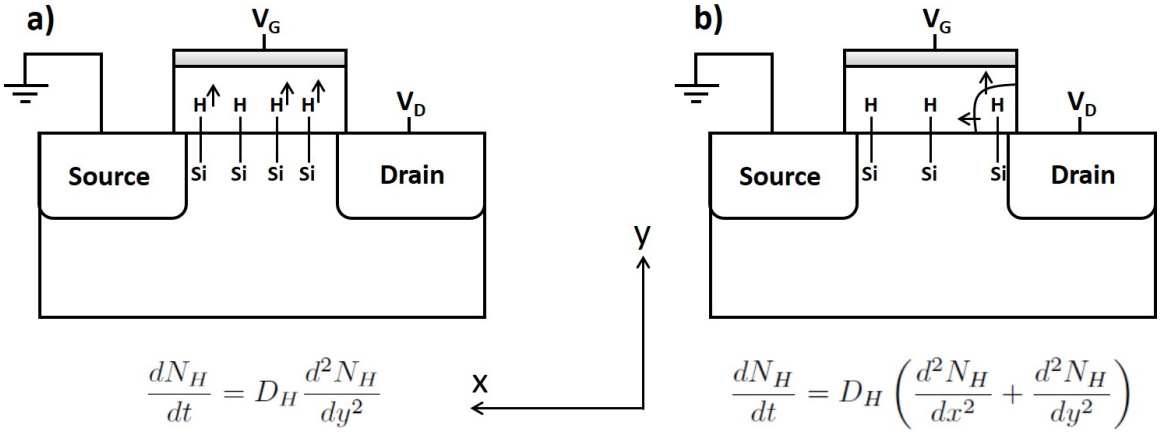


Figure 5.1: Hydrogen diffusion into the oxide due to a) NBTI, 1-D diffusion b) HCI, 2-D diffusion at drain end

holes in inversion layer tunnel into oxide that results in breaking of Si-H bond. The released hydrogen diffuses away from $Si/Si0_2$ interface and results in trap generation for holes. The holes are much heavier than electron, hence they participate in tunneling rather than being "Hot" due to lateral electric field. Fig. 5.1a shows the NBTI diffusion occurs in one dimension (1-D) governed by the diffusion equation, where Y-direction is from silicon substrate to oxide while X-direction is along the channel away from drain [26]. N_H and D_H are the hydrogen density at $Si/Si0_2$ interface at any time t and hydrogen diffusion constant in oxide respectively. During hot carrier injection, as shown in Fig. 5.1b, electrons get sufficient energy due to lateral electric field so that they penetrate through oxide layer and displace hydrogen away from $Si/Si0_2$ interface. As a result of this displacement, interface charges get induced leading to threshold voltage V_{th} increase. At any time t, the interface charge density generated due to NBTI [25] can be given as:

$$N_{IT,NBTI}(t) = \sqrt{\frac{K_f N_o}{2K_r}} (D_H t)^{1/4} \tag{5.1}$$

Similarly, the charge density generated due to HCI in 2-D is given as [25]:

$$N_{IT,HCI}(t) = \sqrt{\frac{\Pi K_f N_o}{12 K_r}} (D_H t)^{1/2} \tag{5.2}$$

where D_H is hydrogen diffusion constant, K_f and K_r are forward dissociation and reverse annealing rate of Si-H bonds respectively, and N_o is initial concentration of Si-H bonds. The detailed explanation related to above equations can be seen in [25, 26]. The change in threshold voltage due to interface charge densities, using Eqn. 5.1 and 5.2, can be written as,

$$\Delta V_{th}(t) = \frac{qN_{IT}(t)}{C_{ox}} = \frac{qt_{ox}(N_{IT,NBTI}(t) + N_{IT,HCI}(t))}{\epsilon_{ox} W_g L_g} \tag{5.3}$$

where t_{ox} is the gate oxide thickness, ϵ_{ox} is the oxide dielectric constant, W_g and L_g are gate width and length respectively. Eqn. 5.3 shows the threshold voltage shift dependence on interface charge density due to NBTI and HCI over time. In order to find

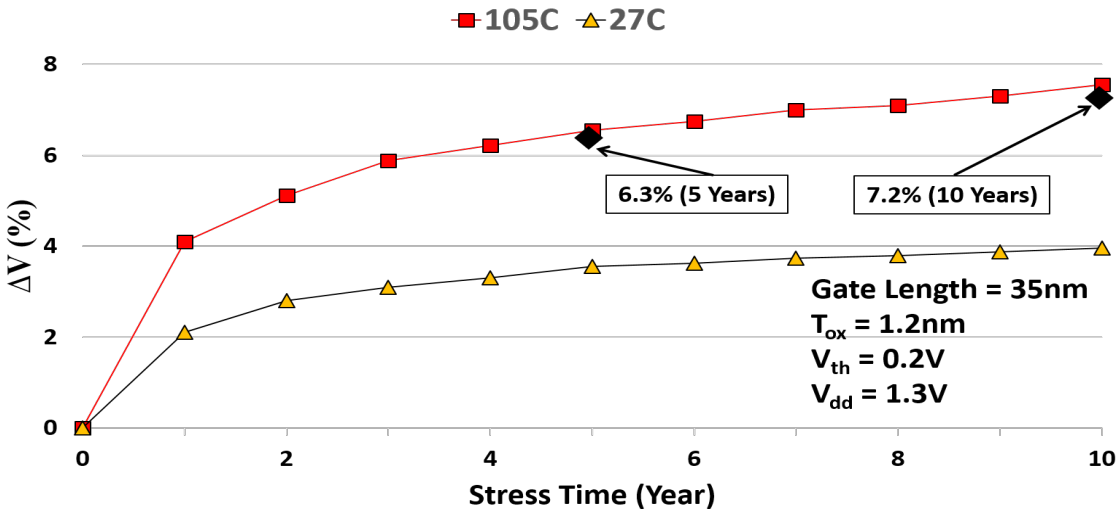


Figure 5.2: Percentage change in threshold voltage V_{th} as a function of time at different temperature

the absolute value of ΔV_{th} as a function of time (in sec), first we have to identify the parameters whose values can be found directly from 28nm standard V_{th} cell based high performance library. The parameters whose values are obtained by another experimental

setup are D_H, k_f and k_r. These are temperature dependent parameters, governed by *Arrhenius Equation*, which depends on activation energies, E_H, E_f and E_r respectively. The activation energies are used from the experiments performed in [95, 102, 103].

Fig. 5.2 shows the percentage change in threshold voltage (with respect to nominal V_{th} of standard cell without aging) as a function of applied stress over time (in years). ***An approx 7.5% and 4% threshold voltage degradation are observed at*** 105^oC ***and*** 27^oC ***respectively after 10 years of device operation***. The device parameters are shown in Fig. 5.2 whereas power supply is accelerated to 1.3 times the nominal DC supply voltage which is 1V in our case. It can also be seen that initial rate of V_{th} degradation is comparatively high and stabilizes after some time. This is due to the power-law of hydrogen diffusion rate that finally saturates over time.

5.3.2 Empirical framework of V_{th} degradation under Aging

In order to estimate the reliability of integrated circuits (IC), it is highly desirable to perform burn-in stress test at accelerated voltage and temperature. Burn-in is an important reliability test of ICs to ensure a guaranteed performance till the end of circuit's life-time at customers end. As burn-in test time is usually in hours which is expensive, substantial work has been published to reduce burn-in time through high voltage stress test [104, 105] and weibull statistical analysis [27]. We have used *weibull statistical analysis* based workload during stress test. However, the workload may be different in real applications, and cover all of them is out of scope of this work. We can guarantee for those paths only which are covered under weibull defined workload. In this work, we first perform functional or structural scan-test to determine highest frequency obtained (F_{max}) on packaged parts just after the production and then we perform burn-in stress test on the same set of parts. Once burn-in is over, we again collect the F_{max} data with same set of patterns and circuits. This helps in estimating the absolute F_{max} degradation that can be further translated into absolute V_{th} degradation.

We have performed burn-in stress test on 500 typical and 500 slow parts chosen randomly from different wafer lots. During stress-test, the DC power supply is scaled to

1.3 times the nominal voltage, which is 1.3x1V in our case, and temperature is increased up-to 105^{o}C. The voltage and temperature acceleration must be identified carefully as they directly impact the device manufacturability of final production lots. During burn-in, all clocks are adjusted to run at 10MHz to ensure that the power dissipation during test mode does not cross the maximum power limit. In order to avoid unwarranted static stress in the device [89], it is fairly important to toggle each circuit node and maintain the clock duty cycle at 50%. More than 95% toggling activity is achieved (dynamic stress) for 48 hours of stress test to cover 10 years of circuit operation. The following procedure is used to estimate F_{max} degradation on a device:

(a) Identify a set of functional or scan patterns that results in identifying F_{max} of target device. Lets assume that F_{max}^{T0} is the maximum frequency observed on a device before applying burn-in test.

(b) Characterization of F_{max} degradation is done at four DC supply voltages (V_{max}, V_{nom}, V_{min}, V_{xmin}) and each voltage node has four temperature corners (-40^oC, -5^oC, 27^{o}C, 105^{o}C). Finally we have to collect F_{max} at 16 different combinations of voltage and temperature which can be written as F_{max}^{T0} (V,T) where V $\in$ (1.1V, 1V, 0.9V, 0.72V) and T $\in$ (-40^oC, -5^oC, 27^{o}C, 105^{o}C).

(c) Put the device into burn-in oven/chamber. Increase the supply voltage and temperature as discussed earlier to promote NBTI and HCI effects on circuit. To cover 5 and 10 years of operation, stress tests are conducted for 30 and 48 hours respectively.

(d) Run the same set of patterns used to observe F_{max}^{T0} (V,T). Now collect F_{max} data again at identified 16 corners (of voltage and temperature) after 30 and 48 hours of stress-test respectively. The notations of 30 and 48 hours of burn-in can be written as F_{max}^{T30} (V,T) and F_{max}^{T48} (V,T).

(e) Now calculate the F_{max} degradation at specified voltage and temperature as,

$$F_{dg}^{T30}(V,T) = \frac{|F_{max}^{T30}(V,T) - F_{max}^{T0}(V,T)|}{F_{max}^{T0}(V,T)} \times 100 \tag{5.4}$$

where $V \in$ (1.1V, 1V, 0.9V, 0.72V) and $T \in (-40^o\text{C}, -5^o\text{C}, 27^o\text{C}, 105^o\text{C})$.

(f) Eqn. 5.4 shows an absolute frequency degradation (in %) after 30 hours of burn-in. The set of data can be collected for 48 hours of burn-in as well.

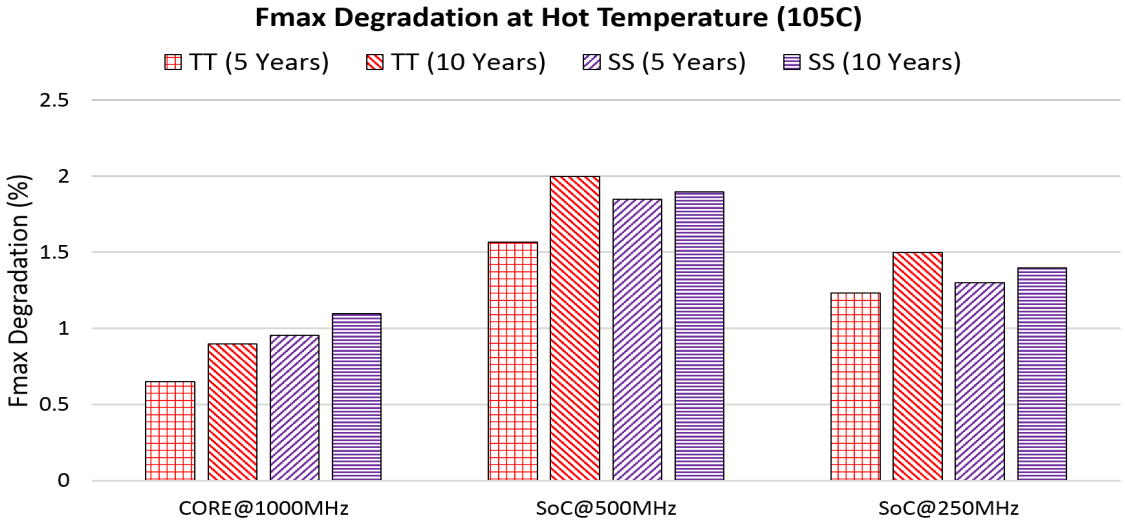


Figure 5.3: F_{max} degradation observed on typical and slow devices at 105^oC

Instead of showing all collected F_{max} data over the experiment, Fig. 5.3 and 5.4 explicitly compare the average F_{max} degradation (in %) observed on typical and slow devices at hot (105^oC) and cold (-40^oC) temperatures for 5 and 10 years of device operation respectively. As we are looking for worst-case scenarios over longer duration of time, the circuits fabricated at best corners are not qualified for F_{max} degradation analysis.

Before we start discussing the outcome of our experiments, it is necessary to mention that different set of patterns are identified to target logic timed at three different frequencies on the same device. The experiments are performed on a system-on-chip (SoC) which consists of a CORE and other related logic. CORE logic is signed-off to work at 1000MHz while rest of the logic is sub-divided into 500MHz and 250MHz respectively. It

can be seen as *CORE@1000MHz, SoC@500MHz* and *SoC@250MHz* in Fig. 5.3 and 5.4 respectively. At hot temperature (Fig. 5.3), typical parts show highest F_{max} degradation for atleast 500MHz and 250MHz domain.

For sub-28nm technologies, low V_{th} standard cells based library show maximum delays at cold rather than hot due to temperature inversion effect [106, 107]. The *temperature inversion* is sometimes referred to as *reverse temperature dependence*, when drain current (I_D) increases with increasing temperature. Such behavior is shown by low V_{th} standard cells, of sub-32nm CMOS technology nodes, that have relatively smaller gate overdrives. On the contrary, high V_{th} standard cells that usually have large gate overdrive follow *normal temperature dependence* where drain current (I_D) decreases with increasing temperature. However, nominal V_{th} standard cells can follow either reverse or normal temperature dependence for drain current and hence, it is difficult to estimate the PVT corner at which they show the worst delays.

The current production parts under discussion consist of low V_{th} standard cells and expected to show the worst delay at lower temperature. This might be a reason that the paths which are speed-limiting at 500MHz and 250MHz (at hot temperature) are not gate dominant, rather they are interconnect dominant. However, the F_{max} limiter paths in CORE still belong to slow corner not the typical one. The experimental results points to a interesting fact that speed-limiting paths for 500MHz and 250MHz frequency domain are interconnect dominant at 105^oC while critical paths of 1000MHz are still gate dominant. Fig. 5.4 is similar to Fig. 5.3 except that the data are collected at cold temperature (-40^oC). It clearly indicates the fact that the real speed-limiting paths exacerbated at cold and F_{max} degradation analysis must be focused at this corner. The results also give a clear indication that the speed-limiting paths at hot and cold did not have much correlation for 500MHz and 250MHz frequency domain. In conclusion, highest F_{max} degradation is seen on circuits fabricated on slow-slow (SS) process corner and tested at -40^oC at 0.9V DC power supply. ***Their absolute F_{max} degradations are 3.72% and 4.3% for 5 and 10 years of operating life***.

The next objective is to translate the absolute F_{max} degradation to threshold voltage

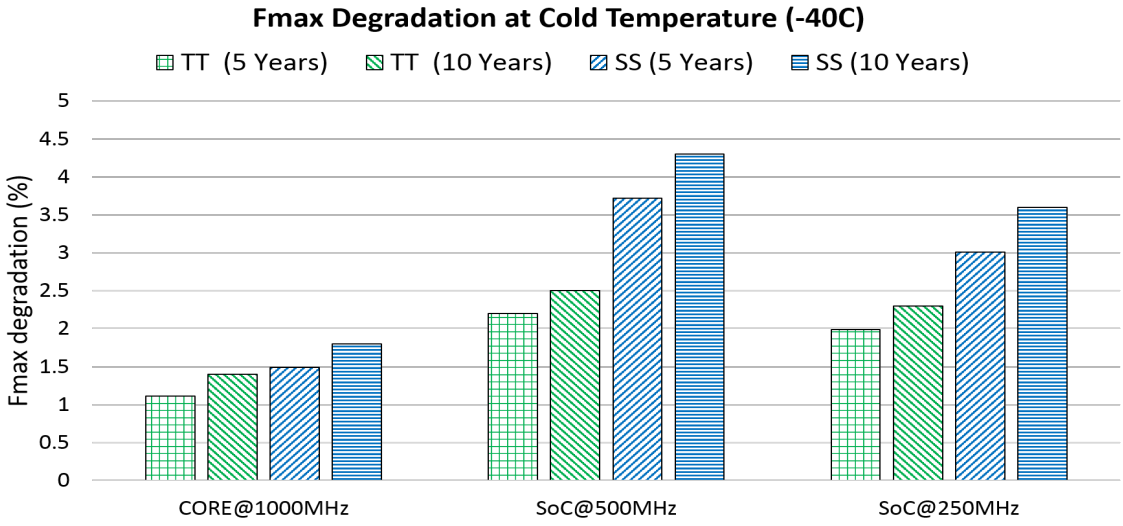


Figure 5.4: F_{max} degradation observed on typical and slow devices at -40^{o}C

V_{th} degradation. This is not straightforward to establish a one to one theoretical relation as the obtained F_{max} degradation does not belong to a single gate. However, we have used a 2-input NAND gate driving the maximum load (lumped resistance and capacitance) as defined in standard cell library to perform SPICE analysis and build a typical relation between F_{max} and V_{th} degradation. In order to fulfill the objective of identifying maximum V_{th} shift, the absolute values of critical load resistance and capacitance [108] are used that are driven by standard driver gates. A relation between F_{max} degradation and threshold voltage degradation is established in Fig. 5.5 where both degradations are clearly marked. It is worth to mention that the highest absolute V_{th} degradation (due to NBTI and HCI) is expected at high temperature and voltage. However, for low V_{th} standard cells with modest V_{th} degradation at low temperature translate to much higher performance penalty compare to the V_{th} degradation at high temperature and voltage. Such observations are explained earlier as reverse temperature dependence [106, 107] of drain current. It must not be confused with the absolute values of V_{th} degradation as it is being calculated very similar to F_{max} degradation of Eqn. 5.4.

In order to cover all paths that can escape our analysis using the proposed methodology, the relation between V_{th} and F_{max} degradation as shown in Fig. 5.5 is plotted

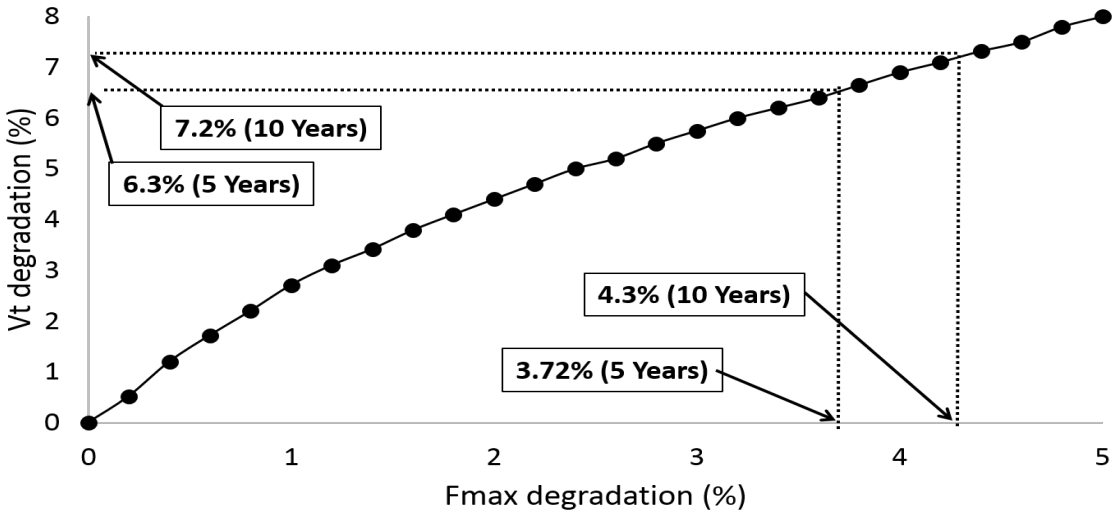


Figure 5.5: A typical relation between F_{max} degradation and V_{th} degradation

at high temperature. The worst F_{max} (or performance) degradation obtained at cold temperature is mapped to V_{th} shift at hot. ***The F_{max} degradation of 3.72% and 4.3%, as predicted from several experiments, translate to V_{th} degradation of 6.3% and 7.2% respectively***. Now locate the V_{th} degradation as obtained from Fig. 5.5 on scatter chart of Fig. 5.2 in Sub-section 5.3.1. The V_{th} degradation obtained from the experiments are labeled as diamond shaped markers at 5 and 10 years of device operation.

The percentage difference between the V_{th} degradation obtained from the set of experiments and as estimated from the theoretical work of [28] is less than 0.3%. This gives us the confidence to use the outcome of Kufluoglu et al. [28] work in our proposed methodology. *It is fair to conclude a worst-case threshold voltage degradation of 7.5% while performing aging analysis on 28nm planar transistors.* This value can be scaled depending upon the chosen technology nodes and type of standard cells used.

5.3.3 External parameters that Impact Aging analysis

In this sub-section, we focused on external parameters that must be taken into account when the effects of temporal variations are worst. In last Sub-section, we collected F_{max}

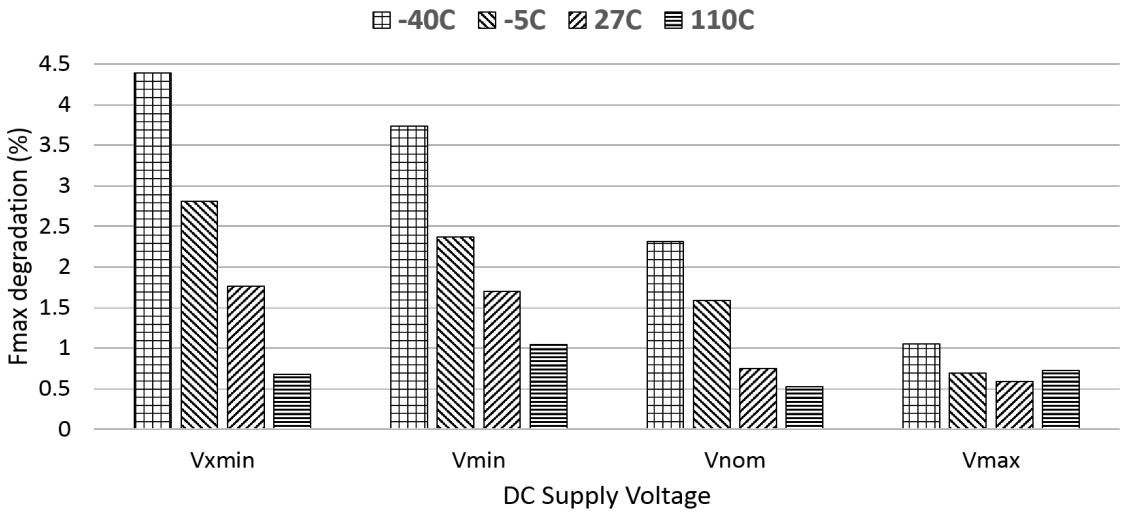


Figure 5.6: Fmax degradation observed at different voltages and temperatures on 500 slow parts after performing stress-test

data at different combinations of voltages V_{max} (1.1V), V_{nom} (1V), V_{min} (0.9V), V_{xmin} (0.72V) and temperatures -40^oC, -5^oC, 27^oC, and 105^oC respectively. As discussed previously, the circuits fabricated on slow-slow corner have shown the worst performance degradation and hence, Fig. 5.6 considers the experimental data of slow parts only. The clustered column chart of Fig. 5.6 is prepared by considering the average values obtained from each device at specified voltage and temperature. The figure clearly outlines the fact that worst-case frequency degradation is seen at cold temperature and must be considered for aging analysis. However, if the high V_{th} standard cells are used while synthesizing the design then worst frequency degradation is expected at high temperature. Furthermore, instead of using V_{xmin} (at cold) as worst-case scenario, we have used V_{min} for worst-case aging analysis. This is due to the fact the a very low DC supply voltage will be unable to promote hot electron effects, time dependent dielectric breakdown etc. *At the end of this section, we are able to identify the worst-case threshold voltage degradation, DC supply voltage and temperature at which, we can perform a realistic aging analysis.*

5.4 Segment Definition, Modeling and Look-up Table generation

In the previous section, we have discussed a framework that must be used to estimate maximum V_{th} degradation expected at a specified technology node and worst PVT corner to be considered to evaluate aging effects. This section broadly covers: *a)* The definition of smallest entity of a circuit, named as *Segment*, which are clustered together to form a target path, *b)* introducing a segment delay model that consists of a driver gate and interconnects, *c)* generation of two dimensional (2-D) look-up table (LUT) for each standard cell in the library by using segment delay model.

5.4.1 Segment Definition and its Delay Model

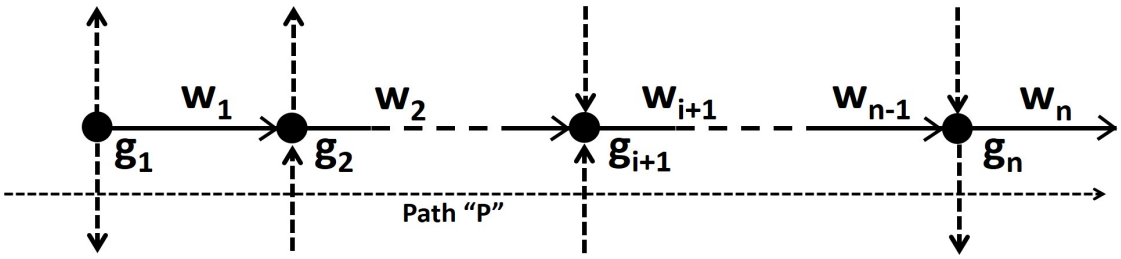


Figure 5.7: Path definition using segments

Definition: A Segment is defined as the smallest entity of a circuit that can be used to define a specified target path. It consists of a driver gate and a load, where load can be modeled as lumped resistance and capacitance or distributed RC tree respectively.

For example, let's assume a path "P" as shown in Fig. 5.7 where each node is a driver gate (g_1-g_n) passing through the interconnects w_1 to w_n. The lines with respective arrows which are going into or out-of the nodes are fan-in or fanout of a specified node, i.e., gate g1 has fan-out of 3 and g2 has fan-out of 2. A path "P" is defined as an ordered set of gates and interconnects as g_1, w_1, g_2, w_2,....g_n, w_n. Hence, any path in a circuit can be defined using ordered set of segments. In this work, a path "P" includes all

path between primary inputs to registers, registers to registers, and registers to primary outputs. Now, by using the basic definition of *Segment*, we can define a path P as $\{s_1, s_2, ...s_n\}$ where segment s_1 consists of g_1 and w_1, s_2 consists of g_2 and w_2, and so on.

It is important to accurately model a segment that may be the part of multiple-paths and helps in estimating the worst-case path delay that must be in close proximity of SPICE simulated results. Fig. 5.8 shows a topology that is a hybrid approach of Kahng

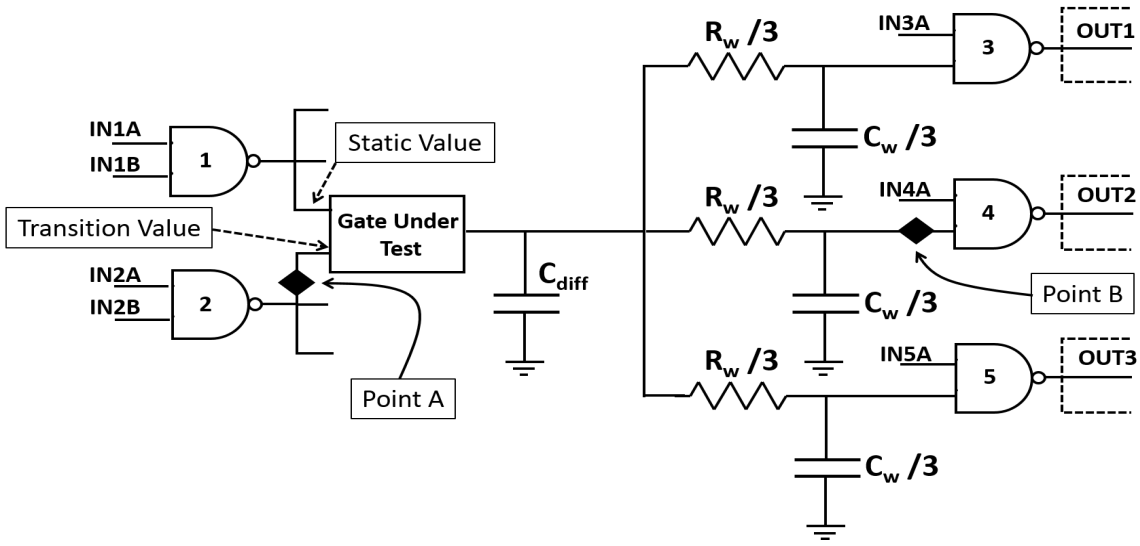


Figure 5.8: Segment Model - driver gate and interconnects as load

et al. [96] and Lopez et al. [87] to offset the pessimism of [3, 5]. The merits of using the proposed model of [96] are: a) use of lumped resistance and capacitance for intermediate and long interconnects, b) reasonable accuracy for routing optimized circuit layouts, c) faster in estimating the interconnect delays at system level. An extensive work on identifying critical paths of the microprocessors has been done in [86, 87]. Their research outcome suggests that a critical path can be modeled as a chain of two-input NAND gates with each having a fanout of 3. In this work, *we have merged the merits of both approaches [87, 96] to define a scalable and fairly accurate segment delay model.*

In the presented segment model, as depicted in Fig. 5.8, we have shown the circuit topology of two-input gate-under-test (GUT) only. However, a similar topology can be used for three or more inputs of the standard cells. As we know that CMOS circuits are

highly sensitive to input arrival time, so we have targeted a single input pattern that can create worst-case delay across segment (driver gate and interconnect). One input of NAND gate remains at non-controlling static value while the other input is used to create transition, and input selection is made on the basis of standard cell library timing characteristics. It can also be seen that propagation delay (rising and falling) of segment would be measured from *PointA* to *PointB* which is clearly shown in Fig. 5.8. CMOS circuits are also sensitive to input transition time (0 to 50% of input voltage) and to avoid additional complexities, we have used a typical value of transition time which is approx 110 ps as suggested in data book of 28nm library.

5.4.2 Look-up table (LUT) generation for standard cells

Before we start discussion about the adopted procedure for look-up table (LUT) generation, it is important to understand the role of LUTs while estimating the path delays under aging. Consider a normal distribution curve of Fig. 5.9 that shows a target path delay distribution over the arrival time. The Y-axis is the fractional number of chips and X-axis shows the path arrival time of a target path. As the target path shows dif-

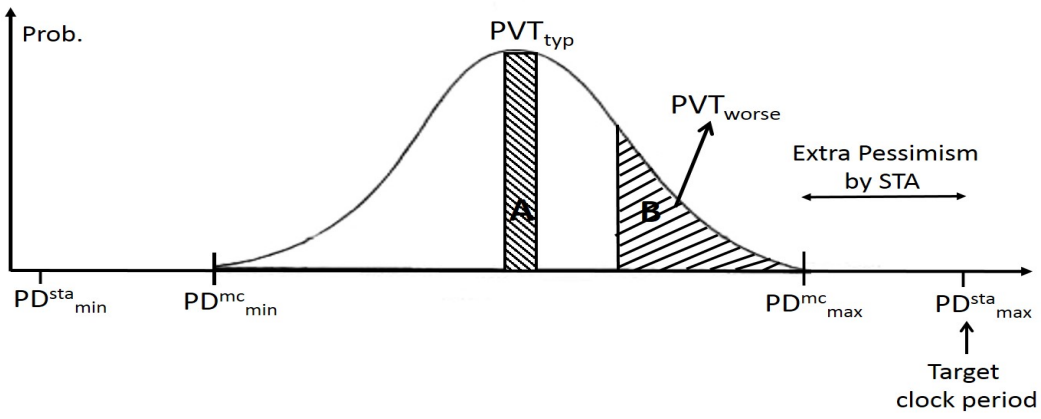


Figure 5.9: Path distribution as a function of arrival time for a target path

ferent delays on different chips of the same wafer due to spatial or temporal variations, we have used the probability density function to explain it. PD^{sta}_{max} and PD^{sta}_{min} are the worst-case and the best-case delays as estimated by STA to fix setup and hold violations. On the other hand, PD^{mc}_{max} and PD^{mc}_{min} are the worst-case and the best-case path delays estimated using SPICE simulator and are supposed to be in proximity of real silicon behavior. The absolute difference in PD^{sta}_{max} and PD^{mc}_{max} can be formally defined as worst-case margin which is used frequently in industrial designs. It is worth to mention that an intelligently chosen STA margin, backed by statistical data, helps in enhancing the device performance along with improvements in overall power (static and dynamic both) and silicon area [109].

In this work, long paths are identified under shaded area marked as B in Fig. 5.9 which corresponds to worst-case corner, PVT_{worst}. The only demerit of STA estimated path delays is, they are over-estimated with respect to target clock period. In order to offset the pessimism, we intend to identify a procedure that helps in estimating path delays in close proximity of SPICE simulated results. The shaded area, marked as A, under the curve of Fig. 5.9 is a set of those paths which have not shown much variability and are close to SPICE simulated results at typical corner, PVT_{typ}. As discussed earlier, a target path is defined as an ordered set of segments and if the segment variations are modeled correctly, we can roughly estimate the worst-case variations for the path as well.

Delay multiplication factor is introduced whose absolute value is used to scale the typical segment delay to worst-case segment delay and its derived value is more realistic in comparison to STA. As shown in Fig. 5.8, the lumped resistance (R_w) and capacitance (C_w) are transformed into different realistic wire loads as W1(20Ω, 0.8fF), W2(40Ω, 1.6fF), W3(80Ω, 3.2fF) progressively increasing upto W30(1160Ω, 46.4fF). Each wire load is assumed to be routed through intermediate metal layers such that it does not exceed maximum wire length of 217 μm. The maximum limit on wire load is purely driven by the standard library built on a specified CMOS technology node. The absolute values of various parameters that corresponds to PVT_{typ} and PVT_{worst} are shown in Table 5.1. PVT_{typ} is a typical-case corner which corresponds to middle of the bell-shaped curve

Table 5.1: 2-D Example LUT for two input NAND gate

	Process	DC Supply Voltage	Temp.	V_{th}
PVT_{typ}	TT	1 V	-5^oC	200 mV
PVT_{worst}	SS	0.9 V	-40^oC	215 mV

of Fig. 5.9 while PVT_{worst} is identified in Section 5.3 that closely relates to realistic worst-case corner. The gate-under-test of segment model (shown in Fig. 5.8) would be replaced by the standard cell, and timing characteristics are observed with each load, i.e. W1-W30. The following procedure is used to generate LUT for each standard cell of the library,

(A) For each wire load (W1-W30) that is eventually driven by a standard cell, measure the typical-case average delay across the driver gate (GD_{typ}), interconnect (WD_{typ}) and segment (SD_{typ}) for each combination of gate and wire load using SPICE simulator. The circuit and external parameters are driven by typical-case corner, PVT_{typ}.

(B) Now change the typical-case corner, PVT_{typ}, to worst-case corner, PVT_{worst}. For each wire load (W1-W30), measure the worst-case average delay across the segment using SPICE simulator that can be observed as SD_{worst}.

(C) For each combination of driver gate and load (W1-W30), find the multiplication factor by using below equation,

$$M_f^a = 1 + \frac{SD_{worst} - SD_{typ}}{SD_{typ}} \tag{5.5}$$

In Eqn. 5.5, M_f^a is defined as a *delay multiplication factor to account for aging* which specifically include NBTI and HCI effects. Lets take an example of 2-D LUT generated for two-input NAND gate with drive strength 1 (NAND2_X1) as shown in Table 5.2. The GD_{typ} and WD_{typ} are used as input search data to a LUT of a specified standard cell. Apart from M_f^a, the table also stores the *delay multiplication factor to account*

Table 5.2: Example 2-D LUT for two input NAND gate

NAND2_X1			
$WD_{typ}(ps)$	GD_{typ}(ps)	M_f^p[5]	M_f^a
1.41	59.3	1.013	1.025
2.73	62.1	1.014	1.022
4.21	64.5	1.015	1.015
5.23	69.5	1.018	1.012
6.9	72.7	1.021	1.011

for process-variation shown as M_f^p [5]. M_f^p includes the random effects such as random dopant fluctuations, line edge/width roughness, gate dielectric variations and systematic variations related to interconnects. The present work is focused on analyzing the effects of aging, hence, for the sake of clarity we prefer not to discuss the process-related variations presented earlier in [5]. If the input search data (GD_{typ} and WD_{typ}) are not found directly in the LUT then the *linear interpolation method* is used to find the corresponding absolute values of M_f^p and M_f^a respectively. Assume that a two-input NAND gate (NAND2_X1 of Table 5.2) is driving a load of 2.73 ps than 2-D LUT would return M_f^p and M_f^a values as 1.014 and 1.022 respectively. These values translate to 1.4% and 2.2% worst-case variation on typical segment delay due to process-related and aging effects.

5.5 Path Delay Estimation and Critical Path Selection Procedure

Previous section has broadly covered the segment definition, its modeling and generation of 2-D look-up table for each standard cell of the library. So far, we are well equipped to estimate the worst-case scaling required on typical delays of the segment. In this section, we will discuss how the segment delay multiplication factors help in estimating the overall worst-case path delays and bottom-up approach in identifying a set of long

paths that may be critical over time.

5.5.1 Path Delay Estimation

As discussed earlier, the segments are the smallest entity of a circuit and any path can be defined using an ordered set of contributing segments. A segment can be a part of multiple paths at the same time and each path uses the following equation to estimate typical path delay,

$$PD_{typ} = \sum_{i=1}^{n} SD_{typ}(i) \tag{5.6}$$

$1 \leq i \leq n$, where n is the number of segments in a path. SD_{typ} is the segment delay estimated by SPICE simulator at typical-case corner. As can be seen in Eqn. 5.6, PD_{typ} is the sum of all segment delays that constitute the target path. It is important to point out that typical corner usually falls at the middle of probability density function of path distribution curve of a circuit. Hence, the worst-case path delays can be estimated using LUT-based approach by following equation,

$$PD_{worst} = \sum_{i=1}^{n} (SD_{typ}(i) * [M_f^p(i) + M_f^a(i) - 1]) \tag{5.7}$$

Now we have introduced two additional terms M_f^p[5] and M_f^a in Eqn. 5.7 to account for worst-case expected delay. These are delay multiplication factors to account for spatial and temporal variations in CMOS circuits. It must be seen in Eqn. 5.7 that each segment of a path uses its own M_f^p and M_f^a, that enables a dynamic scaling factor to be used for every segment that defines a path. *Use of dynamic scaling factors in our approach definitely has an edge over traditional STA approach where designers use a static factor to estimate worst-case delay.* To further understand the proposed path delay estimation approach, we have selected a random path from benchmark circuit s35932 as shown in Table 5.3. The path starts from output Q of a flip-flop (SDFFRPQ_X1M_C35), traverse through INTERCONNECT_I1, and feeds input C of XNOR3_X0P7M_C35. The

Table 5.3: A random path from benchmark s35932 for worst-case path delay estimation

Standard Cell Name	Timing Arc	Delay	SD_{typ}	M_f^p+M_f^a -1	SD_{worst}
SDFFRPQ_X1M_C35	CK↑ → Q↓	64.2			
INTERCONNECT_I1		0	64.2	1.01	64.8
XNOR3_X0P7M_C35	C↓ → Y↓	96.3			
INTERCONNECT_I2		5	101.3	1.015	102.9
NOR2_X1A_C35	A↓ → Y↑	46.8			
INTERCONNECT_I3		12	58.8	1.025	60.3
AOI211_X1M_C35	B0↑ → Y↓	36.6			
INTERCONNECT_I4		8	44.6	1.023	45.6
OAI211_X1M_C35	B0↓ → Y↑	54			
INTERCONNECT_I5		2	56	1.014	56.8
SDFFRPQN_X2_C35	D↑	0			

output Y of XNOR3_X0P7M_C35 is driving input A of NOR2_X1A_C35 through INTERCONNECT_I2 and this way, the path traverse through other gates/interconnects before being terminated at input D of flip-flop (SDFFRPQN_X2_C35). For sake of clarity, interconnects between the standard cells are shown as INTERCONNECT_I1 to INTERCONNECT_I5 in column 1 of Table 5.3. However, standard cell library does not have any predefined model for interconnects. Timing arc, of column 2, represents the timing relation (transition 0 to 1 or 1 to 0) between the input and output ports of a standard cell. Column 3 of Table 5.3 shows the typical delays (in ps) of individual gates and interconnects which are referred as GD_{typ} and WD_{typ} as discussed in Sub-section 5.4.2.

Column 4 shows the segment delay for each combination of driver gate and interconnect. The individual delay multiplication factors for each segment are given in column 5 which are extracted and approximated from generated LUTs as discussed in Sub-section 5.4.2. The column 6 shows the worst-case delay estimated under circuit parameters fluctuation. By using Eqn. 5.6 and 5.7, we can easily calculate PD_{typ} and PD_{worst} as 324.9

ps and 330.4 ps respectively. Hence, we can conclude that an overall delay variation of 1.7% is expected on the target path as mentioned in Table 5.3. In the experimental section, we will demonstrate the accuracy of the proposed methodology in comparison to SPICE simulated results and also compare the results with pessimistic STA on such paths.

5.5.2 Path Selection Procedure

In previous sub-section, we have discussed the approach that helps in estimating worst-case path delays if the order of segments, type of driver gate (of each segment) and its typical timing characteristics are known using LUTs. This section broadly covers the

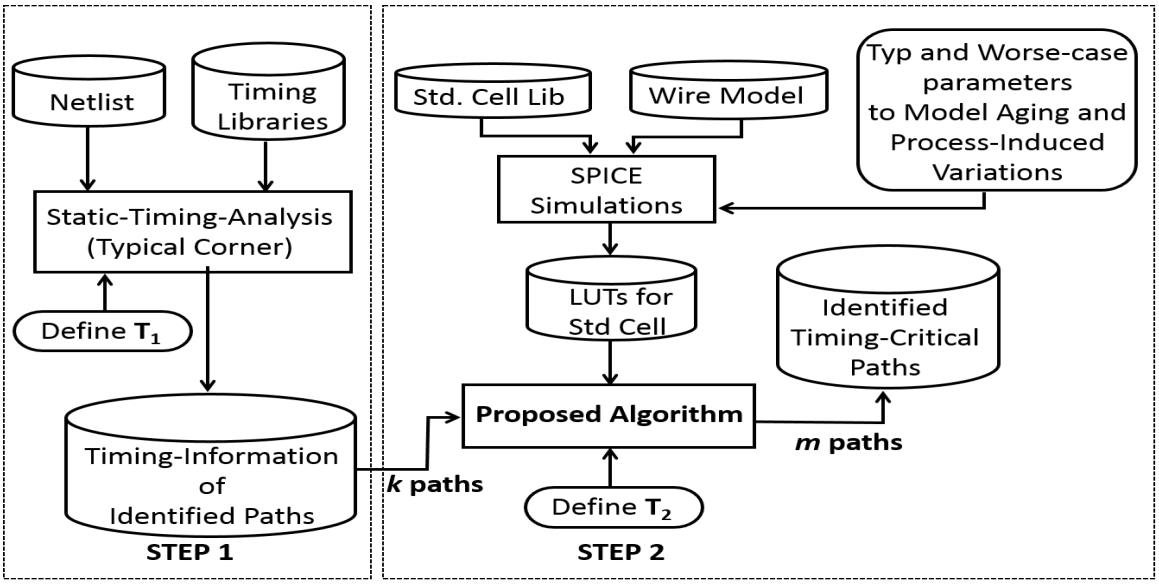


Figure 5.10: Block diagram of the proposed methodology

methodology which can help in identifying a set of long paths that may become critical over time due to circuit unreliabilities. The complete methodology is divided into two steps which can be seen in Fig. 5.10. We first introduce few terminologies that will be used in rest of this paper. T_{sys} is defined as the minimum target clock period at which the circuit is expected to work. T_1 is user defined value which sets the limit to identify those paths whose worst-case path delays need to be re-estimated using the proposed

methodology. It can be seen in Fig. 5.10 that T_1 helps STA to isolate only those paths whose timing delays exceed it's absolute value which is $0.8T_{sys}$ in our case. T_1 can assume any value based on the intermediate path definition as suggested by the circuit designer, $0 < T_1 < T_{sys}$.

T_2 is an another user defined value that formally sets the lower bound limit on the absolute path delay such that if any path whose timing delay exceeds this limit then it can be categorized as speed-limiting path. In our case, we have used $T_2 = 0.9T_{sys}$ to isolate a set of timing critical paths and a formal relation among these user defined values is $0 < T_1 < T_2 < T_{sys}$. It can be seen in Fig. 5.10 that T_1 helps in isolating those paths, using STA, whose timing delay require pessimism removal and re-estimating the path delay with reasonable accuracy to match SPICE simulator estimated worst-case delay. Once the path delays are estimated again in *STEP 2*, a user-defined value T_2 sets the lower bound to identify a set of timing critical paths.

STEP 1: To begin with, we still continue to use STA to identify a bigger set of paths that may be speed-limiting. In STEP 1, STA reads the circuit netlist, timing libraries, physical data like placement, floor plan, routing etc and user-define value T_1 to identify a list of paths whose delay exceeds a threshold value limit T_1 at typical corner-case PVT_{typ}. A set of identified paths from this step qualifies to be analyzed by the radical proposed algorithm, in STEP2, to estimate realistic worst-case path delays in comparison to STA analysis. At the end of STEP 1, path-based timing information is generated for all identified paths (lets say the number is k) whose delays are greater than or equal to T_1.

STEP 2: As shown in Fig. 5.10, the proposed algorithm reads the path-oriented typical timing delays of each segment (gate delay GD_{typ} and wire delay WD_{typ}), and uses generated LUTs to estimate worst-case path delays using Eqn. 5.7. Thereafter, if any path whose delay exceeds a user-defined value T_2 then it must be included in the final set of long paths whose count is m. The pseudo-code and implementation of the proposed algorithm is shown as ***Algorithm 1***.

Algorithm 1: Path Selection Procedure

01: *Procedure: Enumerate all paths whose delay is greater than* T_1 ($PD_{typ} \geq T_1$)
02: ***Set*** $PD_{worst} = 0$;
03: ***while*** *all paths are not covered* {
04: ***while*** *all segments are not covered on selected path* {
05: ***for*** *selected segment* {
07: ***getdelayfactor***(gate_type,drive_strength,WD_{typ},GD_{typ});
08: $SD_{worst} = [WD_{typ}+GD_{typ}] \times [M_f + M_f^a$ -1]
09: ***end for*** }
08: $PD_{worst} = PD_{worst}+SD_{worst}$
10: ***end while***
11: *if* $PD_{worst} \geq T_2$ {
12: ***return*** *(SUCCESS)*;
15: *reset_all_delays*; }
17: ***else*** {***return****(FAILURE)*;
18: *reset_all_delays*; }
19: ***end if*** }
20: ***end while***
24: ***subroutine getdelayfactor (a,b,x,y)***
25: input gate_type, drive_strength
26: input WD_{typ},GD_{typ}
28: The 2-D LUT, as presented in Sub-section 5.4.2, returns the value of M_f and M_f^a

---

The above algorithm works on the principle governed by Eqn. 5.7 as discussed in Sub-section 5.5.1. It enumerates the segment delays first and then estimates the actual path delay. The delay multiplication factors for each segment of a path are determined by using subroutine *getdelayfactor*. The typical wire and gate delays are available at the end of STEP 1 and based on their absolute delays, LUT of a specified standard cell returns the interpolated values of M_f and M_f^g. From the top level overview of the program, it seems like the algorithm can not process multiple paths at the same time. However, multiple paths can be processed by the actual implementation of the algorithm,

based on the available number of threads in the host system. Assume that the number of identified paths in the first step is k and the average number of segments in a path is n, then the time-complexity of the algorithm is $O(kn)$, which is a big number for the complex designs. As the computation of path delay is independent, it can be parallelized to reduce the time-complexity to $O(kn/T)$ where T is number of threads.

5.6 Experimental Results and Discussion

This section demonstrates the effectiveness of the proposed methodology by performing a set of experiments on ITC'99, IWLS'2005 and ISCAS'89 benchmark circuits. These circuits are chosen based on varying size and complexity. The column 1 of Table 5.4 shows all benchmark circuits, associated total number of gates and flip-flops in column 2 and 3 respectively. We have used *Cadence RTL compiler* for generating physical netlist using the 28nm planar CMOS library, *Cadence encounter timing system and SoC encounter* are used for static timing analysis and physical place-n-route using 12 tracks. However, such experiments are not dependent on some specific tools and user can choose any approach for IC design. As discussed earlier in Sub-section 5.5.2, the *STEP1* provides the path-based structural data along with the absolute gate (GD_{typ}) and wire delays (WD_{typ}) per segment. Such information would be used as input search data in LUTs to obtain the realistic worst-case scaling factors (M_f and M_f^g). These scaling factors are used to estimate the path delay under process and aging-induced variations.

5.6.1 Effectiveness of the Proposed Methodology

The path delay estimation is driven from Eqn. 5.7 that works on each segment of a path and provides the overall path delay with reasonable accuracy. In order to validate the accuracy of the proposed approach with SPICE simulated results, a 5% of the total number of paths are chosen from each benchmark circuit to prove the concept. It must be remember that the LUTs are generated at PVT_{typ} and scaled further by using multiplication factors to match SPICE simulated results at PVT_{worst} corner. Fig. 5.11 shows

Table 5.4: Number of Identified Long Paths Using Different Timing Models, $T_2 = 0.9T_{sys}$

Ckt.	#Gate	#Flop	N_{sta}	N_{spice}	N_{pro}
b15_1	3581	449	185	190	209
b17_1	10942	1415	905	587	625
b18_1	29963	3326	1803	1734	1793
b19_1	56919	6654	3608	3556	3594
dma	9100	2200	1427	1335	1404
usb_funct	6382	1766	91	91	93
ethernet	21379	10545	123	84	103
mem_ctrl	3286	1144	243	225	235
vga_lcd	33377	17102	8584	6073	6181
s35932	3268	1728	876	840	866
s38417	3886	1564	119	96	99
s38584	5504	1451	37	26	29

the absolute path delay as estimated by SPICE, an earlier work [3] and the proposed approach. SPICE estimated delays are labeled as *SpiceDelay* whereas the delay estimated by the work presented in [3] and the current proposed work are shown as *CalDelay* and *ProDelay*. X-axis of Fig. 5.11 is the path identity number (Path ID) starting from 1 to 48 which are consecutively identified top four speed-limiting paths from each benchmark circuit. For clarity, we chose to show only top 48 critical paths out of thousands of paths. The circuits are synthesized for high performance to work at 1 GHz (time period 1000 ps). Y-axis shows the absolute path delay in picoseconds. It is observed that the inaccuracy in estimating path delays, compared to SPICE simulators at same PVT corner, has reduced from approx 10% (reported inaccuracy in [3]) to 5.5%. This is also evident from the Fig. 5.11 that *CalDelay* is relatively more pessimistic as compared to the current technique. A *Pearson correlation coefficients* of 0.96 is reported between

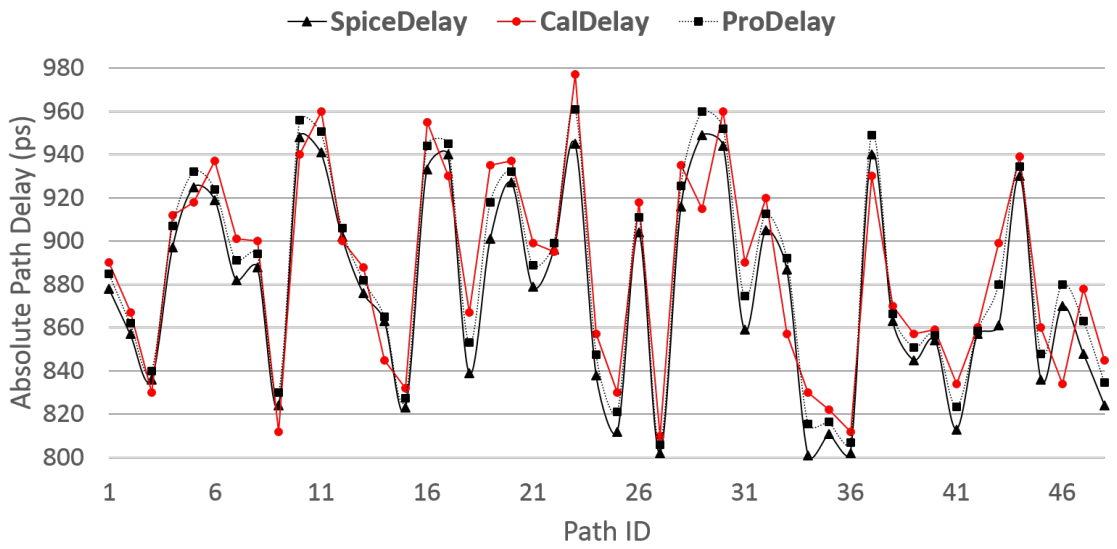


Figure 5.11: Comparison among the worst-case path delay estimated by SPICE, previous work [3], and the proposed approach

the data obtained from the proposed path estimation technique and SPICE simulated delays at PVT_{worst}. A radically identified worst-case PVT corner helps in improving the correlation coefficient of 0.93 as reported earlier in [3].

Now we have to evaluate the quantitative correctness of the proposed approach by analyzing the number of potential timing-critical paths identified by performing STA analysis, SPICE analysis and the proposed approach. The absolute number of m identified paths in *STEP 2* of Fig. 5.10, are shown in column 4, 5 and 6 of Table 5.4. The number of critical paths as identified by performing STA, SPICE and the proposed approach are shown as N_{sta}, N_{spice} *and* N_{pro} with a user-defined threshold value of $T_2 = 0.9T_{sys}$. In Table 5.4, it must be seen that STA estimated number of critical paths are approx 19% pessimistic in comparison to SPICE estimated number of paths. The only exception is benchmark *b15_1* where STA is optimistic as compared to its usual behavior. After analyzing the paths of *b15_1*, it is found that most of the paths are interconnect dominant as compared to other cases. Such optimistic cases of STA are encountered earlier in [5] on real silicon. However, *the proposed approach is only 6.2% pessimistic in estimating a set of critical paths in comparison to SPICE estimated results.* This clearly

shows the advantage of using our proposed methodology in identifying speed-critical paths under aging.

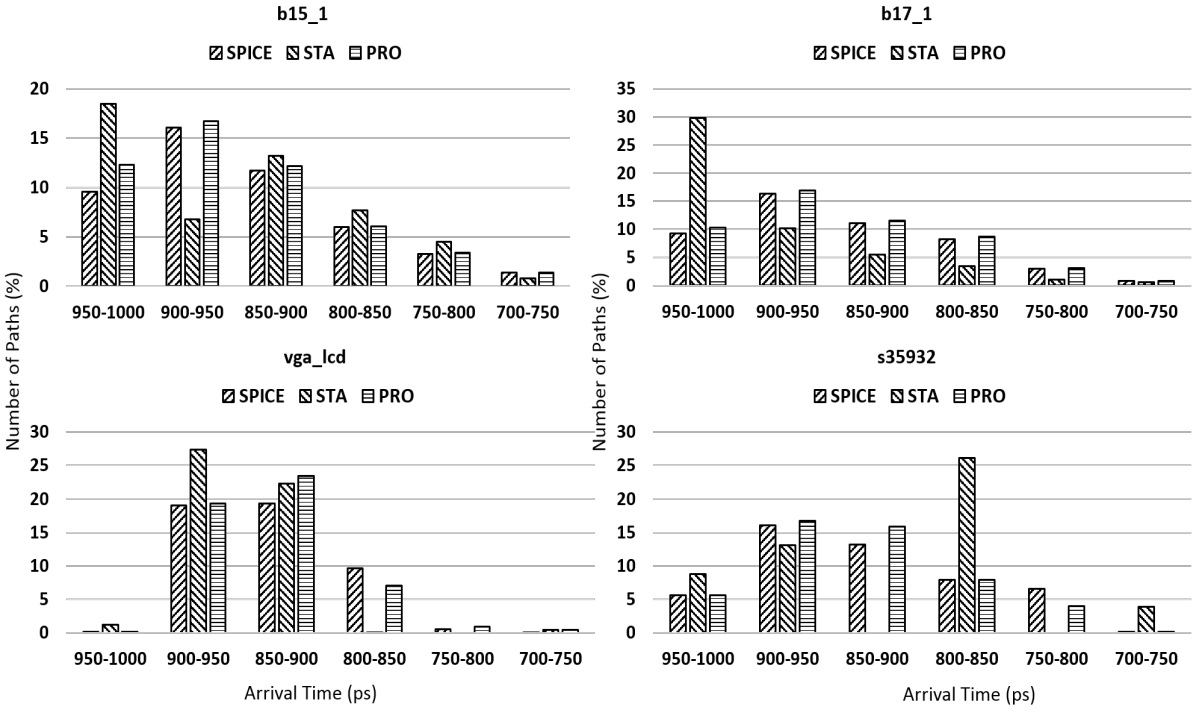


Figure 5.12: Path distribution obtained by using SPICE, STA based approach and the proposed approach (PRO)

In order to capture the granular details at circuit level, we compared the path distributions of few benchmark circuits from 700 ps to 1000 ps of path arrival time. The path distribution charts as shown in Fig. 5.12 are prepared for few benchmark circuits, such as *b15_1, b17_1, vga_lcd and s35932*, to compare the percentage of timing critical paths as estimated by STA, SPICE and the proposed methodology. Consider one of the benchmarks, b15_1, where we perform STA and SPICE analysis separately to estimate delays of all possible structural paths. Now isolate all paths whose absolute delay exceed 950 ps with an assumption that circuit is timed to work functionally at 1000 ps. Lets call the number of paths identified by STA and SPICE analysis as M and N (whose delays fall between 950-1000 ps) and the percentage of paths that lie between 950-1000ps are calculated as [M/(M+N)]x100 and [N/(M+N)]x100 respectively.

Using experimental observations, we obtained M=376 and N=354 between 950-1000 ps arrival time range that finally translates to 18.5% and 9.7% for STA and SPICE respectively which are shown in Fig. 5.12. The figure shows the path distribution obtained over path arrival time using STA and SPICE assisted delay analysis. It can be seen clearly that STA estimated path distribution results are pessimistic and unnecessary over-burdens the functional or ad-hoc scan test to regularly observe the performance of those paths which never turn-out to be speed-limiting. To further validate the effectiveness of the proposed approach, path distribution curves are also prepared for the same set of benchmark circuits to compare against SPICE simulated results. Fig. 5.12 also depicts the path distribution curves obtained using the proposed approach as *PRO* and it can be seen that the path distribution curves of the proposed technique closely follow the curves estimated using SPICE analysis. In conclusion, user can *reduce the number of STA identified critical paths by approx 9% using the proposed approach with path delay estimation accuracy of approx 94% in comparison to SPICE simulated delays.*

5.6.2 CPU runtime comparison

This sub-section analyzes the time complexity of the proposed algorithm, it's timing performance and memory requirements against highly optimized SPICE simulators. As discussed earlier in Sub-section 5.5.2, the algorithmic complexity of the proposed algorithm is *O(kn/T)* where k is the number of qualified paths considered for worst-case delay analysis, n is the average number of segments in a path and T is number of threads that can be handled on a processor(s). In practice, the state-of-the-art processors have limitations in handling multiple threads at the same time and create bottleneck for optimum performance. Hence, we start using T_{max} instead of T to show algorithmic complexity as $O(kn/T_{max})$, where T_{max} is the maximum number of threads that can be handled by the processor without showing performance degradation. Similarly, the algorithmic complexity of SPICE simulators is usually given by *O(k)* where k is the number of paths. Now assume that four paths are chosen for worst-case delay estimation (k=4) with an optimal number of threads as 45 (T_{max}=45). Now, the complexity reduced to *O(4n/45)*

and *O(4)* for the proposed approach and SPICE simulator respectively. This can be seen clearly that the expected speed-up in time from the proposed approach is 45/n, only if $n < 45$, as compared to SPICE simulators.

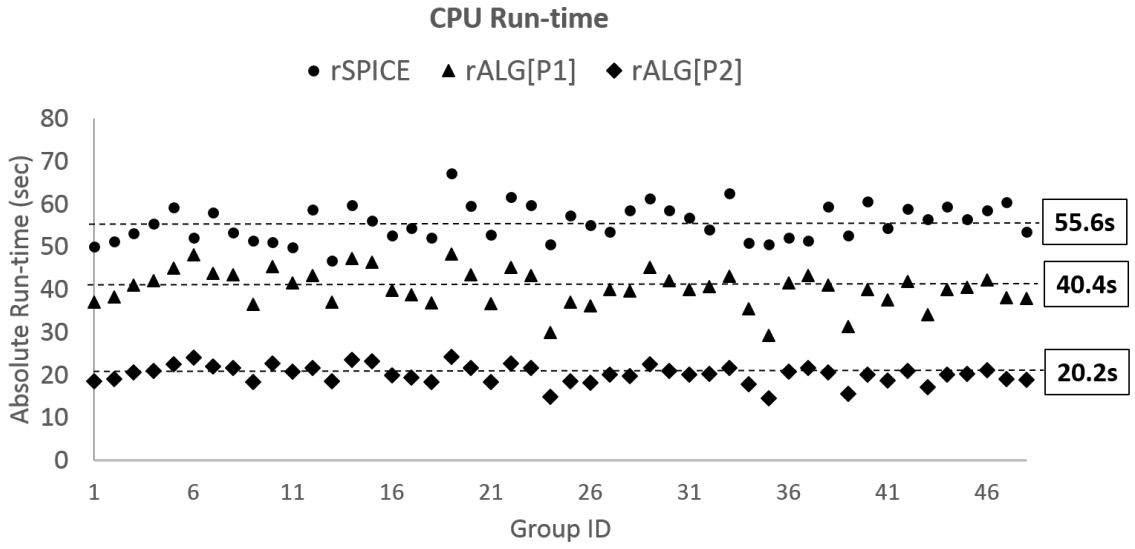


Figure 5.13: CPU runtime comparison

A theoretical performance gain in time using the proposed approach instead of using SPICE simulator, is T_{max}/n. However, the performance gain would saturate when $T_{max} \approx n$. In order to validate the theoretical observations, an experiment is carried out by creating a set of four paths from each benchmark circuits and labeled on X-axis under *Group ID* as shown in Fig. 5.13. The runtime of SPICE simulator, the proposed algorithm running on single processor and the same algorithm running on cluster of two processors are shown as *rSPICE*, *rALG[P1]* and *rALG[P2]* respectively. The Y-axis shows the absolute runtime in sec. The experiments are carried-out on 48 different groups where each group consists of four paths, $k = 4$. In this work, we have used two host systems where first uses a state-of-the-art processor available with a memory of 2GB and running on Linux platform. Second uses a cluster of two processors each having a memory of 2GB running on Linux. With any of the host systems, the average time taken by SPICE simulator (*rSPICE*) to process 48x4 paths is approx 55.6 sec. However, the proposed algorithm takes 40.4sec (*rALG[P1]*) and 20.2sec (*rALG[P2]*) with first and

second host systems respectively. Assume that a cluster of 4 processors is used then a speedup of approx 5.5 times (55.6/10.1) is expected over SPICE simulator and so on.

The runtime of the proposed algorithm can be improved further by increasing the number of processors per cluster and consecutively more number of paths for analysis, at an expense of reduced space efficiency of the algorithm. It must be noted that the run-time efficiency of the proposed algorithm in comparison to SPICE simulation, is achieved by (1) avoiding real-time simulations (as we generated reasonably accurate look-up tables to account for spatial and temporal unreliability of CMOS circuit), (2) reducing the number of elements which constitute a path by defining segments, (3) an optimal cluster of processors to increase the effective threads for parallelism, and (4) avoid multi-corner PVT analysis to save significant simulation time.

5.7 Summary

In order to identify a set of speed limiting paths, that may become critical over time, in stipulated design cycle time is a real challenge for system-on-chip (SoC) designers. One of the most adopted approaches to identify such paths is through STA analysis. The only limitation in using STA is the pessimism involved while estimating the path delay, due to which, it is not easy to identify a realistic set of speed-limiting paths. Another approach is to use SPICE simulations which are reasonably accurate in estimating path delay with respect to delay measured on silicon. However, the huge simulation runtime in path estimation makes it impractical. We proposed a methodology at an abstract level which is substantially faster than SPICE and still maintaining a path delay estimation accuracy of approx 94%. The primary objective is to develop a reliability-aware efficient methodology by which, a set of timing-critical paths can be identified. These set of paths should be tested explicitly through functional or structural scan after production to improve small delay defect (SDD) coverage or it can be periodically monitored for aging analysis under real workload. The proposed methodology requires STA and SPICE models during the procedure and hence, cannot be used to replace industry standard

multi-corner pessimistic STA analysis to sign-off circuit database for fabrication. The future work involves the validation of the methodology on complex industrial designs.

— * — * —

Chapter 6

Conclusion and Future Scope

Localized small delay defects (SDDs), for example degraded transistor drive strength caused by under-grown fin(s), are a growing concern in current FinFET and emerging gate all around (GAA) technologies. These small timing defects usually appear either during manufacturing process steps or due to aging effects in MOS transistors and interconnects under certain workload over the time. However, such defects can also originate while manufacturing of silicon cylindrical ingots, wafer preparation, and die packaging once wafers are fabricated with desired circuits. In literature, researchers have proposed different approaches to target small delay defects such as: a) path delay fault based ATPG testing can target distributed small delay defects assuming a set of timing critical paths is well defined, b) N-detect transition fault based ATPG testing typically sensitizes multiple paths to test the same TDF fault with no guarantee to hit the structurally longest timing path, c) timing-aware ATPG testing uses circuit level timing information to target timing critical transition faults by sensitizing longest timing paths or may choose subsequent longer paths in case of unsuccessful pattern generation.

The conventional ATPG approaches have limitations either due to high pattern volume or low SDD coverage. The practical issues with PDF-based ATPG are low path coverage and test application time which grows dramatically faster with increase in circuit size. The demerits in adopting N-detect ATPG are: a) effort in detecting same fault

n times which makes in expensive in ATPG runtime and pattern volume, b) no guarantee that fault is sensitized through longest testable path as it does not use circuit-level timing information, and c) no small-delay defect (SDD) coverage metric to claim defect free circuit. *In today's most complex SoC/ASIC designs, TDF based testing is mostly adopted because of its efficiency in pattern generation, ATPG runtime with many timing defects being covered. With the transition of technology node from planar to 3D FinFET structure, gross-delay defects which are typically covered by transition delay fault based ATPG is getting ineffective in targeting small timing defects.* This is due to the fact that it excites and propagates the fault effect through easy to control short or intermediate structural path instead of choosing longest timing path of a circuit.

6.1 Thesis Summary

The most adopted ATPG techniques to uncover SDDs are certainly driven by timing information that includes gate and interconnect delays. The TA-ATPG approach is reasonably good to adopt with major drawbacks of increase in test set size and test generation time. These limitations arise due to ATPG's ability to use circuit-level timing information. Apart from this, the requirement of sensitizing longest timing path for each fault puts constraints on ATPG's ability to cover multiple faults through a single pattern. This prunes the ATPG's efficiency in covering many faults through a single pattern by using fault simulation and implication approach.

The **Chapter 3** makes the following contributions to address the shortcomings of conventional approaches of targeting SDDs: a) proposes a new test methodology [4] that for the first time exploits path delay fault based test generation for the timing-critical paths to generate timing tests for many of the targeted timing-aware transition delay faults, b) discusses a new approach to cover SDDs more efficiently to reduce pattern volume and ATPG runtime as compared to traditional techniques, and c) modifies the definition of conventional SDD coverage metric to improve confidence in covering small

timing defects of critical paths. To uncover small timing defects of a circuit, it is important to identify a set of performance limiting path which closely matches silicon behavior and discussed further in chapter 4 and 5 respectively. At the end, the adopted methodology results in approximately 12.5% reduction in pattern volume, 35% reduction in ATPG runtime and also a 5% improvement in delay test coverage when compared to existing commercially available TA-ATPG approach. In the experiments, we have used a range of benchmark circuits to compare the results from a commercially available TA-ATPG [48, 49] with our new approach that efficiently exploit PDF tests wherever possible.

It must be understood that the extra small delays caused by distributed defects would be absorbed on those paths which have higher slack margin and would never result in erroneous circuit response. In order to target SDDs, it is utmost important to identify long paths of a specified circuit. However, identifying a set of timing critical paths under spatial or temporal variations is not straightforward and needs special attention while adopting a certain technique. A set of procedures is available in the literature that may help in identifying long paths. Static timing analysis (STA) [52] is mostly adopted to identify a set of top timing-critical paths under specified user-defined setup slack condition. STA is very powerful tool for design sign-off as it is very fast in meeting timing specifications, exhaustive in covering circuit nodes, and works at extreme process corners to guarantee device operation. However, it is pessimistic in estimating worst-case path delays which typically results in adding more number of paths to be analyzed by TA-ATPG while targeting timing-critical transition faults.

It must be remembered that even a small number of extra paths being added due to pessimistic STA can result in increased pattern volume and ATPG runtime. Another approach of estimating more accurate path delay is dynamic timing simulation (SPICE level simulation) [50, 51]. It estimates timing delays of various paths and a user-defined timing critical condition helps in isolating a set of speed-limiting paths. However, it is traditionally non-exhaustive and very slow in estimating timing delays even for a subset of the circuit. Apart from this, statistical static timing analysis (SSTA) [53] can also be adopted which identifies the speed-limiting paths based on the target yield. SSTA

represents the delay in terms of path distribution function and invariably uses SPICE models to generate industry's first open standard effective current source model (ECSM [92]). Typically, it is also slow as it uses probability density functions to estimate the timing delays as a function of yield.

In today's complex designs, STA is typically used as it is faster than any of the timing-analysis methods available in the literature. The only major limitation is the pessimism in estimating the timing delay due to which, it is not easy to identify a reduced set of speed-limiting paths. The pessimism in delay calculation is due to the fact that it uses abstract level models which provide pessimistic delays of gates and interconnects [83]. In order to time circuit at worst corners, *STA uses the extreme values of all physical parameters related to MOS transistors and interconnects along with worst environmental conditions which are unlikely to happen at the same time.* That is why, when the parametric testing is performed, it is not rare to observe that most of the devices perform better than expected in terms of operating frequency.

All of the previous timing estimation procedures either have limitations in terms of path delay estimation accuracy or overall runtime. *A more accurate procedure demands extensive runtime and computational power. On the contrary, a fast timing estimation procedure can be used by relaxing delay estimation accuracy.* The Chapter 4 and 5 are dedicated to defining a new path delay estimation approach which is substantially faster than SPICE level simulation with reasonably good accuracy in comparison to STA. In order to overcome the conventional shortcomings of timing analysis approaches, **Chapter 4** introduces a new approach in identifying a set of long paths that may limit the circuit performance under spatial statistical variations. The chapter develops a theoretical explanation for the unexpectedly higher process related timing variability shown by long interconnects that are driven by high drive strength gates. This gets even worse due to conventional gate delay variability and other random process effects. Our analysis is supported by actual silicon data and further validated by detailed Monte-Carlo (MC) simulations. Unfortunately, traditional scan based transition delay fault (TDF) timing tests can miss these variabilities induced delay faults on long interconnects which lies

on the critical paths. The Chapter 4 also proposes: a) a methodology to identify high variability paths dominated by such long interconnects, with the aim of developing high quality delay timing tests, and b) develops a heuristic based path selection algorithm to identify potentially slow paths that can contribute to test escapes in production. We further extend our approach to generate high quality delay timing tests for the target paths using the proposed "three pass" method inherited from [4].

The Chapter 5 improves the accuracy of path delay estimation algorithm introduced in Chapter 4 and also extends to accommodate the effects of aging on delay estimation. **Chapter 5** is dedicated to exploring the opportunity of using the proposed methodology, instead of using pessimistic STA or time-consuming SPICE for delay calculations at worst PVT corner, to identify a set of realistic long paths that may become critical during device life-time. This chapter revisits the segment model presented in [3, 5] by replacing the worst-case distributed RC T-network with performance driven balance RC π-network [52, 96], which can be optimized for standard fan-outs. This is a faster high-level approach for estimating path delays in a circuit and has very good accuracy. Then it adopts a unified aging model [25, 28] of NBTI and HCI degradation in terms of threshold voltage (V_{th}) shift over device lifetime. The V_{th} shift and empirically chosen realistic worst-case PVT corner avoids multi-corner delay estimation requirement. This helps in predicting the maximum V_{th} degradation, due to NBTI and HCI, that can be observed at specified process-voltage-temperature (PVT) corner. Theoretically estimated V_{th} degradation based on reaction-diffusion (R-D) model, presented in [28], is empirically compared with the results obtained from 1000 industrial production parts fabricated on 28nm planar CMOS technology node. The chapter also identifies the worst-case PVT corner that shows maximum temporal variations due to predicted V_{th} degradation. This enhances the algorithm introduced in Chapter 4 that is now substantially faster than SPICE simulators [50, 51] with an approximate 94% accuracy in estimating path delays at specified PVT corner. It helps in reducing the number of extra timing critical paths added due to pessimistic behavior of STA, by 50%. This finally results in improving the TA-ATPG runtime and pattern volume with an aim to improving SDD coverage.

The Chapters 4 and 5 collectively develop a methodology of identifying a set of long paths that may fail or degrade the circuit performance over time. The presented methodology uses the proposed algorithm in isolating such speed-limiting paths that can be used to improve SDD coverage. Alternatively, it can be used to periodically monitor the aging impact. It is worth to point-out that *our aim is neither to replace industry standard multi-corner pessimistic STA analysis to sign-off circuit database for fabrication nor to develop the standalone aging-aware timing library.* Our objective to isolate a real set of critical paths, whose timing behavior must be close to silicon, that must be tested to improve the confidence of covering SDDs. As the outcome of the proposed methodology and its sustainability are dependent on delay estimation by SPICE and STA based approaches, it cannot be adopted as independent delay estimation technique.

6.2 Future Work

So far, we have discussed a new test methodology that can be adopted for effective fault detection with improved efficiency in pattern generation and runtime respectively. As discussed earlier in Chapter 3, path-based ATPG is being used in our proposed test methodology where PDF test generation has been exploited to not only generate the timing tests more efficiently, but the resulting TDF test sets are also more compact and perform better on commonly used delay test coverage metrics. This is because all TDF faults along a PDF targeted timing-critical path can be detected efficiently by generating a single PDF test. This efficiency is not explicitly exploited by node oriented TDF test generation even when the TDFs are targeted along the longest paths. In order to make the proposed test methodology more effective and efficient, we may explore following enhancements,

- explore the possibility of using segment-based or ALAPTF-based test generation to improve partial path coverage. This certainly help in improving SDD detection using partial robust-test generation criteria which is discussed earlier in Chapter 2

under Sections 2.1.4 and 2.1.5 respectively. In short, if we can not robustly or non-robustly sensitize a full path then we may choose partial sensitization of segments by relaxing transition launching and fault effect propagation test criteria.

- explore the possibility of dynamically limiting the ATPG runtime if the desired path coverage is achieved while PDF test generation. In other words, for complex designs, sometimes path-based ATPG takes roughly more time than TA-ATPG for targeting long paths in a circuit. Under such circumstances, it is important to prioritize the objective either to improve path coverage or limit ATPG runtime once a desired coverage is achieved. A new approach can be developed which terminates the ATPG run if the desired test coverage is attained.

The proposed worst-case path delay estimation algorithm may be enhanced to include following effects,

- On-chip crosstalk: can be defined as electric field coupling, magnetic field coupling, and common impedance coupling between the circuits or circuit elements once they are fabricated in close proximity to each other. This usually happens if a resistive, capacitive or inductive path exist between the aggressor and victim circuit element. In this thesis, we have not considered any such effects while estimating worst-case path delay under spatial or temporal statistical variations.

- Simultaneous switching noise (SSN): can also be defined as a voltage droop observed at the victim circuit when an another sub-circuit shares the same power or ground rail on the same IC. This phenomenon is also known as power/ground noise, ground bounce, substrate noise or dI/dt noise when the power/ground signal is connected with a bondwire. This thesis does not honor SSN related temporary timing defects in the circuit.

- Soft error or single bit/event upset: The soft error typically happens when energetic particles such as alpha and gamma rays interact with semi-conducting material. Due to the generation of electron-hole pairs in short interval of time, it

may result in non-destructive state change in storage devices. This work can be extended to include such effects under controlled environment.

- Eletromigration: it is one of the dominant aging effects as observed on interconnects and vias in the ICs. The effect causes material transport by gradual movement of the ions in a conductor due to the momentum transfer between conducting electrons and the diffusing metal atoms. The proposed timing estimation algorithm can be extended to include small timing defects caused due electro-migration, however, it is fairly important to validate it on real silicon as well.

$- * - * -$

Appendix A

Experimental Setup and Tools

A.1 Benchmark Circuits

In this thesis, we have used standard benchmark circuits from ITC'99, OpenCore benchmark circuits like WISHBONE LCD/VGA controller, WISHBONE memory controller, ethernet IP, USB functional core, DMA controller, and ISCAS'89 benchmark circuits of varying complexity and size. Fig. A.1 shows the floor plan of a commercial SoC where the benchmark circuits are synthesized as soft IPs and are located into the sea-of-gates enclosed under red mark. The different voltage levels can be observed such as EVDD, O3VDDD, CVDD, O2VDD etc., however, we can not disclose the absolute voltage levels and used IPs or COREs into the device. It is worth to point-out that the benchmark circuits are synthesized and placed with a complex design to observe the realistic effects of circuit components while estimating worst-case path delays. The benchmark circuits did not functionally interact with any of the SoC components/IPs and physically placed for experimental purpose only.

A.2 Synthesis, ATPG and CAD tools

The design is being synthesized using Cadence RTL Compiler with 28nm planar CMOS library and optimized for area under fixed static power budget. After logic synthesis,

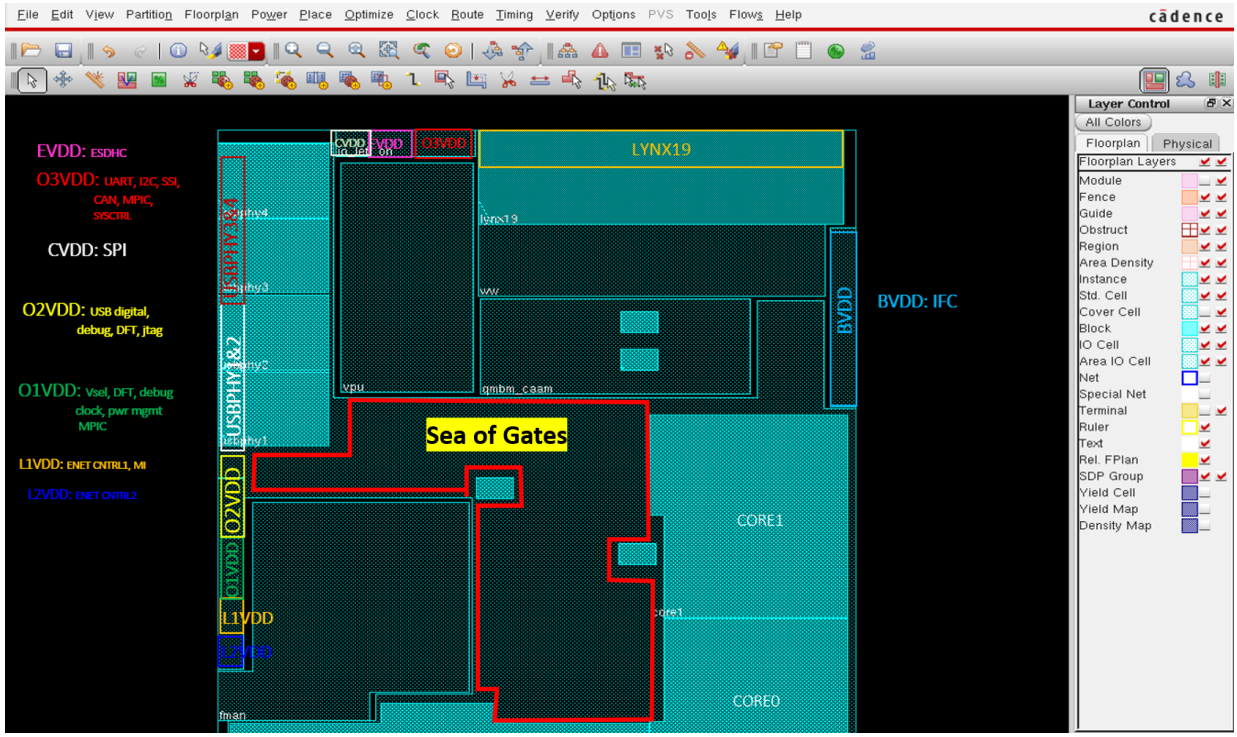


Figure A.1: Benchmark circuits in the sea-of-gates of commercial SoC

Cadence SoC Encounter is used for floor-planning, placement and routing. At the end, static timing analyzer is used to sign-off the design for fabrication. We have used Cadence Encounter Timing System (ETS) as STA tool which uses following data,

- Reading timing libraries: Timing library files contain timing information in ASCII format for all of the standard cells, blocks and I/O pad cells. The ETS software reads Synopsys Technology Library format files (.lib) models.

- Reading timing constraints: To ensure that design meets the timing requirements, designer must specify the relevant constraints. It can use the timing constraints to set, 1) timing context for constraint assertions, 2) boundary conditions such as input and output delays, 3) slew rates, 4) path exceptions such as false paths, path delays and cycle additions, 5) disable certain paths in the design.

- Performing sanity checks: At various stages of the design process, designer can check for missing or inconsistent library and design data.

- Reading delay data: An SDF file contains negative and positive delay changes caused by crosstalk. The full SDF file contains details about the absolute delays for all cells and interconnects, including IR drop and crosstalk impact.

- Reading parasitic data: Designer can specify the parasitic data for more accurate timing analysis using a SPEF file.

- Reading physical data: Tool can read physical information such as placement, floorplan and routing that you created using SOC Encounter in ETS.

In order to perform various structural test based experiments, scan cells are stitched into multiple chains in such a fashion such that number of scan cells do not exceed 120 per chain. We have observed that by choosing the right number of scan cells per chain, it helps in optimizing the pattern volume and test coverage in the presence of on-chip compressor/decompressor logic. The compressor and de-compressor logic are also used, however, while reporting the test coverage, ATPG runtime, and pattern volume, we bypassed the compression-decompression logic as benchmark circuits are only limited to experiments only. Mentor Fastscan is used to generate patterns under timing aware and timing unaware mode with relevant settings as discussed in previous chapters.

Bibliography

[1] N. Ahmed, M. Tehranipoor, , and V. Jayaram, "Timing-based delay test for screening small delay defects," in *Proc. of IEEE Design Automation Conf.*, 2006.

[2] S. K. Goel, N. D. Prasanna, and R. P. Turakhia, "Effective and efficient test pattern generation for small delay defect," in *Proc. of VTS*, 2009.

[3] A. Srivastava, V. Singh, A. Singh, and K. Saluja, "Identifying high variability speed-limiting paths under aging," in *Proc. of IEEE Latin American Test Symposium*, 2017.

[4] A. Srivastava, A. Singh, V. Singh, and K. Saluja, "Exploiting path delay test generation to develop better tdf tests for small delay defects," in *Proc. of Int. Test Conference*, 2017.

[5] A. Srivastava, V. Singh, A. Singh, and K. Saluja, "A methodology for identifying high timing variability paths in complex designs," in *Proc. of IEEE Asian Test Symposium*, 2015.

[6] M. Yilmaz, K. Chakrabarty, and M. Tehranipoor, "Test-pattern selection for screening small-delay defects in very-deep submicrometer integrated circuits," in *IEEE Tran. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, 2010.

[7] H. Kitada, "The influence of the size effect of copper interconnects on rc delay variability beyond 45nm technology," in *Proc. of Int. Interconnect Technology Conf.*, pp. 10–12, 2007.

[8] L. He, A. B. Kahng, K. H. Tam, and J. Xiong, "Design of integrated-circuit interconnects with accurate modeling of chemical-mechanical planarization," in *Proc. of SPIE*, vol. 5756, Design and Process Integration for Microelectronics Manufacturing 3, p. 109, 2005.

[9] E. Demircan, "Effects of interconnect process variations on signal integrity," in *IEEE International SOC Conference*, Sep. 2006.

[10] J. A. Davis, "Interconnect limits on gigascale integration (gsi) in the 21st century," in *Proc. of IEEE*, vol. 89, issue 3, 2001.

[11] M. Yilmaz, K. Chakrabarty, and M. Tehranipoor, "Interconnect-aware and layout-oriented test-pattern selection for small-delay defects," in *Proc. of ITC*, pp. 1–10, 2008.

[12] K. Takeuchi, T. Fukai, T. Tsunomura, A. Putra, A. Nishida, S. Kamohara, and T. Hiramoto, "Understanding random threshold voltage fluctuation by comparing multiple fabs and technologies," in *International Electron Device Meeting*, pp. 467–470, 2007.

[13] C. Shin, X. Sun, and T.-J. K. Liu, "Study of random-dopant-fluctuation (rdf) effects for the trigate bulk mosfet," in *IEEE Transactions on Electron Devices*, 2009.

[14] P. Stolk, F. Widdershoven, and D. Klaassen, "Modeling statistical dopant fluctuations in mos transistors," in *Transaction on Electron Devices*, 1998.

[15] K. Kuhn, "Reducing variation in advanced logic technologies: Approaches to process and design for manufacturability of nanoscale cmos," in *International Electron Device Meeting*, pp. 471–474, 2007.

[16] G. G. E. Maricau, *Analog IC Reliability in Nanometer CMOS*. Chapter 2, Springer Science+Business Media New York, 2013.

[17] J. Xiaobo, W. Runsheng, C. J. Y. Tao, and H. Ru, "Investigations on line-edge roughness (ler) and line-width roughness (lwr) in nanoscale cmos technology: Part i; modeling and simulation method," in *IEEE Transactions on Electron Devices*, 2013.

[18] H.-W. Kim, J.-Y. Lee, J. Shin, S.-G. Woo, H.-K. Cho, and J.-T. Moon, "Experimental investigation of the impact of lwr on sub-100 nm device performance," in *IEEE Transactions on Electron Devices*, 2004.

[19] A. Asenov, S. Kaya, and A. Brown, "Intrinsic parameter fluctuations in decananometer mosfets introduced by gate line edge roughness," in *IEEE Transactions on Electron Devices*, 2003.

[20] G. Klutke, P. Kiessler, and M. Wortman, "A critical look at the bathtub curve," in *IEEE Transaction on Reliability*, pp. 125–129, Volume 52, P125-129, Volume 52, 2003.

[21] Y. Li, Y. M. Kim, E. Mintarno, D. S. Gardner, and S. Mitra, "Overcoming early-life failure and aging for robust systems," in *Proc. of IEEE Design and Test of Computers*, 2009.

[22] R. Kwasnick, "Impact of technology scaling on htol," in *Proc. of IEEE IRPS*, 2012.

[23] W.-S. Liao, Y.-G. Liaw, M.-C. Tang, S. Chakraborty, and C. W. Liu, "Investigation of reliability characteristics in nmos and pmos fin-fets," in *IEEE Electron Device Letter*, July 2008.

[24] P. Magnone, F. Crupi, N. Wils, R. Jain, H. Tuinhout, P. Andricciola, G. Giusi, and C. Fiegna, "Impact of hot carriers on nmosfet variability in 45- and 65-nm cmos technologies," in *IEEE transactions on Electron Devices*, 2011.

[25] Y. Wang, S. Cotofana, and L. Fang, "A unified aging model of nbti and hci degradation towards lifetime reliability management for nanoscale mosfet circuits," in *Proc. of IEEE/ACM NANOARCH*, 2011.

[26] H. Kufiuoglu and M. A. Alam, "A geometrical unification of the theories of nbti and hci time-exponents and its implications for ultra-scaled planar and surround-gate mosfets," in *Proc. of IEEE IEDM Technical Digest*, 2004.

[27] M. F. Zakaria, Z. Kassim, M.-L. Ooi, and S. Demidenko, "Reducing burn-in time through high-voltage stress test and weibull statistical analysis," in *Proc. of IEEE Design and test of computers*, Apr 2006.

[28] H. Kufiuoglu, "Mosfet degradation due to negative bias temperature instability (nbti) and hot carrier injection (hci) and its implications for reliability-aware vlsi design," in *PhD dissertation, Purdue University*, 2007.

[29] S. Han, B. Kim, and J. Kim, "Variation-aware aging analysis in digital ics," in *IEEE transaction on VLSI systems*, 2013.

[30] "The lx2160a multicore communications processor." https://www.nxp.com/products/processors-and-microcontrollers/arm-based-processors-and-mcus/qoriq-layerscape-arm-processors/the-lx2160a-multicore-communications-processor:LX2160A, accessed 04 March, 2018.

[31] E. Eichelberger and T. Williams, "A logic design structure for lsi testing," in *Proc. of 14th Design Automation Conference*, 1977.

[32] E. McCluskey, "Built-in self test structures," in *Proc. of International Test Conference*, 1985.

[33] Suk and Reddy, "A march test for functional faults in semiconductor random access memories," in *Proc. of IEEE Transactions on Computers*, Dec 1981.

[34] G. Hetherington, T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan, and J. Rajski, "Logic bist for large industrial designs: real issues and case studies," in *Proc. of IEEE Internation Test Conference*, Sep 1999.

[35] "Ieee standard for test access port and boundary-scan architecture." http://ieeexplore.ieee.org/document/6515989/, accessed 05 March, 2018.

[36] J. Andrews, "An embedded jtag, system test architecture," in *Proc. Electro/94 International Conference*, 1994.

[37] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Kluwer Academic Publishers, 2005.

[38] A. Krstic and K. T. Cheng, *Delay Fault Testing for VLSI Circuits*. Kluwer Academic Publishers, 1998.

[39] Y. Levendel and P. R. Menon, "Transition faults in combinational circuits: Input transition test generation and fault simulation," in *Proc. of International Fault Tolerant Computing Symposium*, 1986.

[40] G. L. Smith, "Model for delay faults based upon paths," in *Proc. of ITC*, pp. 342–349, 1985.

[41] K. Heragu, J. H. Patel, and V. D. Agrawal, "Segment delay faults: A new fault model," in *Proc. of VTS*, pp. 32–39, 1996.

[42] A. van de Goor, *Testing Semiconductor Memories: Theory and Practice*. Wiley, 1991.

[43] M. Sauer, I. Polian, M. E. Imhof, and A. Mumtaz, "Variation-aware deterministic atpg," in *Proc. of ETS*, 2014.

[44] R. Tayade, S. Sundereswaran, and J. Abraham, "Small-delay defect detection in the presence of process variations," in *Proc. of ISQED*, 2007.

[45] P.-H. Su and Y. Li, "A systematic approach to correlation analysis of in-line process parameters for process variation effect on electrical characteristic of 16-nm hkmg bulk finfet devices," in *IEEE Transactions on Semiconductor Manufacturing*, 2016.

[46] E. S. Avila, J. C. Tinoco, A. G. Martinez-Lopez, M. A. Reyes-Barranca, and A. Cerdeira, "Parasitic gate resistance impact on triple-gate finfet cmos inverter," in *IEEE Transactions on Electron Devices*, 2016.

[47] X. Lin, K. Tsai, C. Wang, M. Kassab, J. Rajski, T. Kobayashi, R. Klingenberg, Y. Sato, S. Hamada, and T. Aikyo, "Timing-aware atpg for high quality at-speed testing of small delay defects," in *Proc. of ATS*, 2006.

[48] "Tessent fastscan." https://www.mentor.com/products/silicon-yield/products/fastscan, accessed 10 March, 2018.

[49] "Tetramax atpg." https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/test-automation/tetramax-atpg.html, accessed 10 March, 2018.

[50] L. W. Nagel and D. O. Pederson, "Spice: Simulation program with integrated circuit emphasis," in *Memorandum No. ERL-M382, University of California, Berkeley*, Apr. 1973.

[51] "Cadence spectre." https://www.cadence.com/content/cadence-www/global/en_US/home/tools/custom-ic-analog-rf-design/circuit-simulation/spectre-circuit-simulator.html, accessed 10 March, 2018.

[52] R. Chadha and J. Bhasker, *Static Timing Analysis for Nanometer Designs, ISBN 978-0-387-93819-6*. Springer, 2009.

[53] C. Visweswariah, K. Ravindran, K. Kalafala, S. Walker, S. Narayan, D. Beece, J. Piaget, N. Venkateswaran, and J. Hemmett, "First-order incremental block-based statistical timing analysis," in *IEEE Trans. on Computer-Aided Design of ICs and Systems*, 2006.

[54] "Ieee standard for integrated circuit (ic) delay and power calculation system," in *DOI: 10.1109/IEEESTD.1999.91518*, 1999.

[55] D. Blaauw, V. Zolotov, and S. Sundareswaran, "Slope propagation in static timing analysis," in *Proc. of IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1180–1195, Volume 21, Oct 2002.

[56] D. Blaauw, K. Chopra, A. Srivastava, and L. Scheffer, "Statistical timing analysis: From basic principles to state of the art," in *Proc. of IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 589–607, Volume 27, April 2008.

[57] "1497-2001 - ieee standard for standard delay format (sdf) for the electronic design process." http://ieeexplore.ieee.org/document/972829/, accessed 10 March, 2018.

[58] "Tempus timing signoff solution." https://www.cadence.com/content/cadence-www/global/en_US/home/tools/digital-design-and-signoff/silicon-signoff/tempus-timing-signoff-solution.html, accessed 10 March, 2018.

[59] A. Srivastava, V. Singh, A. Singh, and K. Saluja, "A reliability-aware methodology to isolate timing-critical paths under aging," in *Journal of Electronic Testing: Theory and Application*, pp. Volume 33, Issue 6, pp 721–739, December 2017.

[60] I. Pomeranz and S. Reddy, "Hazard-based detection conditions for improved transition fault coverage of scan-based tests," in *IEEE Trans. On VLSI Systems*, pp. vol. 18, 333–337, April 2009.

[61] D. Bhattacharya, P. Agrawal, and V. D. Agrawal, "Test pattern generation for path delay faulte using binary decision diagrams," in *IEEE Transactions on Computers, vol. 44, no. 3, pp. 434-447*, Mar. 1995.

[62] S. Bose, P. Agrawal, and V. D. Agrawal, "Generation of compact delay tests by multiple path activation," in *Proceedings of International Test Conference, pp. 714-723*, 1993.

[63] C.-A, Cheng, and S. K. Gupta, "Test generation for path delay faults based on satisfiability," in *Proceedings of Design Automation Conference*, 1993.

[64] I. Pomeranz and S. M. Reddy, "Design-for-testability for improved path delay fault coverage of critical paths," in *Proc. of VLSID*, 2008.

[65] S. Pateras, "Achieving at-speed structural test," in *Design & Test of Computers*, pp. 26–33, 2003.

[66] P. Varma, "On path delay testing in a standard scan environment," in *Proc. of ITC*, pp. 164–173, 1994.

[67] K. Fuchs, F. Fink, and M. Schulz, "Dynamite: an efficient automatic test pattern generation system for path delay faults," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1991.

[68] C. J. Lin and S. M. Reddy, "On delay fault testing in logic circuits," in *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 694–703, 1987.

[69] L.-T. Wang, C.-W. Wu, and X. Wen, *VLSI Test Principles and Architectures*. Morgan Kaufmann, July 2006.

[70] C. J. Lin and S. Reddy, "On delay fault testing in logic circuits," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1987.

[71] P. Gupta and M. S. Hasiao, "Alaptf: A new transition fault model and the atpg algorithm," in *Proc. of ITC*, 2004.

[72] M. E. Amyeen, S. Venkataraman, A. Ojha, and S. Lee, "Evaluation of the quality of n-detect scan atpg patterns on a processor," in *Proc. of ITC*, 2004.

[73] K.-Y. Liao, S.-C. Hsu, and J. C.-M. Li, "Gpu-based n-detect transition fault atpg," in *Design Automation Conf.*, 2013.

[74] C.-W. Tseng and E. J. McCluskey, "Multiple-output propagation transition fault test," in *Proceedings of ITC*, pp. 358-366, 2001.

[75] X. Lin, M. Kassab, and J. Rajski, "Test generation for timing-critical transition faults," in *Proc. of ATS*, pp. 493–500, 2007.

[76] N. Devta-Prasanna, S. K. Goel, A. Gunda, M. Ward, and P. Krishnamurthy, "Accurate measurement of small delay defect coverage of test patterns," in *Proc. of ITC*, 2009.

[77] M. Tehranipoor, K. Peng, and K. Chakrabarty, *Test and Diagnosis for Small-Delay Defects, DOI 10.1007/978-1-4419-8297-12.* Springer Science+Business Media, 2011.

[78] A. L. Crouch, "Exploring the basic of ac scan," in *EE Evaluation Engineering*, 2004.

[79] N. Touba and E. McCluskey, "Applying two-pattern tests using scan-mapping," in *Proc. of VTS*, 1996.

[80] "Mentor graphics." http://www.mentor.com/products, accessed 12 July, 2017.

[81] I. Pomeranz and S. M. Reddy, "Generation of functional broadside tests for transition faults," in *Computer-Aided Design of Integrated Circuits and Systems*, pp. 2207–2218, 2006.

[82] N. Devtaprasanna, A. Gunda, P. Krishnamurthy, S. Reddy, and I. Pomeranz, "Methods for improving transition delay fault coverage using broadside tests," in *Proc. of ITC*, 2005.

[83] "Cadence design systems." http://www.cadence.com/products, accessed 12 July, 2017.

[84] L. C. Wang, J. J. Liou, and K. T. Cheng, "Critical path selection for delay fault testing based upon a statistical timing model," in *IEEE Trans. on CAD of IC and Systems*, pp. 1550–1565, 2004.

[85] J. J. Liou, K. T. Cheng, and D. A. Mukherjee, "Path selection for delay testing of deep sub-micron devices using statistical performance sensitivity analysis," in *Proc. of VTS*, pp. 97–104, 2000.

[86] K. Bowman, "A circuit-level perspective of opportunities and limitations for gigascale integration," in *Ph.D dissertation, Georgia Institute of Technology*, 2001.

[87] G. G. Lopez, "The impact of interconnect process variations and size effects for gigascale integration," in *Ph.D dissertation, Georgia Institute of Technology*, 2009.

[88] W. M. Yee, M. Paniccia, T. Eiles, and V. Rao, "Laser voltage probe (lvp): a novel optical probing technology for flip-chip packaged microprocessors," in *Proc. of Int. Symposium on the physical and failure analysis of ICs*, pp. 15–20, 1999.

[89] A. Srivastava, A. Aggarwal, and R. Bakhshi, "Effect of static stress in burn-in environment on yield of complex designs," in *Proc. of IEEE Workshop on RTL and High Level Testing*, 2015.

[90] A. Srivastava, V. Singh, A. Singh, and K. Saluja, "Path-based approach to identify timing critical paths under aging," in *IEEE Workshop on RTL and High Level Testing*, 2016.

[91] M. Ahmadi and R. Jafari, "A novel technique to detect aging in analog/mixed-signal circuits," in *Proc. of IEEE EWDTS*, 2016.

[92] "Effective current source model." https://www.cadence.com/content/cadence-www/global/en_US/home/alliances/standards-and-languages/ecsm-library-format.html, accessed 22 October, 2017.

[93] K. Jeppson and C. Svensson, "Negative bias stress of mos devices at high electric fields and degradation of mnos devices," in *Journal of Applied Physics*, 1977.

[94] M. Mahmoud and N. Soin, "Bti lifetime reliability of planar mosfet versus finfet for 16 nm technology node," in *Proc. of IEEE IPFA*, 2016.

[95] G. Groeseneken, F. Crupi, A. Shickova, S. Thijs, D. Linten, B. Kaczer, N. Collaert, and M. Jurczak, "Reliability issues in mugfet nanodevices," in *Proc. of IEEE IRPS*, 2008.

[96] . B. Kahng and S. Muddu, "Efficient gate delay modeling for large interconnect loads," in *Proc. of IEEE MCMC*, 1996.

[97] C. Ma, H. J. Mattausch, M. Miyake, T. Iizuka, M. M.-Mattausch, K. Matsuzawa, S. Yamaguchi, T. Hoshida, M. Imade, R. Koh, T. Arakawa, and J. He, "Compact reliability model for degradation of advanced p-mosfets due to nbti and hot-carrier effects in the circuit simulation," in *Proc. of IEEE IRPS*, 2013.

[98] J. Lee, I. Park, , and J. McCluskey, "Error sequency analysis," in *Proc. of IEEE VLSI Test Symp.*, 2008.

[99] M. Tehranipoor, H. Salmani, and X. Zhang, *Counterfeit ICs: Path-Delay Fingerprinting, DOI 10.1007/978-3-319-00816-5_11.* Springer International Publishing Switzerland, 2014.

[100] X. Zhang, K. Xiao, , and M. Tehranipoor, "Path-delay fingerprinting for identification of recovered ics," in *Proc. of IEEE International Symp. on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 3-5 Oct., 2012.

[101] Y. Jin, , and Y. Makris, "Hardware trojan detection using path delay fingerprint," in *Proc. of IEEE International Workshop on Hardware-Oriented Security and Trust*, 2008.

[102] W.-S. Liao, "Investigation of reliability characteristics in nmos and pmos fin-fets," in *IEEE Electron Device Letters*, July 2008.

[103] S.-Y. Kim, "Negative bias temperature instability (nbti) of bulk finfets," in *Proc. of IEEE IRPS*, 2005.

[104] S. Bahukudumbi and K. Chakrabarty, "Power management using test-pattern ordering for wafer-level test during burn-in," in *IEEE Transactions on VLSI*, 2009.

[105] D. Kurz, "An advanced area scaling approach for semiconductor burn-in," in *Microelectronics Reliability, vol. 55, pp. 129*, 2015.

[106] D. Wolpert and P. Ampadu, *Managing Temperature Effects in Nanoscale Adaptive Systems, DOI 10.1007/978-1-4614-0748-5_2*. Springer Science+Business Media, 2012.

[107] P. C, J. John, K. Klein, J. Teplik, J. Caravella, J. Whitfield, K. Papworth, and S. Cheng, "Reversal of temperature dependence of integrated circuits operating at very low voltages," in *Proc. of Int. Electron Devices Mtg., P.71-74*, 1995.

[108] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits: A design perspective, Chapter 9, ISBN-13: 978-9332573925*. Pearson Education, 2016.

[109] U. o. I. a. U.-C. Janak H. Patel, "Can we save energy if we allow errors in computing?." http:slideplayer.com/slide/9969908/, Last Accessed 21 July.