UNIVERSITY OF SOUTHAMPTON

FACULTY OF PHYSICAL AND APPLIED SCIENCES

Electronics and Computer Science

Fault Modelling and Accelerated Simulation of Integrated Circuits Manufacturing Defects Under Process Variation

by

Shida Zhong

Thesis for the degree of Doctor of Philosophy

March 2013

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL AND APPLIED SCIENCES
Electronics and Computer Science

Doctor of Philosophy

Fault Modelling and Accelerated Simulation of Integrated Circuits Manufacturing Defects Under Process Variation

by Shida Zhong

As silicon manufacturing process scales to and beyond the 65-nm node, process variation can no longer be ignored. The impact of process variation on integrated circuit performance and power has received significant research input. Variation-aware test, on the other hand, is a relatively new research area that is currently receiving attention worldwide. Research has shown that test without considering process variation may lead to loss of test quality. Fault modelling and simulation serve as a backbone of manufacturing test. This thesis is concerned with developing efficient fault modelling techniques and simulation methodologies that take into account the effect of process variation on manufacturing defects with particular emphasis on resistive bridges and resistive opens.

The first contribution of this thesis addresses the problem of long computation time required to generate logic fault of resistive bridges under process variation by developing a fast and accurate modelling technique to model logic fault behaviour of resistive bridges. The new technique is implemented by employing two efficient voltage calculation algorithms to calculate the logic threshold voltage of driven gates and critical resistance of a fault-site to enable the computation of bridge logic faults without using SPICE. Simulation results show that the technique is fast (on average 53 times faster) and accurate (worst case is 2.64% error) when compared with HSPICE. The second contribution analyses the complexity of delay fault simulation of resistive bridges to reduce the computation time of delay fault when considering process variation. An accelerated delay fault simulation methodology of resistive bridges is developed by employing a three-step strategy to speed up the calculation of transient gate output voltage which is needed to accurately compute delay faults. Simulation results show that the methodology is on average 17.4 times faster, with 5.2% error in accuracy, when compared with HSPICE. The final contribution presents an accelerated simulation methodology of resistive opens to address the problem of long simulation time of delay fault when considering process variation. The methodology is implemented by using two efficient algorithms to accelerate the computation of transient gate output voltage and timing critical resistance of an open fault-site. Simulation results show that the methodology is on average up to 52 times faster than HSPICE, with 4.2% error in accuracy.

Contents

Declaration of Authorship

I, Shida Zhong , declare that this thesis entitled *Fault Modelling and Accelerated Simulation of Integrated Circuits Manufacturing Defects Under Process Variation* and the work presented in it are both my own, and have been generated by me as the result of my own original research. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University;
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- Where I have consulted the published work of others, this is always clearly attributed;
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- Parts of this work have been published as listed in Section 1.7

Signed: _____ **Date:** _____

List of Figures

List of Tables

List of Abbreviations

ADI	Analogue Detectability Interval
ATPG	Automatic Test Pattern Generator
BSIM	Berkeley Short-channel IGFET Model
CAD	Computer Aided Design
C-ADI	Covered Analogue Detectability Interval
CMOS	Complementary Metal Oxide Semiconductor
DFT	Design for Test
D2D	Die-to-Die
DSM	Deep Submicron
EDA	Electronic Design Automation
FC	Fault Coverage
G-ADI	Global Analogue Detectability Interval
GIDL	Gate-Induced-Drain-Leakage
IC	Integrated Circuit
IDDQ	Supply current (I_{dd}) in the quiescent state
ITRS	International Technology Roadmap for Semiconductors
LF	Logic Fault
LOC	Launch-on-Capture
LOS	Launch-on-Shift
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
nm	nanometre
NMOS	N-channel MOSFET
OP	Observation Point
PMOS	P-channel MOSFET
PTM	Predictive Technology Model
PV	Process Variation
PVT	Process, Voltage and Temperature (variations)
RAM	Random Access Memory
SPICE	Simulation Program with Integrated Circuit Emphasis
SRAM	Static Random Access Memory
TS	Test Set
WID	Within-Die

Acknowledgements

First of all, I would like to thank my supervisor Prof. Bashir M. Al-Hashimi for his support and guidance in this project. Also, I sincerely thank my co-supervisor Dr. Saqib Khursheed for his assistance and suggestions since the beginning of this project. I am grateful for the funding and research facilities provided by the School of Electronics and Computer Science at the University of Southampton and the Engineering and Physical Sciences Research Council (EPSRC).

I am thankful to all my co-authors and all those people who guided and supported me technically during my Ph.D. These include Prof. Mark Zwolinski, Prof. David Flynn, Dr. Harry Oldham (Fellow, ARM), Dr. Robert Aitken (Fellow, ARM), Prof. Sudhakar Reddy (University of Iowa), Prof. Krishnendu Chakrabarty (Duke University) and Prof. Sandip Kundu (University of Massachusetts) for their valuable input. I also thank Dr. Matthew Swabey and Dr. Peter Wilson for their support with the software and server that were necessary for the project.

This work has directly and indirectly benefited from discussions, advice and feedback from colleagues and friends to whom I would like to greatly acknowledge. I am thankful to Dr. Rishad A. Shafik, Dr. Mustafa Imran Ali, Dr. Amit Acharyya, Dr. Ke Li, Sheng Yang, Fan Lu, Dr. Jatin Mistry, Dr. Evangelos Mazomenos, Hamed Shahidipour, Jedrzej J Kufel, and Luis A Maeda-Nunez for their helpful discussions and feedback.

Finally, I would like to thank my parents (Yanqun Zhong and Yanjun Chen), my lovely wife (Dandan Liang), my sisters (Kaixuan Zhong and Xiaoxuan Zhong) and my brother (Shiming Zhong) for their love, patience, support and understanding throughout my Ph.D over the past three years.

Chapter 1

Introduction

The aim of manufacturing test of digital integrated circuits (ICs) is to ensure that the design operates correctly and meets the desired specification to prevent delivery of defective ICs to the customers. Manufacturing test of an IC is the process of exercising the circuit with test patterns, collections of logic 1s and 0s, and comparing the circuit response with the expected response. If the comparison does not match and an error is propagated to the output of the circuit, the circuit is said to be faulty; otherwise, the circuit is fault-free.

Transistor miniaturisation (<100-nm) has enabled integration over the billion-transistor count per IC, but this increased integration capacity is creating new test challenges for ICs. One important challenge for improving the quality of manufactured ICs, as highlighted by ITRS-2010, is mitigating the effect of fabrication process variation. Semiconductor manufacturing test is affected by process variation as demonstrated by recent academic and industrial research [1, 2]. Running tests using existing methods and without considering process variation leads to defects being missed during test. Manufacturing test employs fault models for testing digital ICs. Fault models are used to determine the error of the corresponding defects at the output of the circuit under test. Fault models serve as a backbone of manufacturing test to improve test quality and reliability, and to reduce test cost. In order to develop new and efficient test methods capable of mitigating the impact of process variation on deep submicron (DSM) defects, this thesis focuses on investigating and developing fault models and accelerated simulation methodologies targeting manufacturing defects. The developed fault models and accelerated simulation methods should take into account the effect of process variation accurately and efficiently, leading to improved test quality of ICs.

The aim of this chapter is to provide preliminary information for the subsequent chapters in the thesis. Manufacturing process variation is discussed in Section 1.1. Section 1.2 summarises the most important deep submicron manufacturing defects which are targeted in this research. Section 1.3 provides the background on fault models and fault

simulation while Section 1.4 discusses the test methods which are targeted in this thesis. The motivation for the research in subsequent chapters is outlined in Section 1.5. The outline of this thesis is given in Section 1.6, and finally Section 1.7 presents the list of publications generated from this research.

1.1 Process Variation

CMOS technology has scaled down to a level where transistor gate length is only tens of nanometres, which is referred to as Deep Submicron (DSM) technology. In DSM manufacturing test, transistor parameters (oxide thickness, gate length) are scaled to nanometre level which means that a small variation in the transistor parameters will have an impact on defect behaviour leading to loss of test quality [2, 3]. As silicon manufacturing processes scale to and beyond the 65-nm node, process variation can no longer be ignored [4, 5, 6, 7, 8, 9]. A recent study based on ISCAS 85, 89 benchmarks and a 45-nm gate library has shown that tests generated for the nominal scenario without considering process variation can lead to 10% loss of test quality due to additional faults when considering resistive bridge defects [2].

In DSM, the gate oxide (or dielectric) of the transistors is scaled down to only ten atom layers, therefore even small variation in the transistor parameters can affect the performance of the manufactured integrated circuit. The IC parameters also become difficult to control due to the fact that more transistors are integrated into a single design. These IC parameters include the concentration of doping atoms in the N-well and P-well substrates, the thickness of the gate oxide/dielectric and the length L of a given transistor, among others [3]. It is often found that the actual values of some IC parameters in manufactured ICs from the same manufacturing fabrication process are slightly different, this is called process variation. Fabrication process variation is mainly due to random dopant distribution, sub-wavelength lithography, line edge roughness and stress engineering which lead to the change in transistor threshold voltage (V_{th}), oxide thickness (t_{ox}), effective mobility (μ_{eff}), and transistor length (L) [10, 11, 12, 13, 14]. Random dopant distribution is the fluctuations in the number and location of the doping atoms in the substrate of the transistor, which influences the transistor threshold voltage [15]. The impact of random dopant distribution on the transistor threshold voltage can lead to up to 10% of variation on 65-nm technology node [11]. The IC manufacturing process uses Photo-lithography to draw structures onto the wafer by using light to transfer a design pattern from a photomask to a light-sensitive chemical material. Sub-wavelength lithography is a technique to manufacture transistors that measure shorter than the wavelength of the light source [16]. Sub-wavelength lithography is associated with diffraction effects leading to the case where ICs differ from their original design, which mainly affects the transistor length and threshold voltage. Another effect that influences the transistor length and threshold voltage is line edge roughness which reflects

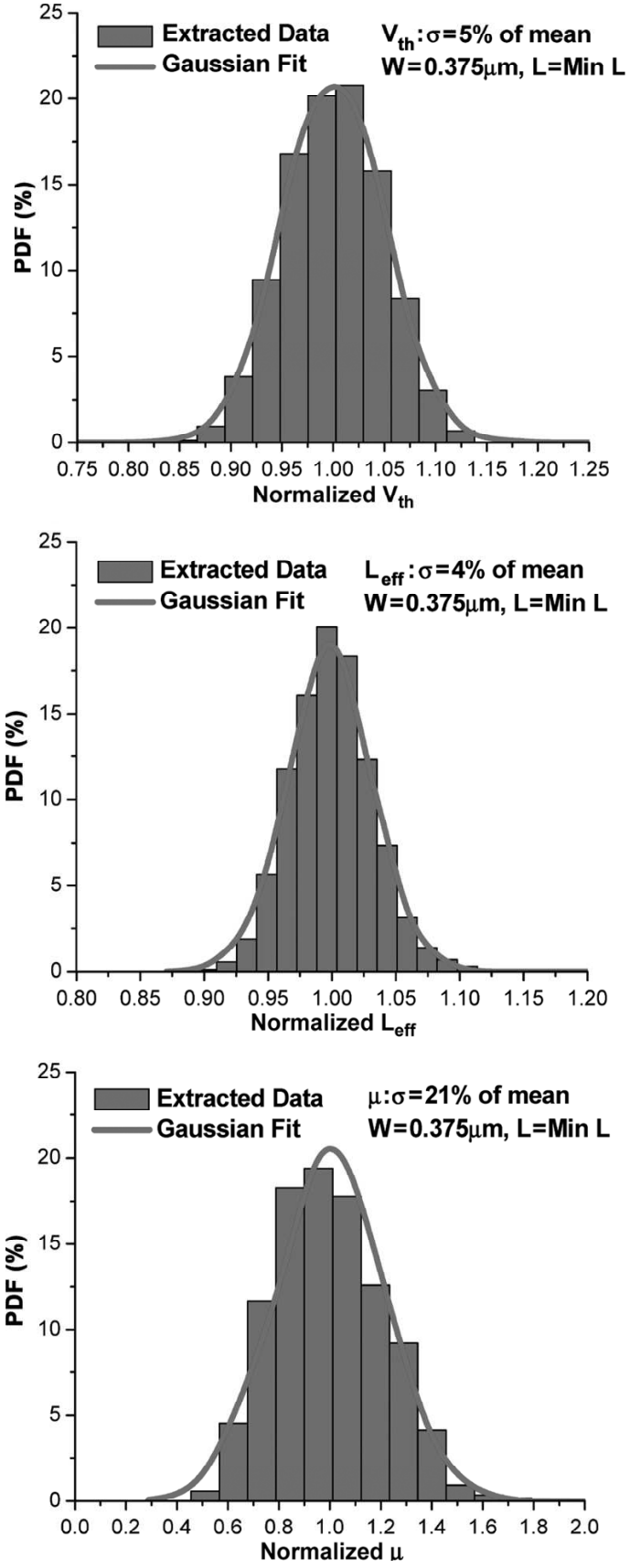


Figure 1.1: Variation of primary transistor parameters [10].

the difficulty of making the sides of the transistor channel completely smooth. It has been reported [17] that the impact of line edge roughness on transistor length leads to a variation of about 5-nm when technology scales below 100-nm node. Stress engineering is the local fluctuation of the mechanical stress which comes from the strained silicon process or the parasitic stress [10]. Stress engineering mainly affects the effective mobility of the transistors [10]. The atom-layers oxide thickness is difficult to manufacture. Furthermore, the very thin gate oxide can lead to an increase in the gate tunnelling leakage current. The change in transistor parameters (V_{th}, t_{ox}, μ_{eff}, L) due to process variation causes the transistor to deviate from its intended performance. Measured results from a recent study [10] have shown that, due to process variation, the drive current of a transistor observed on a 65-nm device can be 30% larger or smaller when compared to a transistor operating in nominal conditions.

Process variation is considered either across different dies (die-to-die) or within each die (within-die) [18, 19] depending on the distance between circuity that has correlated with IC parameters. Section 1.1.1 discusses die-to-die (D2D) process variation while Section 1.1.2 discusses within-die (WID) process variation.

1.1.1 Die-to-die Variation

Die-to-die (D2D) variation or inter-die variation represents the variation that arises between different dies in the same wafer or different wafers. D2D variation affects all transistors and interconnects on each die with the same random effect, which means that parameter variation crossing different dies or wafers is independent and random [20]. Therefore D2D process parameters variation can be modelled as independent and random parameter fluctuation. D2D variation is considered to be the leading source of variation in deep submicron CMOS ICs. A recent study used a 65-nm CMOS library with a PTM model [10, 21] to extract transistor parameters to model the effect of process variation found threshold voltage (V_{th}), gate length (L), and mobility (μ_{eff}) to be the leading sources of process variation. They are treated as independent random variables following Gaussian distribution with standard deviations (σ) of 5% for V_{th}, 4% for L and 21% for μ_{eff}. This is shown in Figure 1.1. The Gaussian distribution is shown in Eq. 1.1, where μ is the mean value and σ is the standard deviation. Another study used 35-nm and 13-nm technologies based on 100,000 statistical smaples of different transistors to study random-dopant-induced variation on threshold voltage (V_{th}) [12]. Results show that the fluctuation of V_{th} also follows Gaussian distribution.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{1.1}$$

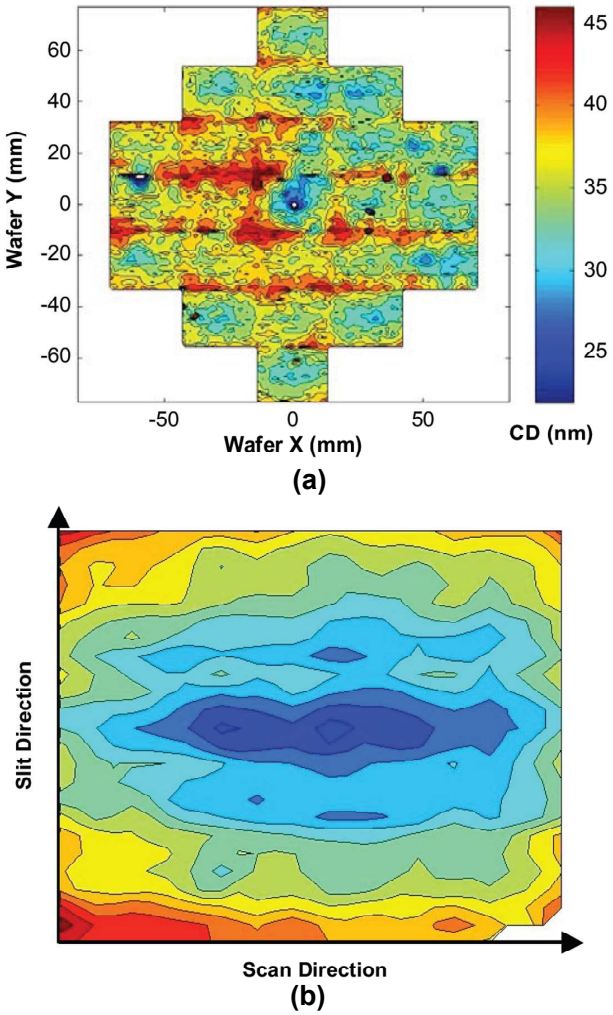


Figure 1.2: Example of systematic within-die variation: (a) Full-wafer transistor length measurements; (b) Average within-die result [22].

1.1.2 Within-die Variation

Within-die variation (WID) is also known as intra-die variation. It represents the variation that arises between different transistors and interconnects within the same die. Within-die variation has no correlation with parameter values in other dies, which means that parameter variations between different dies are considered to be random process variations, but the variation within the same die is considered to be systematically spatially correlated [20]. Variation within the same die is related to the variation in polysilicon gate dimension and depends on design implementation (layout) [18]. Transistor length (L) is considered to be the leading source of WID process variation and it has shown spatially correlated variation effects due to lithography [22, 23, 24, 25]. The result of transistor length measurement from different dies on a full 200mm wafer using 130-nm technology from a study [22] is shown in Figure 1.2(a). Figure 1.2(b) shows the average within-die measurement result of transistor length. Figure 1.1(b) clearly

shows that WID variation on transistor length is a strong systematic correlated variation, which means that the parameter variation is correlated by a function of separated distance between the transistors [22, 23, 24]. The spatial correlation function that correlates the gate length of different transistors within the same die is given by the following relationship:

$$\rho = \begin{cases} 1 - \dfrac{x}{X_L}(1-\rho_B), & x \leq X_L \quad (1.2) \\ \rho_B, & x \geq X_L \quad (1.3) \end{cases}$$

where ρ is the correlation co-efficient that relates the gate length of different transistors, X_L is the correlation length, ρ_B is the correlation baseline and x is the separation between transistors. The correlation function shown in Eq. 1.2 and Eq. 1.3 indicates that transistor parameters separated by relatively short distances x have the highest correlation ρ, and as the separation increases, the correlation ρ keeps decreasing until the transistor separation distances x is larger than the correlation length X_L, then ρ becomes constant and equals to ρ_B.

1.2 Manufacturing Defects

A manufacturing defect is an unintended difference between the manufactured ICs and the intended design. Defects can be caused by design errors, fabrication errors, fabrication defects and physical failures [26]. Design errors can be caused by incomplete or inconsistent specifications, incorrect mappings between different levels of designs and violation of design rules. Fabrication errors are normally caused by incorrect writing and shorts due to improper soldering. Fabrication defects are due to imperfect manufacturing process which may cause shorts, opens, improper doping profiles and mask alignment errors [27]. Physical failures occur during the life-time of a product which is mainly caused by wear-out effect including overstress, electromigration, corrosion and cosmic radiation.

In deep submicron CMOS, among all the defects mentioned above, bridge[1] defects and open defects are considered to be the two dominant defect types. For example, an industrial study found that in a 130-nm IC fabricated design, bridge defects and open defects account for about 58% of all defects [28]. Bridge defects are unintended connection between two nodes of a manufactured circuit. Open defects are connection breaks between two nodes in a manufactured circuit that should be connected. Such defects in ICs are due to the presence of many interconnection layers, a growing number of connections between each layer, and denser interconnection lines and they are likely to

[1]For consistency, short defects in this thesis are referred to as bridge defects.

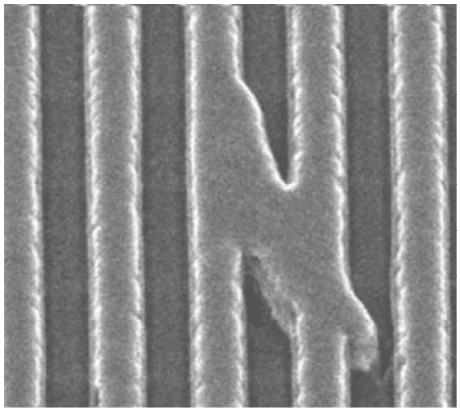


Figure 1.3: Example of bridge defects [31].

become more prominent in next generation process technologies [6]. Bridge defects and open defects change the circuit performance and logic function leading to circuit failures; in this case they are aggressively targeted during manufacturing test. Recent research also shows that logical and delay behaviour of open and bridge defects is sensitive to process variation [1, 2, 29, 30] which may lead to a loss of test quality, and therefore efficient fault models and new high quality manufacturing test methods are required to increase the quality of deep submicron ICs. This thesis considers bridge and open defects. Section 1.2.1 and Section 1.2.2 give the background on bridge defects and open defects respectively.

1.2.1 Bridge Defects

Bridge defects represent a major class of defects in deep submicron CMOS. They are formed by unwanted metal connections between two or more nodes (nets) of the circuit. A typical bridge defect is shown in Figure 1.3. The likelihood of forming a bridge is affected by many factors such as the fabrication process, the physical localities of circuit nodes and operational conditions among other factors [32]. Therefore bridges are likely to happen in any of the circuit nodes. It is summarised in [33] that there are seven possible types of physical layout locations to form bridge defects. As shown in Figure 1.4, these seven types of physical locations are Side-to-Side (S2S), Corner-to-Corner (C2C), End-of-Line (EOL), Via-to-Via (V2V), Side-to-Side over wide metal (S2SOW), Via-Corner to Via-Corner (VC2VC) and Side-to-Side with minimum width (SW2S). The study reported in [33] pointed out that knowing the bridge locations can help to generate test patterns and improve fault coverage. Another study presented in [34] shows that the location of bridge defects can be identified using Weighted Critical Area calculation, which means the calculation can find two nets that are physically close to each other and which may possibly form a bridge. Another way of determining likely locations of bridge defect is through extraction of coupling capacitance between two nets from physical layout [3]. If an extracted capacitance value is larger than a given value

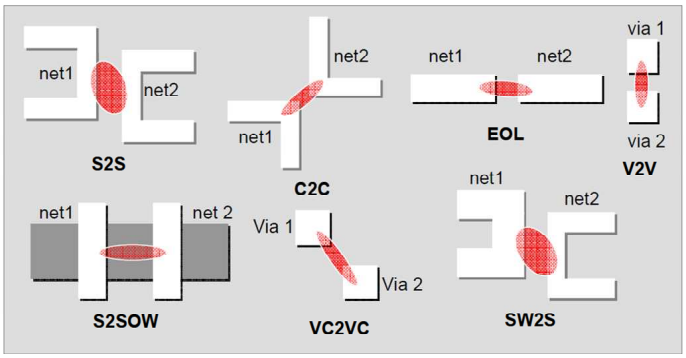


Figure 1.4: Seven possible locations for bridge defects to form [33].

(say $0.1fF$), that means the nets are close to each other and it is possible to form a bridge. The coupling capacitance of a circuit can be extracted using commercial tools, for example the "extractRC" tool from Cadence.

Bridge defects can occur in different physical layout locations, therefore a bridge is possible between any nodes of the manufactured circuit. Bridge defects can be categorised into three types, as shown in Figure 1.5. Figure 1.5(a) shows a bridge connecting two nets which are between gates. This type of bridge is called an inter-gate bridge. Inter-gate bridges only affect the circuit behaviour when two nets are driven in different logic values. As shown in Figure 1.5(a), net A is driven to logic-1 while net B is driven to logic-0. Bridges that connect internal transistors of a gate to other nets as shown in Figure 1.5(b) are called intra-gate bridges. Intra-gate bridges can also occur within one gate [32]. Another type of bridge defect is called feedback bridge, which forms between two nets (or nodes within a gate) and creates a feedback loop, as shown in Figure 1.5(c). Feedback bridges can occur on both inter-gate bridges and intra-gate bridges [35]. A number of experimental studies presented in [36, 37, 38] shows that inter-gate bridges have a much higher occurrence than intra-gate bridges. Results reported in [39] shows that the occurrence of feedback bridges is lower than non-feedback bridge. *Therefore this thesis only focuses on non-feedback and inter-gate bridge defects.* From this point, non-feedback and inter-gate bridge defect is referred to as bridge defect in this thesis.

Bridge defects can only affect circuit behaviour when two bridge nets are driven to

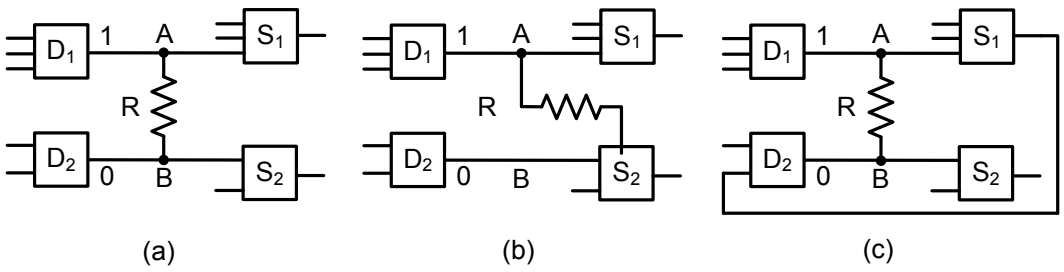


Figure 1.5: Bridge defect types: (a) Inter-gate bridge; (b) Intra-gate bridge; (c) Feedback bridge.

opposite logic values. Measurements on bridge defects found in [40, 41] have shown that most bridges have a resistance value between 0 Ω and 500 Ω. Current fault models developed for bridge defects can be summarised in two types, logic fault models without considering resistive bridge and the resistive bridge fault models. More details about fault modelling of bridge defects is presented in Section 2.1. Normally bridge defects cause static logic faults and they can be detected by using logic test, for bridge defects with higher resistance value do not have an effect on the logic state of the circuit but can cause additional delay. Therefore delay test should be used for higher resistance bridge defects [42]. Other studies have recommended IDDQ testing for bridging faults [43, 44], because the active bridge allows a current to flow from the supply voltage rail, through the gate that is driving high, through the defect and through the gate that is driving low to the ground rail. IDDQ test is used to measure the quiescent current from V_{dd} to V_{ss} when the circuit is in stable state. This additional current could be detected by IDDQ testing to detect the presence of a bridge defect. However, due to higher leakage current IDDQ test is less effective in deep submicron design [3, 45]. *Therefore this research only uses logic test and delay test for testing bridge defects.* Bridge defects have received increased attention on modelling, simulation and test generation in the past 20 years [46, 47, 48, 49, 50]. Recently, the impact of process variation on resistive bridge defects has been considered on test generation which can lead to loss of test quality [2]. Therefore new high quality manufacturing test methods and efficient fault modelling techniques targeting bridge defects are needed to minimise test escapes and to increase the test quality of DSM design. A review of bridge fault modelling, simulation and bridge defect behaviour under process variation is presented in Chapter 2.

1.2.2 Open Defects

Open defects are another major type of defects in deep submicron CMOS [51]. Open defects occur as a result of unconnected nodes that should be connected in manufacturing circuits and therefore leads to defect behaviour. Open defects in transistors within a

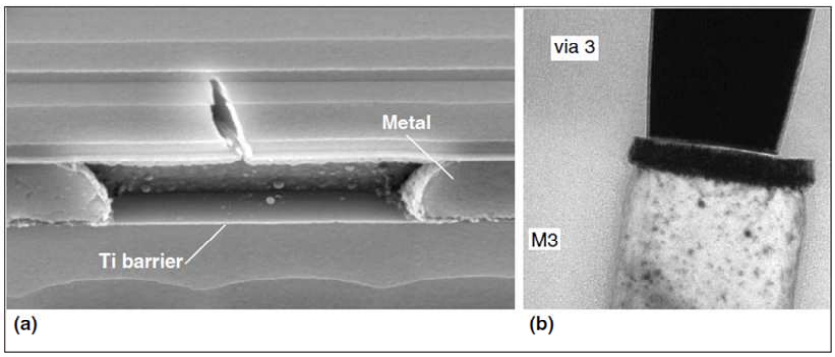


Figure 1.6: Example of open defects: (a) A cross section of metal open line; (b) A resistive open via [28].

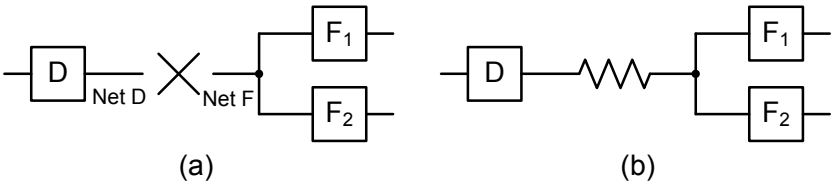


Figure 1.7: Open defects: (a)Full open defect; (b) Resistive open defect.

gate may affect noise margin and speed of operations, while in the interconnect lines of the circuit, it may increase delay and increase IDDQ. Open defects are possible to occur in any locations in manufactured circuit. In deep submicron technology, open defects are mainly caused by defective interconnect wires, contacts or vias. Interconnect wires are metal lines, contacts are generally used to connect metal layers to transistors, and vias are used to connect two different metal layers. An example of open defects caused by defective metal line and defective vias is shown in Figure 1.6. For accurate testing and diagnosis of open defects, it is very important to locate the defect position in a circuit's physical layout. According to the defect locations in a circuit, open defects can be classified into intra-gate open defects and inter-gate open defects. Intra-gate open defects occur within a logic gate while inter-gate open defects occur on the interconnect lines outside the logic gates. A study [52] reported that most open defects occur on interconnect lines outside the logic gates which means that inter-gate open defects are the most important one. Intra-gate opens have also been studied by a number of researches [41, 53, 54, 55, 56, 57, 58]. *Therefore this thesis only focuses on inter-gate open defects.* From this point on, open defects are referred to as inter-gate open defects in this thesis.

Open defects can be classified as full open which means a complete break between two nodes and resistive open which means that an extra resistance is added between two nodes. In [28] a full open is defined as the one with resistance greater than 10 MΩ and resistive open with resistance less than 10 MΩ. An example of full open and resistive open is shown in Figure 1.7. Full open causes logic failures that can be tested using static tests while resistive open shows timing-dependent effects that should be tested using delay tests [51, 59]. Full open defects completely separate a net from its driver leading to a floating net, as net F shown in Figure 1.7(a). The voltage on a floating net is affected by the following factors: the logic state of the neighbouring nets and coupling capacitances between the floating net and its neighbours, the capacitances to power supply lines and substrate, initial trapped charge, the internal capacitances of the gates driven by the floating net and gate tunnelling leakage currents [51, 60, 61]. For nanometre CMOS technologies, the reduction of oxide thickness leads to a significant increase in gate tunnelling leakage currents. Therefore the modelling of full open can be classified into two types, with and without the influence of gate tunnelling leakage currents. Without the consideration of gate tunnelling leakage, the voltage on the floating net (net F in Figure 1.7(a)) will depend on trapped charge and capacitive

coupling to the neighbouring nets and internal capacitances of driven gates [60]. With the influence of gate tunnelling leakage, voltage on the floating net will depend on the equilibrium state of charge on the floating net, where the charge is injected by driven gates' leakage currents [61]. More details about fault modelling and simulation of full open defects are discussed in Section 2.2.1.

Resistive open defects can be modelled as an extra resistance between two nodes, as shown in Figure 1.7(b). Resistive opens change the current in the circuit, therefore it can be tested by using IDDQ test [45, 62]. IDDQ test is used to measure the quiescent current from V_{dd} to V_{ss} when the circuit is in stable state. But due to high leakage current in nanometre technologies, IDDQ test becomes ineffective. The study in [63] shows that delay test is better for detecting resistive opens than IDDQ test. This is because resistive open defects cause timing-dependent behaviour which can be tested by using delay test [64]. Traditional delay test can only detect a defect when it causes a longer delay than the delay on the longest path in a fault-free design [65]. If the defect lies out of the longest paths and cannot cause higher delay than that of the longest path, it will remain undetected. Therefore, test techniques to target small delay transition faults have been developed for resistive opens [66, 67]. The small delay transition fault is the delay that introduces less than one clock cycle delay [68].

Research on resistive open defects show that the defects behaviour of resistive opens is sensitive to process and voltage variation which can affect the test quality of manufacturing test [29, 69, 70]. Therefore, efficient fault modelling, simulation and test techniques are needed. More details about open defects fault modelling and simulation are presented in Chapter 2.

1.3 Fault Models and Fault Simulation

Manufacturing test employs fault models and fault simulation to test digital ICs to ensure that they operate as intended and meet the desired specification. Fault models are used to capture the defect behaviour to determine the error of the corresponding defects at the output of the circuit under test. Fault simulation simulates the behaviour of a circuit in the presence of faults, and compares the results with fault-free circuit to determine fault detection. The following sub-sections give a brief introduction to fault models and fault simulation, and a discussion of fault modelling and fault simulation using SPICE.

1.3.1 Fault Models

Fault models are used to model the behaviour of physical defects at the device level. They are developed and employed to predict how faults occur and the impact of these faults

on circuit behaviour. The complexity of defect behaviour analysis is greatly reduced by modelling the physical defects as fault models. For example, when a defect is modelled as a logical fault, the analysis of the defect can be explained in logical terms. In addition, many physical defects can be modelled by the same fault model thus reducing the number of individual defects that have to be considered. A fault model is a formal description of how a defect causes the faulty behaviour in a circuit. It typically specifies the faulty behaviour that can occur and where such behaviour can occur. That means the fault model identifies the possible fault locations and predicts the number of possible faults in a given circuit. By knowing the number of faults and their locations in a circuit, the quality of a given test can be evaluated through the ratio of the detected number of faults to the total number of considered faults (fault domain), which is called the fault coverage. The fault coverage is represented in Eq. 1.4. One hundred per cent fault coverage means that all possible fault locations that are specified by the fault model can be detected. Fault models are important for manufacturing test because fault simulation, diagnosis and test generation are all built around fault models. Fault simulation is based on fault models and are meant to measure the fault coverage (Eq. 1.4). Test patterns are guided by fault models to generate the required test to excite and propagate the faults to primary output(s). The quality of generated test is evaluated through fault coverage.

$$FC = \frac{DetectedFaults}{TotalFaults} \times 100 \tag{1.4}$$

Fault models are used to study and simulate the behaviour of physical defects. There are many different physical defects, for example, bridges, opens, transmission gate open [71, 72], gate oxide shorts [73], threshold voltage shift [74] etc. Therefore there is no single fault model that can capture the impact of each one of these at higher level of abstraction. This is why test is generated considering a number of defects and their respective fault models. Some well-known and commonly used fault models are:

- **Stuck-at fault:** This models the defects that cause a logic signal that connected to one of the power rails, i.e., V_{dd} or GND, causing the logic node to be clamped at the voltage of the rail [59]. This fault model has two faulty behaviours which are referred to as "stuck-at 0" or "stuck-at 1". Stuck-at 0 describes a node that is connected to the ground rail while stuck-at 1 is a node that is connected to the power rail (V_{dd}). The stuck-at fault model is one of the most widely-used fault models for test generation. It can be used to detect many different physical defects, such as bridges and opens at a part of the fault domain.
- **Stuck-open and Stuck-short fault:** Stuck-open models the behaviour of a transistor where drain or source is disconnected inside a gate leading to faulty behaviour [75]. The detection of stuck-open fault need two test vectors: the first one drives the output logic of a gate to logic high or low, while the second one

uses a transistor in the pull-up or pull-down network of a gate to drive the output logic value to the opposite value [59]. Stuck-short fault models a conducting path between V_{dd} and ground that may be detected by using IDDQ test [76].

- **Bridging and Open fault:** The bridging fault models a physical scenario where interconnect lines are accidentally connected with one another, thereby deviating the circuit behaviour from ideal [59]. The bridge fault is excited only by driving two connected lines at opposite logic values. The open fault models a physical scenario where the connection breaks between the two nodes in a manufactured circuit that should be connected and therefore causes the circuit to deviate from its ideal behaviour.

- **Delay fault:** The delay fault models the behaviour caused by physical defects or process variation that cause additional circuit delay, which can lead to the circuit failing to meet the performance requirements (e.g. a specified clock period). The delay is the time interval that measures on a gate or a path from one logic state to another logic state. It can be gate transition delay or path delay. The gate transition delay fault models the behaviour of a signal while propagating through a gate if a signal violates its timing due to excessive delay through the gate [77]. The path delay fault models the cumulative delay of a path to include gates and interconnects that exist in that path of a circuit [78]. A small delay fault is a class of path delay fault and it models defects that lie outside the critical path in a circuit and normally introduce less than one clock cycle delay [68, 69]. Due to overlap between fault detection using different fault models, a test to detect delay faults may also detect stuck-at faults or bridging faults. This is why manufacturing test commonly applies tests targeting delay faults first, followed by stuck-at faults, bridge faults and finally transistor level stuck-open faults to achieve a high fault coverage in the minimum possible test application time [51].

1.3.2 Fault Simulation

Fault simulation plays an important role in manufacturing test. This is because many test generation techniques use fault simulation to evaluate a generated test set TS through fault coverage (Eq. 1.4). Fault simulation simulates the faulty output responses of a circuit based on targeted fault models, and the fault simulation results with applied test set TS are compared to results that come from the fault-free circuit with the same test TS to determine the fault coverage of TS. TS is changed according to the results of the fault simulation until the obtained coverage is considered satisfactory. A fault simulation in manufacturing test normally consist of the following five tasks:

- **Fault-free simulation:** simulation on fault-free circuit to record fault-free output response and then it is used to compare output responses from the faulty circuit.

- **Fault specification:** generate a list of targeted faults at a given circuit.
- **Fault insertion:** select a subset of faults to be simulated and use corresponding fault models to indicate the detection of faults.
- **Fault propagation:** generate faulty behaviour through fault insertion and propagate the faulty behaviour to primary outputs.
- **Fault detection:** evaluate the fault detection through fault coverage. If the fault coverage is considered satisfactory, the detected faults are discarded from the fault list to indicate that the fault can be detected with the specific test set TS. The remaining undetected faults are targeted in the next round of test generation.

Fault simulation is also used to analyse the behaviour of a circuit in the presence of defects for better modelling of defects and generation of test to detect faults caused by defects. Another application of fault simulation is to construct pre-generated databases for test generation [26]. Normally, a pre-generated database stores the output response to test set TS of every faulty circuit corresponding to targeted fault models. Using a database with every possible faulty output response to test digital ICs can help in reducing test time, for example, using a histogram database as developed in [79, 80]. It is also used for test compaction database [81, 82].

1.3.3 Fault Modelling and Simulation in SPICE

One of the commonly used approaches in fault modelling and fault simulation is SPICE. SPICE (Simulation Program with Integrated Circuit Emphasis) is a computer simulation and modelling program used to mathematically predict the behaviour of electronics circuits. It was initially developed at the University of California at Berkeley. The current commercial versions of SPICE include Synopsys HSPICE [83] and Cadence PSPICE [84], there is also an open source version of SPICE called NGSPICE [85]. SPICE can be used to simulate electrical circuits in steady-state (DC simulation), transient (TRAN simulation), and frequency domains (AC simulation). Fault modelling and fault simulation using SPICE is the most accurate method of modelling and simulating faulty behaviour of a circuit in the presence of defects. SPICE simulation is accurate because SPICE employs advanced convergence algorithms to achieve results within the specified accuracy tolerance and sophisticated semiconductor device models such as BSIM4 MOSFET models [86] to accurately simulate the device behaviour in a circuit. Fault modelling using SPICE can be summarised as:

1. Insert physical defect description into a circuit netlist;
2. Simulate circuit behaviour in the presence of defects using DC, TRAN or AC simulation;

3. Analyse faulty output behaviour of the defective circuit;
4. Create data structures or functional description as fault models to model the faulty behaviour caused by the defects;
5. Use the created data structures or functional description in fault simulation to indicate the fault detection.

Recent researches reported in [2, 10, 80] indicate that modelling the impact of process variation on deep submicron defects through SPICE is the most accurate method. SPICE uses Monte-Carlo simulation to model variation on device parameters. Monte-Carlo simulation generates random values for the targeted device parameters by following the specific distributions (i.e. Gaussian distribution as shown in Eq. 1.1) in each repeated simulation. Every repeated simulation contains a set of values for the targeted device parameters. A set of values is also called a sample. By increasing the number of samples, the simulated effect of variation can capture more possible behaviours caused by variation. Monte-Carlo simulation normally uses a large number of samples to capture the effect of variation, therefore leading to a long simulation time in SPICE. A detailed SPICE simulation flow of DC and TRAN simulation with Monte-Carlo analysis is shown in Appendix A.

1.4 Manufacturing Test

The manufacturing process of deep submicron CMOS integrated circuits (ICs) is highly complex. Due to the complexity of the manufacturing process, defects may occur and not all the manufactured designs operate correctly according to specification. Therefore, manufactured ICs need to be tested to ensure that every gate and register in the ICs operate as intended and meet the desired specification with the aim of preventing the delivery of defective parts to customers. Manufacturing test of an IC design is the process of exercising the circuit with test patterns, which are the collection of logic 1s and 0s, and comparing the circuit response with the expected response. If the comparison does not match the expected response and an error is propagated to the output of the circuit, the circuit is said to be faulty; otherwise, the circuit is considered fault-free [59]. The purpose of manufacturing test is to use fault modelling and fault simulation to increase fault coverage, therefore improving test quality. Fault coverage is the ratio percentage of detected faults to the total fault domain which is shown in Eq. 1.4. A delay in identifying and repairing a defective device during manufacturing test may lead to 10 times additional cost [87]. So manufacturing test methods that can detect manufacturing defects with higher fault coverage are essential.

The manufacturing defects that are studied in this thesis are bridge and open defects. The most commonly used manufacturing test methods for bridge and open defects can be

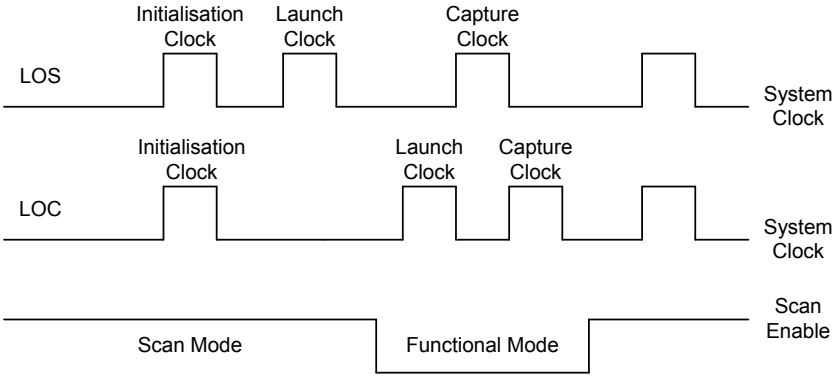


Figure 1.8: Waveforms for Launch-on-Shift and Launch-on-Capture.

divided into logic test and delay test. The following sub-sections give a brief introduction to these two test methods.

1.4.1 Logic Test

Logic test is used to detect defects that cause time-independent logic malfunction. Logic test is applied to combinational circuits by using stuck-at fault or bridge fault (see Section 1.3.1), and it can also applied to sequential circuits by using Design for Test (DFT) techniques such as scan chains. A logic test for a single stuck-at fault in a sequential circuit may require a long sequence of test vectors for detection. This is because the memory elements within sequential circuits reduce the controllability and observability of a node under test, which leads to logic faults in sequential circuits may become untestable or detectable logic faults may need a large number of test vectors, therefore leading to an increase of test cost. The memory elements such as flip-flops can be converted into scan cell by using scan chains, which allows easier access to all nodes in sequential circuits; therefore sequential designs can be treated as combinational during logic test mode.

1.4.2 Delay Test

Delay test employs delay fault models to detect defects that cause time-dependent faulty behaviour such as additional circuit delay leading to the situation whereby the circuit does not meet its performance requirements. An introduction of delay fault models is presented in Section 1.3.1. Resistive bridge (Section 1.2.1) and resistive open (Section 1.2.2) are the defects that can be detected by delay fault testing. The purpose of delay fault testing is to detect defects that causes the transition gate delay or path delay to violate the timing specifications for the desired performance. Delay test requires two test vectors, one that initialises the circuit and the other that causes a transition in logic state and propagates the transition to the output. The time for the transition to

pass through the circuit is compared to the clock period by capturing the logic value at the outputs into the scan flip-flops after a time corresponding to the clock period has passed. There are two techniques available for applying delay fault testing. These techniques are called launch-on-shift (LOS) [88] and launch-on-capture (LOC) [89]. In launch-on-shift, the first vector is scanned in and the next vector is shifted by one bit from the first vector. In launch-on-capture, the first vector is scanned in and the resulting functional response is used as the second vector. Figure 1.8 shows the waveforms of the clock for launch-on-shift and launch-on-capture as well as with the corresponding scan enable signal. The use of these two techniques (LOS and LOC) depends on the circuitry and the defects that need to be tested.

1.5 Motivation and Research Aims

With continuous scaling of process technology, digital ICs offer high clock frequencies, low power and high density. However, advances in technology have also led to more manufacturing defect types with the most prominent being resistive bridges and resistive opens. It is reported in [90] that the frequency of resistive open and resistive bridge defects increases with technology scaling. As an example, an industrial study estimated resistive open and resistive bridge defects account for as much as 58% of all defects [28] found in an IC fabricated design using 130nm. Resistive open and resistive bridge defects alter the IC delay performance and change the logic function leading to IC failures, and therefore they are aggressively targeted by industry during manufacturing test. The considerable increase of resistive open and resistive bridge defects in nanometre ICs is due to the presence of many interconnected layers, a growing number of connections between each layer, and denser interconnection lines, and they are likely to become more prominent in next generation process technologies [90, 91].

Recent research shows that logic and delay behaviour of resistive open and resistive bridge is sensitive to process variation [2, 29, 30, 79, 80, 92] which may lead to test escape. The current fault modelling techniques and simulation methodologies are either time consuming or not accurate to capture process variation induced behaviour. Fault modelling through SPICE is the most accurate method of modelling fault behaviour of defects under the influence of process variation [2, 79, 80]. However, recent research demonstrates that SPICE requires a long computation time to model and simulate additional fault behaviour due to process variation. A study in [2, 3] reported that to generate a database for ISCAS 85, 89 benchmarks through SPICE when detecting resistive bridges, it took nearly a week with 8 computers working in parallel. Similarly, for the study reported in [79, 80], it took 10 days with a 32-node cluster to generate a database for simple logic gates when considering process variation. The analysis presented in Chapter 4 also shows that, using SPICE with a 65-nm gate library, the delay fault modelling of a resistive bridge under process variation takes on average

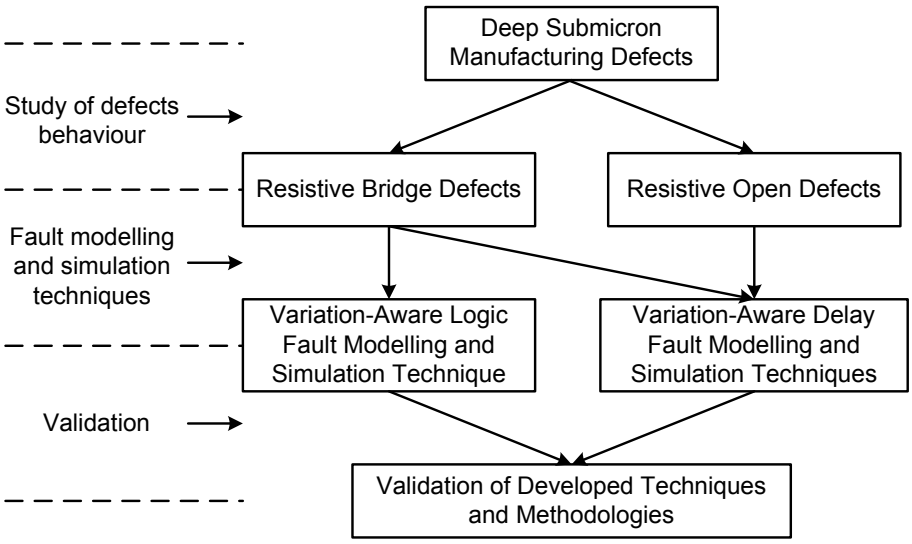


Figure 1.9: Project aims.

about 19 minutes per fault-site with 600 permutations of Monte-Carlo simulation when using a Quad-Core 2.7 GHz processor with 12 GB RAM. Therefore, new high quality and efficient fault modelling techniques and fault simulation methodologies targeting such defects are needed to minimise test escapes and to reduce manufacturing test cost. Research in variation-aware manufacturing test is still in its infancy and significant academic and novel research is still needed. This research is timely because the availability of effective and low-cost test methods developed specifically to mitigate the impact of process variation are of paramount importance if the test cost of nanometre ICs is to remain acceptable for the highly competitive microelectronics industry. The development of variation-aware manufacturing test methods requires fast and accurate fault modelling techniques and fault simulation methodologies for resistive bridge and resistive open defects to model and simulate their logical and timing behaviour under process variation. The aim of this research is to investigate the impact of process variation on fault modelling and fault simulation for resistive bridge and resistive open defects in order to develop fast and accurate process variation-aware fault modelling techniques and simulation methodologies that can be used for new manufacturing test methods, which is shown in Figure 1.9 and summarised as follows:

1. Study the defects behaviour to identify the variation-induced logic and delay faults due to process variation on DSM defects with particular emphasis on resistive bridge and open defects, which is presented in Chapter 2;

2. Develop a process variation-aware logic fault modelling and simulation technique for resistive bridge defects with low computational time for running large circuit simulations and validate the technique in comparison with HSPICE. The developed technique is presented in Chapter 3;

3. Develop a delay fault modelling and simulation methodology for resistive bridge defects with low computational time for running large circuit simulations under the influence of process variations and validate the technique in comparison with HSPICE. The proposed methodology is presented in Chapter 4;

4. For resistive open defects, develop a process variation-aware delay fault modelling and simulation methodology with low computational time for running large circuit simulations and validate the technique in comparison with HSPICE. The proposed methodology is presented in Chapter 5;

At present there is little reported process variation aware fault modelling and simulation methods for resistive open and resistive bridge defects [2, 79, 92]. The developed fault modelling techniques and simulation methodologies will facilitate the development of efficient test generation methods in terms of higher defect coverage (better test quality) with faster simulation time when compared with the state-of-the-art test methods (logic and delay) reported in [2, 29, 30, 79, 92].

1.6 Thesis Outline

This thesis has six chapters. These are organised as follows:

Chapter 1 Introduction

This chapter provides background information for the subsequent chapters in the thesis. It includes an introduction to manufacturing process variation, manufacturing defects, fault models, fault simulation and test methods that are targeted in this thesis. A list of publications generated from the presented research is included in this chapter.

Chapter 2 Literature Review

This chapter presents a detailed review of recently reported research in modelling the fault behaviour of bridge and open defects and their fault simulation techniques in nominal operating conditions. Defect behaviour due to process variation is discussed and a review of the current fault modelling and fault simulation techniques when considering process variation is presented to achieve the first objective of this thesis. This chapter also outlines a number of important research problems that are addressed in this thesis to develop efficient variation-aware fault modelling and fault simulation techniques for resistive bridge and resistive open defects.

Chapter 3 Logic Fault Modelling of Resistive Bridge Defects

This chapter presents a fast and accurate logic fault modelling technique to model the effect of process variation on resistive bridge defects, which meets the second objective

of this thesis. Voltage and temperature variation are also considered. The speedup of the presented technique is achieved by incorporating two efficient voltage approximation algorithms for calculating logic threshold voltage of driven gates (gate S_1 and S_2 in Figure 1.5(a)) and voltages (voltages of node A and B in Figure 1.5(a)) on bridge lines of a fault-site for efficient generation of logic faults and the calculation of bridge critical resistance without using SPICE simulation. The presented technique is accurate because the approximation algorithms use the most recent transistor model (BSIM4, Berkeley Short-Channel IGFET Model [86]) to calculate the voltages. Based on the developed fault modelling technique, an efficient logic fault simulator for resistive bridge defects is developed. Experiments are conducted on a 65-nm gate library (for illustration purposes), and results show that on average the proposed technique is more than 53 times faster, and in the worst case, error in bridge critical resistance is 2.64% when compared with Synopsys HSPICE.

Chapter 4 Accelerated Delay Fault Simulation of Resistive Bridge Defects

This chapter presents a delay fault simulation methodology to accelerate the computation of delay faults on resistive bridge defects when considering process variation to achieve the third objective of this thesis. The accelerated fault simulation is achieved by identifying the influential variables needed to accurately compute delay faults and then by reducing their computation time. A key identified variable is transient gate output voltage. A three-step strategy is employed to speed up calculation of this variable without losing accuracy. The presented methodology has been incorporated in an open-source SPICE (NGSPICE) with BSIM4.7 transistor model. Results based on a 65-nm gate library (for illustration purposes) show that the proposed methodology is on average 17.4 times faster than Synopsys HSPICE, with error in accuracy of 5.2%. The developed methodology is used to determine the most effective class of transition delay test classes that can achieve maximum coverage in the presence of process variation, and compares bridge resistance coverage using logic test and delay test at multiple voltage settings to identify the best voltage setting and test type for detecting resistive bridge defects.

Chapter 5 Accelerated Delay Fault Simulation of Resistive Open Defects

This chapter presents an efficient delay fault simulation methodology for resistive open defects when considering the influence of process variation to achieve the last objective of the thesis. The proposed methodology is implemented by using two efficient algorithms. The first algorithm employs the transient gate output voltage calculation technique developed in Chapter 4 to speed up the calculation of transient gate output voltage from a resistive open fault-site. The second algorithm uses an efficient approximation method to determine timing critical resistance of an open fault-site. Simulation results based on a 65-nm gate library and a 45-nm gate library show that the proposed technique is on average up to 52 times faster than HSPICE, with error in accuracy of 4.43%.

Chapter 6 Conclusion and Future Work

This chapter summarises the contributions presented in this thesis and describes how the aims of this research have been achieved. This chapter also outlines a number of research problems worthy of further investigation to achieve efficient and cost-effective manufacturing test under process variation.

1.7 Thesis Contributions

The contribution of the research work presented in Chapter 3 to Chapter 5 have been published as follows:

- Khursheed, S., **Zhong, S.**, Al-Hashimi, B. M., Aitken, R., and Kundu, S., *Modeling the Impact of Process Variation on Resistive Bridge Defects*, International Test Conference, 31^{st} Oct to 5^{th} Nov, 2010, Austin, America.
- **Zhong, S.**, Khursheed, S. and Al-Hashimi, B. M., *A Fast and Accurate Process Variation-aware Modeling Technique for Resistive Bridge Defects*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 30(11):1719-1730, Nov. 2011.
- **Zhong, S.**, Khursheed, S., Al-Hashimi, B. M., Reddy, S. M., and Chakrabarty K., *Analysis of Resistive Bridge Defect Delay Behavior in the Presence of Process Variation*, Asian Test Symposium, 21^{st} to 23^{rd} Nov, 2011, New Delhi, India.
- **Zhong, S.**, Khursheed, S., Al-Hashimi, B. M., *Delay Fault Modeling and Simulation of Resistive Open Defects Under Process Variation*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, (Under review).

Chapter 2

Literature Review

This chapter provides an overview of state-of-the-art research that is related to this thesis. The overview includes a discussion on fault models and fault simulation methods for the considered defects, namely bridge defects (Section 2.1) and open defects (Section 2.2). With regard to the focus of this thesis which is studying the impact of process variation on manufacturing test, Section 2.3 reviews the process variation induced behaviour of defects and the fault modelling techniques and simulation methodologies when considering process variation.

2.1 Bridge Faults

Bridge defects have received increased attention on modelling, simulation and test generation in the past 20 years [46, 47, 48, 49, 50, 93, 94, 95]. Fault modelling for bridge defects has been developed from simple and abstract logic fault models without considering the behaviour caused by the bridge resistance to detailed fault models that consider the complete analog behaviour caused by the resistive bridge. Therefore the bridge fault models can be categorised into two types: non-resistive bridge fault models and the resistive bridge fault models. The following sections discuss these two types of fault models (Section 2.1.1 and Section 2.1.2) and their fault simulation methods for test generation (Section 2.1.3).

2.1.1 Non-Resistive Bridge Fault Models

This section considers bridge fault models that do not take into account the analog behaviour caused by the resistive bridge, which means that the value of the bridge resistance is considered to be 0 Ω. The early and simple modelling approach to model the logic effects of a bridge was the single stuck-at fault. Later fault models began to

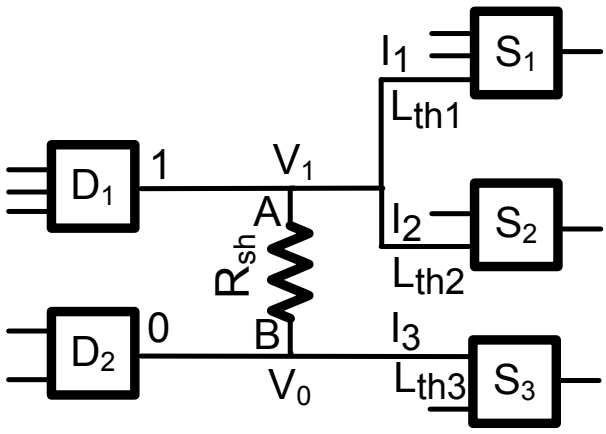


Figure 2.1: Resistive bridge forming potential circuit fault.

consider the logic state of both nets that are connected by the bridge. In general, an activated bridge fault requires that the connected bridge nets are driven to opposite logic values. As shown in Figure 2.1, net A is driven to logic-1 by driving gate D_1 while net B is driven to logic-0 by driving gate D_2. Gates S_1, S_2 and S_3 are the successor gates that are connected to the bridge nets. Simple fault models that describe the logic behaviour caused by the bridge were the wired-logic model [96] and the dominance-behaviour model [97]. The wired-logic model describes the logic state of the bridge nets in two modes: wired-AND or wired-OR. The wired-AND means that the bridge net driven to logic-0 (net B in Figure 2.1) determines the logic value of the other bridge net, while the wired-OR means that bridge net driven to logic-1 dominates the other net. As reported in [98], the wired-logic model is not accurate for COMS circuits. Unlike the wired-logic model, the dominance-behaviour model describes that either net A or net B can be the dominant net without considering their logic states. A net that is determined to be the dominant net is called the aggressor net and the other net is called the victim net, because the victim net changes its logic value due to bridge fault. Based on the dominance-behaviour model, the 4-way model was developed [99, 100]. Because there are two digital values in digital circuits, the bridge fault-site shown in Figure 2.1 can behave in four different ways: net A dominates net B with logic-1, net A dominates net B with logic-0, net B dominates net A with logic-1 and net B dominates net A with logic-0. The 4-way model only considers the logic behaviour of the bridge without relating to any of the analog behaviour of CMOS circuits.

In addition to the simple logic fault models without considering the analog behaviour of CMOS circuits, there are fault models that use the information of the drive strengths of the driving gates to determine the voltage on the bridge nets, and therefore determine the aggressor net. One of the well-known model is called the voting model [101, 102]. In the voting model, the logic value on the bridge nets is determined by identifying the stronger driving capability of the driving gates. The driving capability of a gate is determined by two key factors. The first factor is the physical parameters of the gate, such as the

structure of the transistor networks (in parallel or in series) and process parameters of the transistors (length, width etc.). The second factor is the input assignment to the gate that determines which transistor network (pull-up network or pull-down network) is conducting, and the transistors within this network are active or inactive. It has been reported [103] that the voting model is inaccurate because it assumes the value of logic threshold voltage for all inputs of all gates to be a single fixed value, for example, equal to half of the supply voltage ($V_{dd}/2$) [101]. The logic threshold voltage of a gate input is defined as the input voltage at which the output reaches half of the supply voltage, while other inputs of the gate are at non-controlling value(s) [104]. The logic threshold voltages from different inputs and different gates are not necessarily equal. Therefore using a single fixed value may lead to wrong predictions. As an example, assume that voltages on V_1 and V_0 in Figure 2.1 is $0.55V_{dd}$ due to bridge $R_{sh} = 0\ \Omega$. If the logic threshold voltage L_{th1} for the input I_1 of successor gate S_1 is $0.58V_{dd}$, then I_1 reads faulty logic value 0 due to $V_1 < L_{th1}$. Similarly, I_2 reads faulty logic 0 when $L_{th2} = 0.56V_{dd}$ ($V_1 < L_{th2}$) and I_3 reads faulty logic 1 when $L_{th3} = 0.48V_{dd}$ ($V_0 > L_{th3}$). If the logic threshold voltages (L_{th1}, L_{th2} and L_{th3}) are considered to be fixed to a single value to $0.5V_{dd}$, then only I_3 can propagate faulty logic value while I_1 and I_2 read correct logic values. Therefore the biased voting model [103] is developed to improve the voting model by considering that each gate input has a different logic threshold voltage. The biased voting model can further increase the modelling accuracy of the bridge fault by determining the logic state of the bridge nets through the driven strengths of the driving gates and the logic threshold voltages of the successor gates.

All the fault models discussed in this section did not consider the effect caused by the resistance of the bridges. In fact, the majority of the bridges have a non-zero resistance. To accurately model the fault behaviour of a bridge, the behaviour caused by the resistance of the bridge needs to be taken into account, which is discussed next.

2.1.2 Resistive Bridge Fault Models

As pointed out in Section 2.1.1, the majority of bridges have a non-zero resistance. A study reported in [40] shows that about 96% of bridges have a resistance value which is lower than 1 kΩ based on the measurement results from 14 wafers made from different batches and production lines. As reported in [50, 95], it was shown that higher bridge resistance can be detected with lower supply voltage in logic test. A study reported in [42] shows that delay test can be used to detect higher bridge resistance than using logic test. It shows that the resistance of the bridge can change the circuit behaviour. Therefore it is important to model the effect caused by the resistance of the bridges.

The resistance of a bridge is an unknown parameter that needs to be considered for the whole continuum of bridge resistance values ($[0\Omega,\infty)$). Studies on modelling the effect of resistive bridge and the developed fault model, fault simulation and test generation

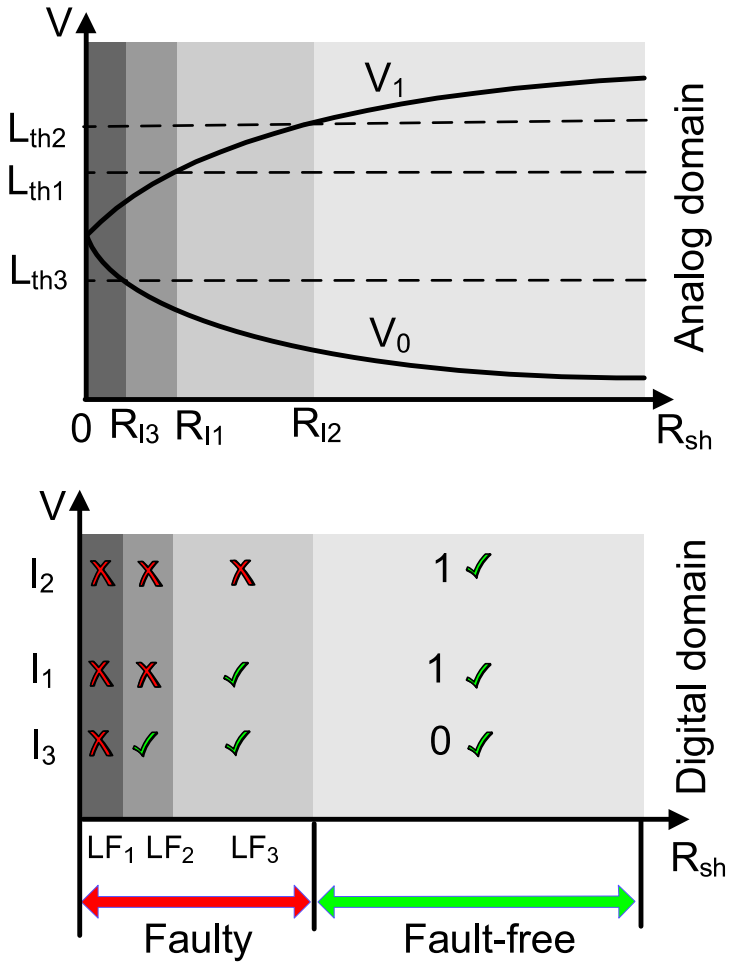


Figure 2.2: Bridge fault example and its behaviour in analog and digital domain [50, 51].

tools on low power deep submicron designs have been proposed in recent years [47, 50, 93, 94, 95, 105]. The model proposed in [47] called the parametric resistive bridge fault model is capable of efficiently and accurately capturing the effect of a circuit that is connected by a bridge with random resistance value. The resistive bridge fault models divide the whole continuum of bridge resistance values into a finite number of discrete intervals. This is based on the fact that different values of a bridge resistance can lead to different voltages on the bridge nets which vary from 0 V (logic-0) or V_{dd} (logic-1) to some intermediate values. The resistive bridge fault model associates a resistance interval to each logic fault which is determined by the drive strength of the driving gates, the bridge resistance and the logic threshold voltages of the successor gates.

Based on the concept of the parametric resistive bridge fault model, a typical bridge fault behaviour in nominal scenario is illustrated in Figure 2.2 for a bridge fault-site shown in Figure 2.1. Figure 2.1 shows a resistive bridge R_{sh}, D_1 and D_2 are the gates driving the bridged nets, while S_1, S_2 and S_3 are the successor gates. Let us assume that the output of D_1 is driven high and the output of D_2 is driven low. The dependence of the voltage levels on the outputs of D_1 (V_1) and D_2 (V_0) on the equivalent resistance

of a physical bridge is shown in Figure 2.2 (based on SPICE simulation using 65-nm library). To translate this analog behaviour into the digital domain, the input threshold voltage levels L_{th1}, L_{th2} and L_{th3} of the successor gates S_1, S_2 and S_3 have been added to the plot shown in Figure 2.2. For each value of the bridge resistance $R_{sh} \in [0, \infty)$, the logic values read by inputs I_1, I_2 and I_3 can be determined by comparing V_1 and V_0 with the input threshold voltage of the corresponding input. These values are shown in the second part of Figure 2.2 (marked as "digital domain"). Crosses are used to mark the faulty logic values and ticks to mark the correct ones. It can be seen that, for bridges with $R_{sh} > R_{I2}$, the logic behaviour at the fault site is fault-free (all inputs read the correct value), while for bridges with R_{sh} between 0 and R_{I2}, one or more of the successor inputs are reading a faulty logic value. A number of bridge resistance intervals can be identified based on the corresponding logic behaviour. For example, bridges with $R_{sh} \in [0, R_{I3}]$ exhibit the same faulty behaviour in the digital domain (all successor inputs read the faulty logic value), similarly, for bridges with $R_{sh} \in [R_{I3}, R_{I1}]$, successor gates S_1 and S_2 read the faulty value, while S_3 reads the correct value. For the resistance range $R_{sh} \in [R_{I1}, R_{I2}]$, all successor gates other than S_2 read the correct logic value, and finally for $R_{sh} > R_{I2}$ all the successor gates read the correct logic value. Consequently, each interval $[R_a, R_{a+1}]$ corresponds to a distinct logic behaviour occurring at the bridge fault-site. The R_{sh} value corresponding to R_{I2} is normally referred to as "critical resistance" (R_{crit}), as it represents the crossing point between faulty and correct logic behaviour. Methods for determining the critical resistance have been presented in several publications [49, 106]. These distinct logic behaviours at the bridge fault-site are referred to as *Logic Faults*, where each individual logic fault comprises of the following variables: boolean input to the driving gates, boolean values interpreted by the driven inputs of the successor gates (I_1, I_2 and I_3, as in Figure 2.1) and the covered resistance range of the bridge R_{sh}. Figure 2.2 shows three logic faults (marked as "LF_1", "LF_2" and "LF_3") corresponding to distinct logic behaviours occurring at the bridge fault-site. These logic faults associated with a range of bridge resistance are called the Analogue Detectability Interval (ADI) [93, 105]. The fault domain of a bridge fault-site that comprises of the union of the ADI is called the Global Analogue Detectability Interval (G-ADI). Basically, G-ADI represents the entire detectable resistance range of the bridge defect. With a given test set TS, the Covered Analogue Detectability Interval (C-ADI) represents the union of one or more ADIs that can be detected by TS. Therefore the fault coverage FC of the resistive bridge faults caused by a physical bridge defect (b) with a given test set TS is defined in Eq 2.1 [3, 49]. For a circuit with a set of bridge locations B, the fault coverage is represented in Eq 2.2.

$$FC(b, TS) = \frac{\|C - ADI(b, TS)\|}{\|G - ADI(b)\|} \tag{2.1}$$

$$FC(B, TS) = \frac{\sum_{b \in B} FC(b, TS)}{\|B\|} \tag{2.2}$$

As illustrated in Figure 2.1 and Figure 2.2, the resistance of a bridge affects the circuit behaviour and the test results; therefore the study in this thesis is based on the resistive bridge fault model in order to better capture the effects caused by process variation. A review of fault simulation and test generation using the resistive bridge fault model is discussed next.

2.1.3 Fault Simulation of Resistive Bridges

This section discusses the available fault simulation techniques for resistive bridge when operating in nominal conditions. The quality of a generated test set TS is evaluated through fault simulation by comparing the simulated faulty output responses to the fault-free output responses to determine the fault coverage (Eq. 1.4). Normally bridge defects cause static logic faults and they can be detected by using logic test through logic fault simulation. Bridges with large resistance value do not have an effect on logic state of the bridge nets, and therefore cannot be detected through logic test, but these large resistance bridges can cause additional delay. Therefore delay test is used to cover the bridge induced delay faults. This section discusses the fault simulation techniques in manufacturing test for resistive bridge that include logic fault simulation and delay fault simulation.

2.1.3.1 Logic Fault Simulation

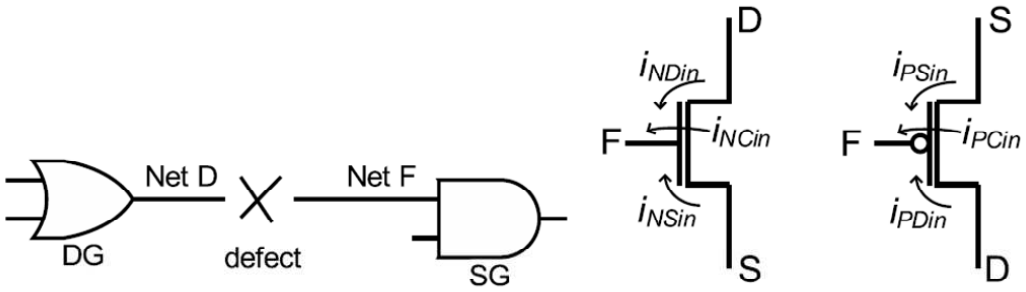
Logic fault simulation simulates defect behaviour that cause logic malfunction without considering the timing effect. Logic test through logic fault simulation is the main test technique applied to bridge defects because about 96% of the bridge resistance values are lower than 1 kΩ [40] which cause static logic faults. In logic test, fault simulation and ATPG tools for resistive bridges have been proposed in recent publications [50, 94, 95, 106, 107, 108]. These studies are all based on the concept of Analogue Detectability Interval (ADI) [93, 105] to run fault simulation and generate test patterns. The study reported in [106] identifies the logic fault only associated with the largest resistance interval and determines the corresponding test pattern. Another study [107] proposed a sectioning approach to consider all the sections (resistance intervals) that correlated to different logic faults. This approach improves the test quality when compared with the method reported in [106], but the number of considered faults also increases. A method proposed in [108] combined the advantages of the interval based [106] and the sectioning approach [107] into a more efficient test generation procedure by targeting the logic fault of the section with the highest resistance values first. Fault simulation is then used to identify all other sections covered by the test pattern and only the resistance intervals that have not been covered are considered for test generation. Targeting sections with the highest resistances first increases the probability of finding a test pattern that covers the whole resistance range early, which then improves test efficiency by reducing fault

simulation time. In order to reduce the simulation time when running test for resistive bridge, a Fitted Model that used a method of estimating the currents and voltages through the bridge net and calculating the bridge resistance value according to the successor gate's logic threshold voltage is proposed in [94]. In [94], the logic threshold voltages of the successor gate are obtained from SPICE results. This model uses Shockley Transistor Model [109] to derive an equation by using fitted parameters obtained from SPICE results to calculate the critical resistance of a bridge fault-site, instead of using SPICE simulation. The Fitted Model can reduce the fault simulation time but it is technology and gate library dependent, which means that a new set of fitted parameters is needed for different technology and gate libraries. Further discussion of using Fitted Model to model the effect of process variation is discussed in Chapter 3. Test methods proposed in [50, 95] are used to extend the test pattern generation to detect bridge defects when the design is using multi-voltage and different temperature settings. All the methods discussed above do not take into account the effect of process variation. A discussion about fault modelling and fault simulation in manufacturing test when considering process variation is presented in Section 2.3.

2.1.3.2 Delay Fault Simulation

Delay fault simulation simulates defect behaviour that cause additional circuit delay which affects performance requirement. Bridge with a large resistance value (generally greater than 1kΩ) does not affect the static logic state of the bridge net, but these can change the timing performance of the circuit, therefore delay fault test is used for such bridge defects. As reported in [42], delay test is classified into three classes (Class-I, Class-II and Class-III) of transition delay test, each of which depends on the location and the number of the transition signals applied. Class-I is defined as a transition signal applied to only one of the bridge nets while the other net is kept at a constant value, Class-II is transition signals applied to both the bridge nets and Class-III is transition signal at the gate that is driven by the bridge net. Through simulations in nominal operating conditions using a number of resistive bridge fault-sites, it was shown in [42] that test through delay fault simulation covers higher bridge resistance range than logic fault test, this study also compares the resistance coverage of each of the three delay test classes and shows that Class-I can cover higher resistance range. Another study [65] reports that test through delay fault simulation detects bridge resistance values greater than those detected by logic test. It also points out that depending on the logic state of the bridge net and the transition signal applied, the bridge can either create additional circuit delay or speed up the circuit.

Process variation can lead to additional faults, therefore when using delay test on large circuits, it becomes a challenging problem to model the bridge defect behaviour using SPICE. More discussion on delay fault modelling and simulation when considering process variation is presented in Section 2.3.2.



(a) Full open defect

(b) Tunneling leakage current affects the floating net

Figure 2.3: The impact of gate tunnelling leakage on a full open defect [3].

2.2 Open Faults

This section discusses the fault modelling and simulation of another dominant defect type in deep submicron CMOS: open defect. Open defect occurs as a result of unconnected nodes in a manufactured circuit that was designed to be connected and therefore leads to defective behaviour. It can be classified as full open with resistance greater than 10 MΩ and resistive open with resistance less than 10 MΩ [28]. Full open causes logic failures that can be tested using static test while resistive open show timing-dependent effects that should be tested using delay test [51, 59]. Section 2.2.1 discusses the available fault models for full open, while Section 2.2.2 discusses the available fault models for resistive open, and Section 2.2.3 discusses the fault simulation in manufacturing test in nominal operating conditions for open defects.

2.2.1 Full Open Fault Models

Full open defect is an important deep submicron defect and it is expected to increase in future technologies [61, 110, 111]. There are two fault models in literature that model the behaviour of full open defects; one is the capacitance based full open fault model [53, 112, 113, 114, 115] and the other is the gate tunnelling leakage full open fault model [61, 110, 111, 116, 117].

Capacitance based full open model is used to model the voltage of the floating net F (shown in Figure 2.3(a)) as a function of trapped charge on the floating net. According to [3, 115], the floating net voltage V_F is represented as

$$V_F = \frac{C_{High}}{C_{High} + C_{Low}} V_{dd} + \frac{Q_{trap}}{C_{Gnd}}, \tag{2.3}$$

where V_F is voltage on the floating net, C_{High} and C_{Low} is capacitance due to neighbouring lines driving high and low respectively, V_{dd} is the supply voltage, and Q_{trap}/C_{Gnd}

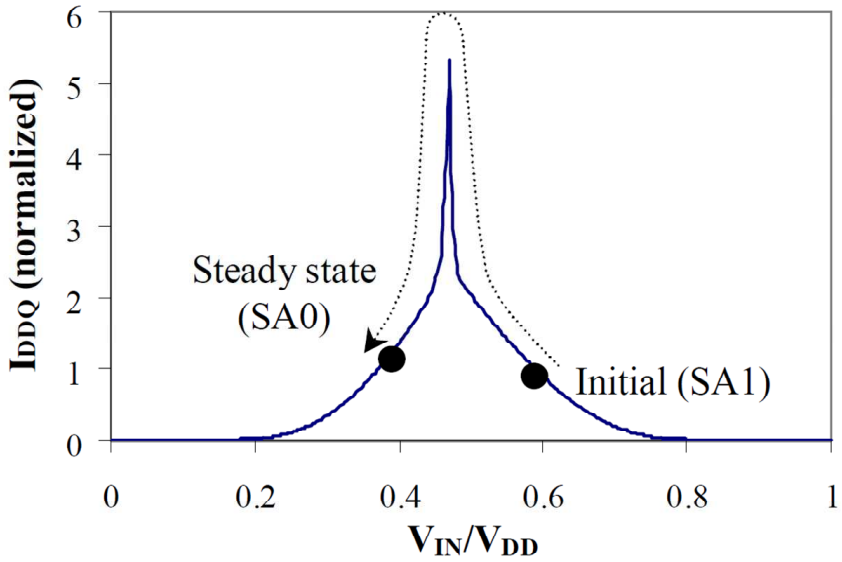


Figure 2.4: Change in logic state due to gate tunnelling leakage [111].

represents the trapped charge on the floating net. From Eq. (2.3), full open defects can be detected through V_F because when the floating net voltage V_F is higher than the logic threshold voltage of the gate input, the full open fault behaves as a stuck-at 1 fault. Similarly a stuck-at 0 fault can also be induced on the floating net when V_F is lower than the logic threshold voltage of the gate input. The fault effect can then be propagated to any of the primary outputs for detection [115]. The results in [3, 118] also show that the detection of full open defects is not affected by the variation of V_{dd} settings when using capacitance based full open model.

Gate tunnelling leakage full open model is used to model the gate tunnelling current through a thin gate oxide that affects the voltage on the floating net (being disconnected from its driver) [51]. In deep submicron designs, small currents can be tunnelled through the thin gate oxide which has been scaled down to ten atom-layers [51]. Figure 2.3 shows the impact of gate tunnelling leakage on the floating net of a full open defect. The floating net F is affected by the tunnelling leakage currents from NMOS and PMOS transistors in gate SG which is illustrated in the second part of Figure 2.3. Results of experiments based on an inverter synthesised using 45-nm technology with a floating input show that the voltage on the floating net increased from 0V to 0.17V due to gate leakage through the PMOS when the inverter output goes to logic high, and the floating voltage is reduced from 0.8 V to 0.58 V due to gate leakage through the NMOS when the inverter output goes to logic low [110]. The study reported in [111] showed that the logic state of a floating net can be transformed (from stuck-at 1 to stuck-at 0) in about 2 seconds due to gate tunnelling leakage currents when using 0.18 μm technology with an open defect, which is shown in Figure 2.4. Another study [111] also predicted that the time to reach steady state will reduce to tens or hundreds of μs for future technology. The studies reported in [3, 118] use static test and leakage-aware fault model for detecting full open defects show that the fault coverage does not vary across V_{dd} settings. It

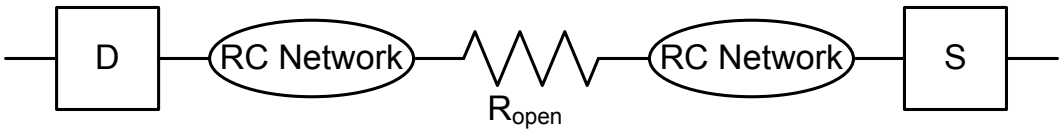


Figure 2.5: Resistive open fault model [51].

means that the detection of full open defects is not affected by the variation of V_{dd} settings. A study [119] used the gate leakage full open model to study the impact of process, voltage and temperature variation on the voltage of the floating net by using basic gates from 90-nm, 65-nm and 32-nm technologies. The results show that process, voltage and temperature variation have a low impact on the voltage of the floating net, which only affects the voltage on the floating net up to 4 mV in variation. The study in this thesis did not focus on full open defects, because process variation has a small impact on defect behaviour [118, 119]. Also, many studies have investigated full open defects [61, 110, 111, 115, 118, 119, 120], therefore this thesis only focuses on resistive open defects.

2.2.2 Resistive Open Fault Models

This section discusses the fault behaviour caused by resistive open defect and the recent fault modelling technique for this class of defects. Fault models for resistive opens have been studied extensively in the last ten years [64, 66, 67, 121, 122, 123]. The study in [64] reported that resistive open defects can cause timing-dependent effects and it shows better detectability when using delay test. Using simple inverter chain with resistive open defect, simulation results in [64] show that delay fault behaviour caused by resistive open is affected by process, voltage and temperature variation. A fault model proposed in [121] models the resistive open defects as a combination of delay faults according to the fanout branches that connect to the open interconnect line. For example, if there are k fanout branches, the total number of single open faults is $2(2^k\text{-}1)$. This fault model lists all the possible faults that can be caused by resistive open defects. For a large design the number of faults can be very large, also this fault model does not take into account the analog behaviour caused by the resistive open. Resistive open defects can be modelled as a resistor between two unconnected nodes with negligible small inductive/capacitive component [122]. A typical resistive open fault model is shown in Figure 2.5. The components “D” and “S” represent the driver and successor gate respectively. It has been reported [123] that the active transition (opposite transition) on the neighbouring nets can change the timing behaviour of the defective net caused by resistive open. Therefore a fault model based on Figure 2.5 with the consideration of neighbouring coupling capacitances is proposed. These fault models [121, 122, 123] can be used to detect resistive open by using delay test. Delay test is used to detect defects that cause timing failure in an IC due to unexpected additional

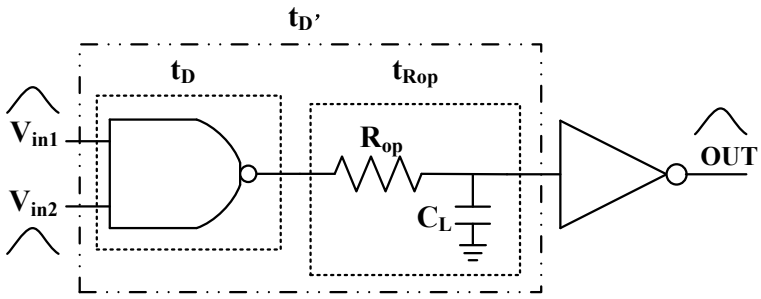


Figure 2.6: Calculation of resistive open delay [66].

delay. In delay test, a defect can only be detected when it causes a longer delay than that of the longest path in a fault-free design [51, 65]. It was shown in [124] that majority of the tested paths show less than one-third delay in comparison to that of the longest path. Therefore a defect in any of these shorter paths can only be detected if it causes a higher delay than that of the longest path in the design. It was reported in [125] that traditional delay test cannot detect resistive open faults that lie out of the longest paths. Therefore test techniques that target small delay transition faults have been developed for resistive open [66, 67]. The small delay transition fault is the delay that introduces less than one clock cycle delay [68, 126]. A model proposed in [66] is used to compute the delay caused by resistive open, which is shown in Figure 2.6. Figure 2.6 shows that a faulty gate delay $t_{D'}$ is obtained from the sum of the gate delay t_D and resistive open delay $t_{R_{op}}$, which is represented in Eq. (2.4). The value of resistive open delay is estimated by Eq. (2.5).

$$t_{D'} = t_D + t_{R_{op}} \tag{2.4}$$

$$t_{R_{op}} = R_{op} * C_L \tag{2.5}$$

A method proposed in [67] is used to calculate the faulty delay caused by the resistive open defect based on fitted value obtained from SPICE simulation. The faulty delay (t_f) of a gate driving an interconnect with a resistive open defect with resistance R_{op} is shown in Figure 2.7 and Eq. 2.6,

$$t_f = t_{nocharge} + \alpha \cdot C_L + \beta \cdot R_{op} \cdot C_L, \tag{2.6}$$

where $t_{nocharge}$ is the delay value which does not consider the load capacitance and can be found from the gate library, α is the constant factor of C_L which is typically included in a gate library, C_L is the lumped load capacitance which is the sum of input capacitances of all successor gates driven by the open interconnect and the parasitic capacitance of the interconnect. As shown in Figure 2.7, C_L is calculated as $C_L = C_{line}$

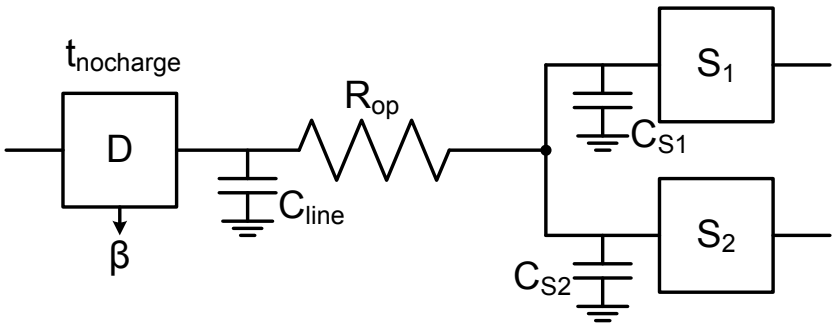


Figure 2.7: Fitted model for resistive open delay calculation [67].

+ C_{S1} + C_{S2}. β is a factor that depends on the electrical parameters of the driving gate. β is constant for a given type of gate and does not depend on R_{op} and C_L. β can be easily determined through SPICE simulation and used as a pre-calculated values for all the gates in the gate library. This method can capture the faulty delay efficiently in nominal operating conditions. When considering process variation, the fitted variable β may become inaccurate because the electrical parameters of the driving gate change due to process variation. The models proposed in [66, 67] do not accurately relate to the transistor parameters, which means it cannot be extended to accurately model the effect of process variation in deep submicron devices. Therefore, a fault model that can accurately capture the effect of process variation is required.

2.2.3 Fault Simulation of Resistive Opens

This section discusses the current fault simulation methods used for resistive opens when operating in nominal conditions. Test methods for detecting resistive open defects include IDDQ test and delay test [45, 64]. The study in [45] reported that IDDQ test becomes ineffective for resistive opens due to high leakage current in nanometre technologies. As reported in [64] resistive opens produce timing-dependent effects and it can be tested by using delay test. Delay fault simulation methods for resistive opens have been reported in many publications [56, 67, 125, 127]. One of these publications [127] analyse simulation results for different test conditions, such as different supply voltage and different operating temperature. Results based on 116 defective chips using 0.18-μm technology node show that delay caused by resistive opens changes in different supply voltage and temperature, for example, delay value increases as the temperature increases. This shows that resistive open defects are both voltage and temperature dependent. In [56], two major sources of resistive open defects are analysed, i.e., incompletely filled vias and partial break in the poly of the transistor (due to salicidation). The experiments are based on multi-V_{dd} settings without considering the effect of process variation. Results show that resistive open defects show voltage dependent detectability and they are better detected on silicon at reduced V_{dd} setting.

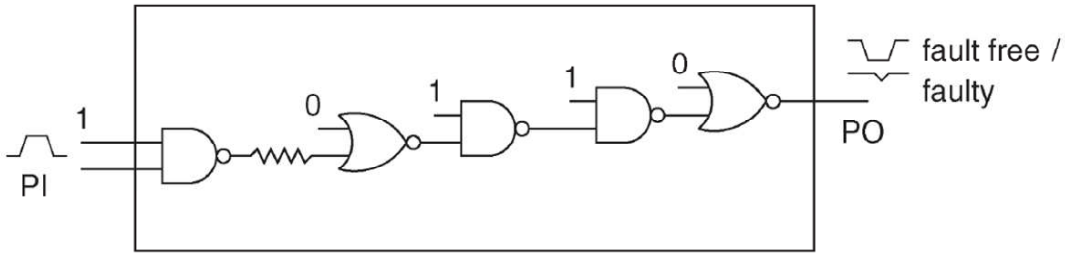


Figure 2.8: Basic idea of the pulse propagation method.

In the same study [56] delay is propagated through the longest path (critical path) in the circuit to be tested, however, defects that lie out of the longest paths and do not cause higher delay than the delay of the longest path are undetected. A study reported in [125] proposed a method through pulse propagation to detect resistive open and resistive bridge in non-critical paths. This method is based on experimental results that a pulse propagate through the affected path by the defects will be dampened, which is shown in Figure 2.8. This method detects the defects by propagating the pulse through a faulty path and comparing output pulse to the fault-free case. Results show that the pulse propagation method can be used to detect resistive opens and resistive bridges. Test techniques that simulate small delay transition faults have been developed for resistive opens. The method reported in [67] calculates the coverage of delay defects with delay size less than one clock cycle. The way of calculating the coverage of small delay defects on resistive open is using Eq. 2.6 as discussed in Section 2.2.2 to map the critical size of the delay with the resistance range, and then using the probabilistic fault coverage metrics (as proposed in [93, 105]) to determine the fault coverage of resistive opens. This method can be used to determine the fault coverage of resistive open defects in fault simulation by using small delay test. Fault modelling and fault simulation of resistive opens when considering process variation is discussed next.

2.3 Fault Modelling when Considering Process Variation

Fabrication process variation has been taken for granted for years and over many scaled technology nodes. Fabrication process variation is mainly due to sub-wavelength lithography, random dopant distribution and line edge roughness, and affects the transistor threshold voltage (V_{th}), oxide thickness (T_{ox}), effective mobility (μ_{eff}), and its geometry (W, L) [10, 11, 12]. As silicon manufacturing processes scale to and beyond the 65nm node, process variation can no longer be ignored [4, 5, 6]. The impact of process variation on integrated circuit performance and power has received/is receiving significant research input. It has been demonstrated that it is possible to control the impact of process variation on circuit performance and power through process tolerant design and improved fabrication techniques [19]. It is also reported that process variation causes the manufactured chips to deviate from the specification to which they were designed [31],

including variation in delay and leakage power [17]. This thesis aims to study the defects behaviour under the influence of process variation and develop fault models for process variation aware test. The following sections discuss the impact of process variation on defect behaviour, which includes logic fault behaviour and delay fault behaviour. Section 2.3.1 discusses the logic fault behaviour of resistive bridges under the influence of process variation, followed by a discussion on current logic fault simulation techniques used for logic test in detecting resistive bridges when considering process variation. Section 2.3.2 discusses the delay fault behaviour of resistive bridges and resistive opens under the influence of process variation, and it also discusses the current delay fault simulation techniques used in delay test when considering process variation.

2.3.1 Logic Faults

To understand the impact of process variation on deep submicron defects, it is necessary to discuss some concepts related to existing fault models and how their behaviour deviates from nominal scenario under the influence of process variation. This section shows the effects of process variation on resistive bridge behaviour for a better understanding of fault modelling and simulation when considering process variation. Due to process variation, the logic behaviour of a resistive bridge deviates from the nominal scenario (Figure 2.2 in Section 2.1.2), leading to loss of fault coverage. This change in resistive bridge behaviour is briefly described next.

The impact of process variation affects two important parameters: drive current of driving gates (D_1 and D_2) and logic threshold voltages of the driven gates (S_1, S_2 and S_3) [2]. The change in these two parameters may introduce additional logic faults resulting in expanding the fault domain of a bridge fault-site. These two parameters are examined individually to clearly illustrate the impact of their change, however in practice (and in all the experiments reported in this thesis) these two parameters vary together and exhibit cumulative effect. First, we illustrate the effect of drive current variation of the driving gates, while keeping the original logic threshold voltages of the driven gates. This is shown in the first part of Figure 2.9, which shows an increase in the voltages on the two nets (V_1 and V_0 for the same value of R_{sh}) and a change in the covered resistance range. It can be seen that in comparison to the nominal scenario shown in Figure 2.2, the critical resistance has changed as $R_{sh} \in [\bar{R}_{I2}, \bar{R}_{I3}]$ now covers the maximum resistance range. From test generation point of view, a test generated to propagate the fault effect through I_2 (gate S_2 as in the case of nominal scenario) will lead to a loss of fault coverage, as $R_{sh} \in [\bar{R}_{I2}, \bar{R}_{I3}]$ will be missed. Therefore in this case, an additional test is needed to propagate the fault effect through I_3 (gate S_3) to cover the new logic fault, added to the fault domain due to process variation.

Next consider the second part of Figure 2.9, which illustrates the effect of logic threshold variation only. It can be seen that the critical resistance has again changed when

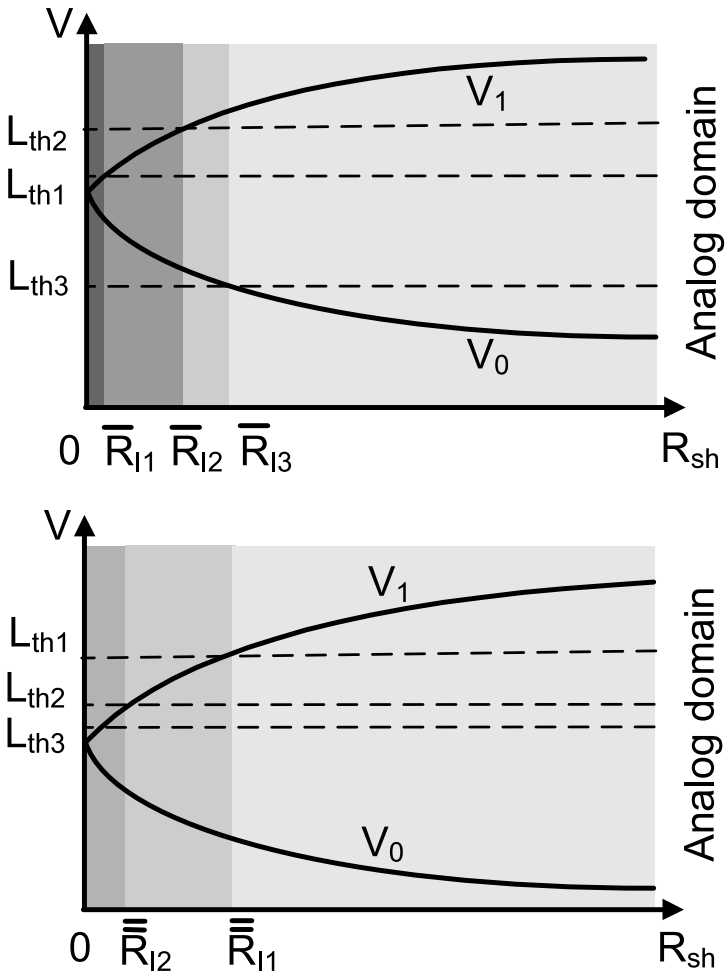


Figure 2.9: Change in drive current and logic threshold voltage (due to process variation) from the behaviour shown in Figure 2.2.

compared with the nominal scenario (Figure 2.2), as $R_{sh} \in [\bar{\bar{R}}_{I2}, \bar{\bar{R}}_{I1}]$ now covers the maximum resistance range. This also leads to loss of fault coverage as test generated to propagate the fault effect through the gate S_2 will only cover resistance range $R_{sh} \in [0, \bar{\bar{R}}_{I2}]$. Therefore an additional test is needed to propagate the fault effect through I_1 (gate S_1) to cover the new logic fault added to the fault domain.

As discussed in Section 2.1.3.1, resistive bridge can be tested by using logic test through logic fault simulation. Process variation has been considered in testing logic faults caused by bridge defects in CMOS digital ICs in recent years [2, 128, 129]. A study in [128] used a logic fault based model to generate test for resistive bridges under the influence of process variation. This logic fault model is developed by including all possible logic behaviour which can be caused by process variation. Results from [128] show that test for resistive bridges can provide fault coverage up to 80.5%. In [129], the fault model from the study discussed previously [128] was improved to reduce the number of considered logic faults without causing a test escape, and a test generator was presented by using Monte-Carlo simulation to model process variation. Results in [129] show that there is an upper limit

to the number of test patterns required to achieve full defect coverage for a given bridge. This means that process variation-aware test generation for bridges is feasible. However, the methodologies presented in [128, 129] were independent of electrical IC parameters. That means it cannot model the variation in IC parameters properly. It is also reported that N-detection test based on single stuck-at fault model can be used to detect bridge defects through the increase of N [33, 130, 131]. N is the number of different test patterns applied to each single fault under test to increase the probability of detecting the faulty behaviour of un-modelled defect. The first attempt at modelling the impact of process variation on resistive bridge faults by considering the variation of electrical IC parameters through SPICE based on Monte-Carlo simulation is reported in [2]. For each bridge fault-site, it uses SPICE simulation to determine the voltages (V_1 and V_0, Figure 2.1) at discrete bridge resistance intervals and stores the outcome in a database for subsequent use. The nominal values of V_1 and V_0 are then used to generate a new set of variation-induced logic faults by Monte-Carlo simulation and for this purpose four transistor parameters (threshold voltage (V_{th}), width (W), length (L) and oxide thickness (T_{ox})) are varied by Gaussian distribution with a standard deviation of approximately 10% of mean value through 500 permutations to generate a new set of variation-induced logic faults (Figure 2.9). Results based on ISCAS 85, 89 benchmarks and a 45-nm gate library show that tests generated for the nominal scenario without considering process variation can lead to as much as 10% loss of fault coverage [2]. Fault modelling and simulation through SPICE is accurate but when considering process variation, Monte-Carlo simulation in SPICE requires a long computation time to generate fault behaviour. It is reported in [2, 3] that to generate a database for ISCAS 85 89 benchmarks it took nearly a week with 8 computers working in parallel.

2.3.2 Delay Faults

Recently, testing resistive bridges and resistive opens using delay fault modelling and simulation of these two dominant DSM defects with the consideration of process variation have attracted the attention of the test community. Process variation affects transistor parameters leading to change of logic threshold voltages (L_{th}) of the successor gates and drive strength(s) (I_{ds}) of the driving gate(s), as observed in [2, 42]. This change in L_{th} and I_{ds} introduces additional delay faults, which may lead to loss of fault coverage if tests are generated without considering process variation. Delay test for these two defects under the influence of process variation have been studied recently [1, 29, 30, 63, 64, 69, 79, 92]. The study reported in [64] shows that resistive open and resistive bridge defects are affected by process variation and results show that delay fault testing is an effective way for testing such defects. In [63] two test techniques are used to detect resistive open defects in a ripple-carry adder in 16-nm technology under the influence of process variation. The first test technique is the IDDT test method that detects unexpected transient power supply and ground currents while the second method uses

propagation delay test by measuring propagation delay from the primary inputs to the circuit outputs. IDDT test is used to observe instantaneous or mean values of transient power supply current, which is proposed to replace or complement IDDQ test [132]. Results show that delay test can better detect resistive open defects even in the case of extreme process variation.

The variation of transistor parameters in CMOS circuits due to process variation tends to affect current and logic threshold voltage of a gate leading to the change of gate delay and path delay from their nominal values caused by resistive bridges and resistive opens. The problem of determining the longest path in terms of delay under process variation has been addressed in [1, 29]. A longest path through a net under process variation is defined as a path with a configuration of IC parameter values generated due to process variation that has the maximum delay among all paths through the net [29]. So for each net, there can be multiple paths, where each is the longest path under different configurations of the IC parameters. The study in [29] provides a method for calculating delay as a function of IC parameters and uses delay results to select the longest path to target with delay fault testing under process variation. An algorithm is proposed to generate the set of longest paths by pruning paths that are not longest to reduce test running time while keeping high fault coverage. In [1] a process variation-aware delay fault test method to select longest paths is proposed. The paths are selected by considering the probability of each node in the circuit. The delay test method proposed in [1, 29] can only detect delay caused by defects with longer than the delay of the longest path. However, defects that lie out of the longest paths and cannot cause higher delay than the delay of the longest path remain undetected. The small delay defects test is a way of detecting such defects in shorter paths. An test method based on small delay defect testing has been proposed in [69] to screen chips that have resistive open defects under the presence of process variation. The method first identifies scan outputs that can fail only if there are defects under the possible worst case process variation and then assumes that a chip failed due to a defect if the number of faulty scan outputs in a test pattern is much larger than that of faulty scan outputs of a typical defect-free chip. The defective chips are screened by comparing the number of fail patterns. Experimental results using 10 benchmark circuits show that this small delay testing method was able to successfully screen more than 90% of defective chips for 8 circuits. The change of gate delay and path delay in a circuit can come from the impact of process variation but can also come from lower voltage and higher temperature caused by high switching activity during delay test. Delay change due to switching activity is called false delay and it can cause false delay test failures. To avoid false delay test failures under process variation, a variation-tolerant delay fault test generation technique is presented in [30]. This technique avoids false delay test failures by minimising the switching activity during the transitions and optimizing the test pattern set. The study in [79] proposed a method for statistical library characterisation when considering the influence of process variation. It uses Monte Carlo SPICE simulation at electrical level

to extract delay distributions of gates in the presence of resistive bridges and resistive opens and for the defect-free case. The characterised process variation induced delay behaviour of each gate is represented as histograms and stored in a histogram database in order to provide interface for test generation algorithms at higher levels of abstraction. The database needs to be generated every time for each different technology or different gate library, also the computation time of the database is very long. It has been reported in [79] that the complete characterisation for three gates (NAND2, NOR2 and inverter) takes about 10 days on a HPC-Cluster with 32 nodes.

2.4 Concluding Remarks

This chapter has presented an overview of recent works on fault modelling of bridge and open defects and their fault simulation techniques. The fault models for bridge defects are categorised into two types, non-resistive bridge fault models and the resistive bridge fault models. This research only focuses on resistive bridge fault model because it better captures the analog effect caused by the bridge. The open defects can be categorised into two type, full opens (Section 2.2.1) and resistive opens (Section 2.2.2). This research targets only resistive opens. Full opens are not considered in this thesis because recent research shows that process and voltage variation have small impact on full open defect behaviour [119, 118], and many studies have investigated full open defect behaviour [61, 110, 111, 115, 119, 118]. This literature review shows that the impact of process variation on deep submicron manufacturing designs cannot be ignored due to continuous scaling of CMOS. Research shows that logic and delay behaviours of resistive open and resistive bridge are affected by process variation [2, 29, 30, 79, 92] and may lead to loss of fault coverage. Fault modelling and simulation through SPICE is the most accurate method of modelling and simulating fault behaviour. However, SPICE requires a long computation time to model and simulate fault behaviour when considering process variation [2, 3, 79, 80]. Therefore, new efficient fault modelling techniques and simulation methodologies targeting resistive bridges and resistive opens under process variation are needed for manufacturing test to increase fault coverage and reduce test cost. The research aim of this project is to develop efficient process variation-aware fault modelling techniques and simulation methodologies targeting resistive bridges and resistive opens as outlined in Section 1.5, Chapter 1.

Chapter 3

Variation-Aware Logic Fault Modelling of Resistive Bridge Defects

Chapter 2 highlights the need of fast and accurate variation-aware fault modelling technique for resistive bridge defects that can be used for new manufacturing test methods. This chapter presents such a logic fault modelling technique to model the effect of process variation which includes die-to-die variation and within-in die variation on resistive bridge defects. The technique includes two efficient voltage calculation algorithms to calculate the logic threshold voltage of driven gates and voltages on bridged nets of a fault-site for calculating the bridge critical resistance without using SPICE simulation. The technique also takes into account the effect of voltage and temperature variation on resistive bridge defects. Results based on a 65-nm gate library show that on average the proposed modelling technique is more than 53 times faster and in the worst case, the error in bridge critical resistance is 2.64% when compared with HSPICE.

3.1 Introduction

Resistive bridges represents a major class of defects in deep submicron (DSM) CMOS and has received increased attention on modelling and simulation [94]. Manufacturing test employs fault models for testing digital circuits to emulate the physical behaviour of a defect at device level. Accurate fault models are important for fault simulation and test generation. The resistance of a bridge (R_{sh}, Figure 3.1[1]) is a continuous parameter which is not known in advance. Resistive bridge changes the voltage on the bridged nets (V_1 and V_0, Figure 3.1) from 0 V or V_{dd} to some intermediate value, which varies

[1]Figure 2.1 in Chapter 2 is repeated in Figure 3.1 for convenience.

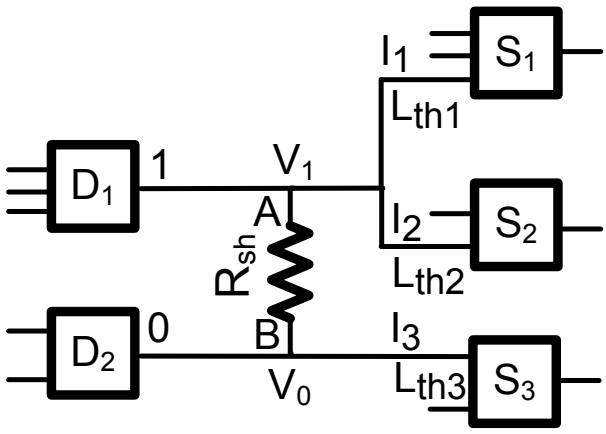


Figure 3.1: Resistive bridge forming potential circuit fault.

with R_{sh} of the bridge fault. A number of methods have been proposed in literature to determine the behaviour of the bridge fault-site in the presence of this unknown (R_{sh}) parameter. The first fault model to take into account the intrinsic resistance of a bridge is proposed in [47], which is based on Shockley transistor model. It uses curve fitting to match results with SPICE data to achieve high accuracy. To account for DSM behaviour, a more advanced transistor model (BSIM4) is used to compute bridge critical resistance [94]. These two resistive bridge fault models [47, 94] are intended for designs operating in nominal conditions, however, due to continuous scaling of CMOS, DSM designs are affected by process variation [4, 6]. Fabrication process variation is mainly due to sub-wavelength lithography, random dopant distribution, line edge roughness and stress engineering [10, 12]. In a recent study, it has been shown that more than 30% error in the drive current of a transistor is observed on a 65-nm device due to process variation, when compared to a transistor operating in nominal conditions [10]. Process variation also affects the behaviour of a resistive bridge defect [2]. Using ISCAS 85, 89 benchmarks and a 45-nm gate library, it was shown that tests generated for nominal scenario without considering process variation can lead to as much as 10% loss of fault coverage [2]. Two important parameters are affected by process variation leading to additional logic faults and loss of fault coverage due to these additional logic faults. These two parameters are the drive current of the driving gates (D_1 and D_2, Figure 3.1) and the logic threshold voltage of the driven gates (S_1, S_2 and S_3, Figure 3.1), which is discussed in Section 2.3.1. Bridge defect critical resistance calculation through Shockley transistor model with curve fitting to match SPICE data is accurate only in nominal operating conditions and it loses accuracy under the influence of process variation [133].

The first attempt of modelling the impact of process variation on bridge faults has been reported in [2] using SPICE based on Monte-Carlo simulation, which is discussed in Section 2.3.1, Chapter 2. It uses SPICE simulation to determine the voltages (V_1 and V_0, Figure 3.1) at discrete bridge resistance intervals and stores the outcome in a database for subsequent use. This method has two limitations: Firstly, when scaling

from one technology node to another, the database (with SPICE information) needs to be regenerated, since it is technology-specific; Secondly, the database generation (per technology node) requires a long computation time. A recent study has reported that it took nearly a week with 8 computers working in parallel to generate a database for ISCAS 85, 89 benchmarks [3]. See Section 3.2 for more details on limitations of available fault modelling techniques. In this chapter, these two limitations are addressed by developing a fast and accurate model of resistive bridge defects, while incorporating the effect of process, voltage and temperature variation. The proposed modelling technique is accurate because it is based on the most recent transistor model (BSIM4.7: Berkeley Short-Channel IGFET Model) [86]. The proposed modelling technique is fast because it employs highly efficient voltage calculation algorithms to compute the logic threshold voltages (L_{th1}, L_{th2} and L_{th3}, Figure 3.1) and the bridge critical resistance. The effect of process variation is considered for both die-to-die variation (Section 1.1.1, Chapter 1) and within-die variation (Section 1.1.2, Chapter 1). The die-to-die variation is modelled using uncorrelated parameter fluctuations by considering three transistor parameters: gate length (L), threshold voltage (V_{th}) and effective mobility (μ_{eff})[2] as reported in a recent study [10]. The within-die variation is modelled using spatially correlated parameter fluctuations by considering the correlation coefficient on transistor gate length (L), which is identified as the major contributor of such variations [22, 23, 24, 25]. The effect of voltage variation is directly applied by changing the supply voltage; finally, the effect of temperature variation is incorporated by using temperature dependent transistor models of threshold voltage, mobility and saturation velocity using BSIM4 [86]. Simulation results verify that the proposed modelling technique is accurate (worst-case deviation of 2.64%) and leads to significant speedup (on average 53 times) in critical resistance calculation when compared with HSPICE. This is the first reported modelling technique for resistive bridge defects that incorporates the influence of process, voltage and temperature variation without using HSPICE.

The chapter is organised as follows: State-of-the-art fault modelling techniques and their limitations are discussed in Section 3.2. The proposed variation-aware bridge defect modelling technique is discussed in Section 3.3. Simulation setup and results are considered in Section 3.4, and Section 3.5 concludes the chapter.

3.2 State-of-the-Art Fault Models

This section examines the available device-level fault models for resistive bridge defects and their limitations in modelling the effect of process variation. In general, the available fault models (in nominal operating conditions) can be categorised into two: SPICE-based [2] and Fitted Models [94]. These two models offer a trade-off between speed and

[2]Mobility varies due to variation in effective strain in a strained silicon process [10], which is discussed in Section 1.1.

accuracy. SPICE-based Models offer high accuracy at the expense of long simulation time and the Fitted Models offer very fast computation time but are less accurate than SPICE-based Models.

The investigation that integrates the effect of process variation in a resistive bridge model is reported in [2, 3]. It uses a SPICE-based Model, and to integrate the effect of process variation, it uses the following four transistor parameters: threshold voltage (V_{th}), width (W), length (L) and oxide thickness (T_{ox}). These parameters are assumed to be statistically independent and are varied by Gaussian distribution with a standard deviation of approximately 10% of mean value. The experiments are conducted on a 45-nm gate library with Predictive Technology Model (PTM) transistor models [21]. For each bridge fault-site, it uses SPICE simulation to determine the voltages (V_1 and V_0, Figure 3.1) at discrete bridge resistance intervals and stores the outcome in a database for subsequent use. The nominal values of V_1 and V_0 are then used to generate new set of variation-induced logic faults by Monte-Carlo simulation and for this purpose the four parameters are varied through 500 permutations to generate a new set of variation-induced logic faults (Figure 2.9, Chapter 2). This method has two limitations: Firstly, when scaling from one technology node to another, the database (with SPICE information) needs to be re-generated, as that is technology-specific; Secondly, the database generation (per technology node) requires a long computation time. It is reported that it took nearly a week with 8 computers working in parallel to generate a database for ISCAS 85, 89 benchmarks [3].

The database generation can be avoided by calculating the critical resistance of a bridge by using I-V_{ds} based electrical equation of a Shockley transistor model [47]. Since Shockley model is a simple transistor model [134, 135], curve fitting is used to match the results with SPICE data, leading to what is called a "Fitted Model", which uses additional coefficients to achieve higher accuracy than Shockley model [94, 136]. The Fitted Model is intended for nominal operating conditions and only 0.4% worst-case error is reported when compared with SPICE results on a 0.35μm gate library [94]. When considering the effect of process variation, the problem with the Fitted Model is that of accuracy, i.e., the percentage of error increases as process variation is introduced. To study the effect of process variation on the Fitted Model, the I-V_{ds} equation of the Shockley model and a 65-nm PTM transistor model card [21] are used. The I-V_{ds} Shockley model is fitted using HSPICE simulation results in nominal operating conditions, and the I-V characteristics of an NMOS transistor are shown in Figure 3.2. These plots are generated by increasing V_{gs} from 0 V to 1.2 V with a step size of 0.3 V. It can be seen that the Fitted Model matches well with that of HSPICE in nominal operating conditions. Next, the effect of process variation is introduced by varying the transistor gate length by 5-nm (for illustration purposes) and re-generating the plots using HSPICE and the Fitted Model. The result is shown in Figure 3.3. As can be seen the Fitted Model deviates from the HSPICE simulated results. This is because, the Fitted Model uses a

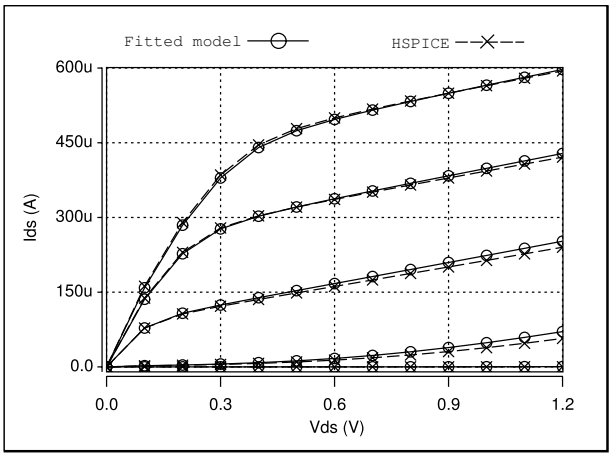


Figure 3.2: Nominal operating conditions: HSPICE and Fitted Model.

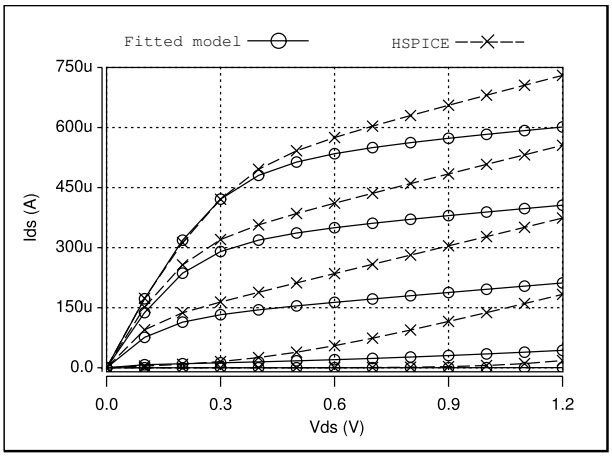


Figure 3.3: Effect of gate length variation: HSPICE and Fitted Model.

simple Shockley transistor model, which does not take the effect of process variation into account, leading to inaccurate results. To improve its accuracy, additional models that relate the inter-dependencies between different transistor parameters are needed [137]. For example, scaling of the gate length results in reducing V_{th}, while increasing sub-threshold swing and Drain Induced Barrier Lowering (DIBL). Therefore to accurately model the impact of process variation, more accurate transistor models should be used to relate different electrical parameters with the device structure. This means that curve fitting at nominal operating conditions using a simple (Shockley) transistor model and SPICE simulation data cannot be extended to accurately model the effect of process variation in deep submicron devices.

3.3 Variation-Aware Bridge Logic Fault Modelling Technique

The modelling technique employs two efficient algorithms to generate the logic fault (Figure 2.2, Chapter 2) of a bridge fault-site without using HSPICE. The first algorithm involves calculating the logic threshold voltage (L_{th}) of driven gates (S_1, S_2 and S_3, Figure 3.1) of a bridge fault-site. Logic threshold voltage is defined as the gate input voltage at which the gate output voltage is equal to $\frac{V_{dd}}{2}$, while all other inputs of the gate are at non-controlling value(s) [104]. This calculation is necessary, since L_{th} is needed for critical resistance calculation of a given fault-site and is calculated using BSIM4 transistor model. The second algorithm of the proposed technique computes the critical resistance of a bridge fault-site through the bridged net voltage (V_0 and V_1, Figure 3.1) approximation algorithm using BSIM4 transistor model. The bridged net voltage approximation algorithm can use linear search method or binary search method. Employing BSIM4 transistor model through HSPICE incurs high simulation time, which is reduced by using efficient calculation algorithms (Section 3.3.1 and Section 3.3.2) for logic threshold calculation and voltages on the bridged nets. The proposed technique is faster than HSPICE because of the following two reasons. Firstly, critical resistance calculation through HSPICE requires sweeping the resistance range from 0Ω to (typically) 20,000Ω [50] to observe the point where faulty value changes to fault-free value (R_{crit}). This requires two hundred HSPICE DC simulations, assuming a DC-sweep step size of 100Ω [3, 50]. The proposed technique uses the logic threshold voltages of the driven gates and calculates R_{crit} at these specific voltage points, thereby reducing the number of iterations for calculating R_{crit}. Secondly, in a DC sweep, HSPICE initialises and calculates about 400 more variables than actually needed for calculating R_{crit}, the proposed technique achieves the speed up by calculating only the necessary variables for calculating R_{crit} thereby achieving speedup without compromising accuracy. The proposed technique is as accurate as HSPICE because it also uses BSIM4 transistor model as used in HSPICE for critical resistance calculation. Through these algorithms

discussed in Section 3.3.1 and 3.3.2, high accuracy is achieved at low computation cost as demonstrated in Section 3.4. These two algorithms therefore determine the values of all parameters needed to determine the logic fault behaviour of a bridge fault-site at nominal operating conditions. It is recently demonstrated that bridge fault is negatively affected by process variation [2]. Therefore the effect of process variation, includes die-to-die variation and within-die variation, on the logic behaviour of a bridge fault-site is incorporated into the proposed technique. The proposed technique also takes into account the effect of voltage and temperature variation on bridge fault behaviour.

The logic threshold voltage of a fanout gate and the voltages on bridged nets of a bridge fault-site are calculated by using the BSIM4 transistor model, which accurately relates different electrical parameters with transistor device structure and takes into account various inter-dependencies between different transistor parameters. For example, scaling of the gate length results in reducing V_{th}, while increasing subthreshold swing and drain induced barrier lowering (DIBL). It is therefore well-suited to model the effect of process, voltage and temperature variations [86]. The following equations models the transistor drain current of a CMOS transistor:

$$\begin{aligned} I_{ds} = & \frac{I_{ds0} \cdot NF}{1 + \frac{R_{ds} \cdot I_{ds0}}{V_{dseff}}} \left[1 + \frac{1}{C_{clm}} ln \left(\frac{V_A}{V_{Asat}} \right) \right] \\ & \cdot \left(1 + \frac{V_{ds} - V_{dseff}}{V_{ADIBL}} \right) \cdot \left(1 + \frac{V_{ds} - V_{dseff}}{V_{ADITS}} \right) \\ & \cdot \left(1 + \frac{V_{ds} - V_{dseff}}{V_{ASCBE}} \right) \end{aligned} \tag{3.1}$$

where I_{ds} is the drain current equation for both linear and saturation regions, I_{ds0} (Eq. (3.2)) is the drain current valid from the subthreshold to the strong inversion regime, NF is the number of device fingers, R_{ds} is the source/drain resistance, V_{ds} is the source/drain voltage, V_{dseff} is the effective V_{ds}, C_{clm} is the channel length modulation, V_A is the Early voltage, V_{Asat} is the Early voltage at $V_{ds} = V_{dsat}$, V_{ADIBL} is the Early voltage due to Drain Induced Barrier Lowering (DIBL), V_{ADITS} is the Early voltage due to Drain Induced Threshold Shift (DITS), V_{ASCBE} is the Early voltage due to substrate current induced body effect (SCBE).

$$I_{ds0} = \frac{W \mu_{eff} Q_{ch0} V_{ds} \left(1 - \frac{V_{ds}}{2V_b} \right)}{L \left(1 + \frac{V_{ds}}{E_{sat} L} \right)} \tag{3.2}$$

where μ_{eff} is the effective mobility of the carriers, Q_{ch0} is the channel charge density, V_b is given by $\frac{(V_{gsteff} + 2v_t)}{A_{bulk}}$, V_{gsteff} is the effective $(V_{gs} - V_{th})$, v_t is the thermal voltage, A_{bulk} models the bulk charge effect, E_{sat} is the critical electric field at saturated carrier

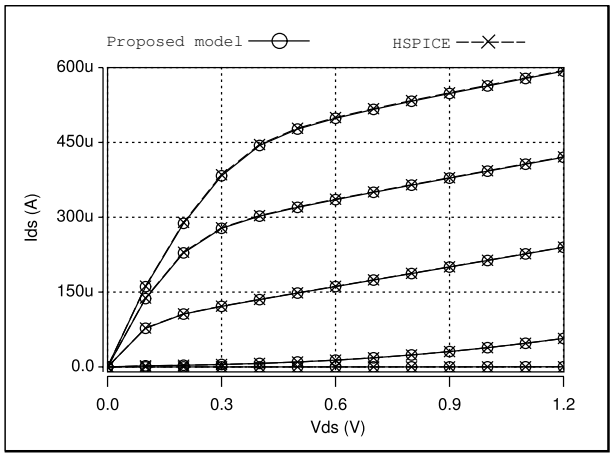


Figure 3.4: I-V$_{ds}$ characteristics using a BSIM4 transistor model and HSPICE.

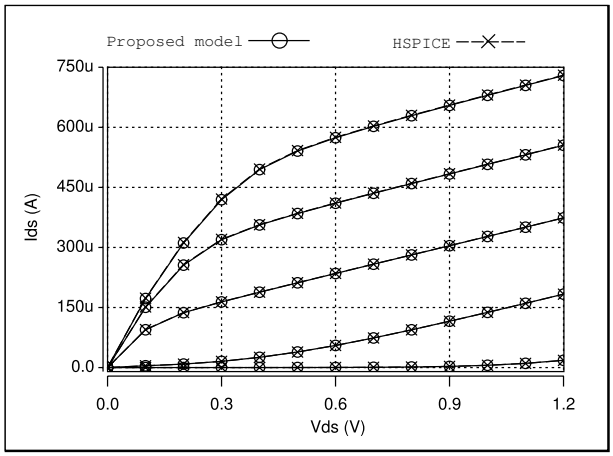


Figure 3.5: Effect of gate length variation on BSIM4 transistor model in comparison to HSPICE.

velocity. The above equations (Eq. (3.1) and Eq. (3.2)) are solved by using the device parameters (per transistor) through a transistor model card (for example, PTM [21]) and the variables (for example, C_{clm}, V_{ADIBL}, V_{ADITS} etc.) of Eq. (3.1) and Eq. (3.2) are obtained from the BSIM4 transistor model equations [86]. Leakage current including gate tunnelling current, sub threshold channel current (calculated as part of I_{ds0}) and Gate-Induced-Drain-Leakage (GIDL) current is also included by using their respective current models from BSIM4. Note that body effect is incorporated in BSIM4 transistor threshold voltage model, which is used in I_{ds} equation. It has been validated by comparing the results with HSPICE using 65-nm PTM model card as discussed in details in Section 3.4. Using Eq. (3.1) and SPICE simulations, Figure 3.4 shows the I-V characteristics of an NMOS transistor in nominal operating conditions, and Figure 3.5 shows the effect of varying the gate length by 5-nm. Both graphs show that results generated by using Eq. (3.1) have an excellent match with HSPICE, which demonstrates accurate modelling of the gate variations. Figure 3.4 and Figure 3.5 are generated by using a C/C++ program based on Eq. (3.1), and the device parameters of Eq. (3.1) are provided by a PTM transistor model card [21].

3.3.1 Logic Threshold Voltage Calculation Algorithm

Logic threshold voltage (L_{th}) of a gate can be calculated using HSPICE, however that is a time consuming process and negatively affects the computation time of critical resistance calculation. Using 350 fault-sites, each with up to 5 driven gates per bridged net, it is shown in simulation results (Section 3.4.3.1) that the improvement in critical resistance calculation time is reduced to only 10% when using HSPICE for L_{th} generation in comparison to 7 times improvement with a pre-computed L_{th} database. This motivates the need for an efficient logic threshold generation algorithm.

Figure 3.7 shows the algorithm for calculating logic threshold voltage of a gate, which is applicable to both simple and compound gates (such as OR4, AND4 and AO22). Logic gates (simple or compound) can be divided into a number of internal stages, where each stage can be sub-divided into pull-up and pull-down networks. This is illustrated in Figure 3.6 that shows a compound gate with "k" stages ($1 \leq i \leq k$) and each stage has a pull-up and pull-down network. When calculating the logic threshold voltage of such a gate, the stage connected to the gate output ($V_{out,1}$) is first started and its individual logic threshold voltage is calculated. For the first stage, the voltages across the pull-up and pull-down networks is set to $\frac{V_{dd}}{2}$ [104]. The logic threshold of this stage acts as the output voltage of the previous stage (Figure 3.6) $V_{out,2}$=$V_{in,1}$. This voltage ($V_{out,2}$) is used to calculate the logic threshold voltage of the second stage and this calculation continues until the logic threshold voltage of the last stage ($V_{in,k}$) of the gate is determined.

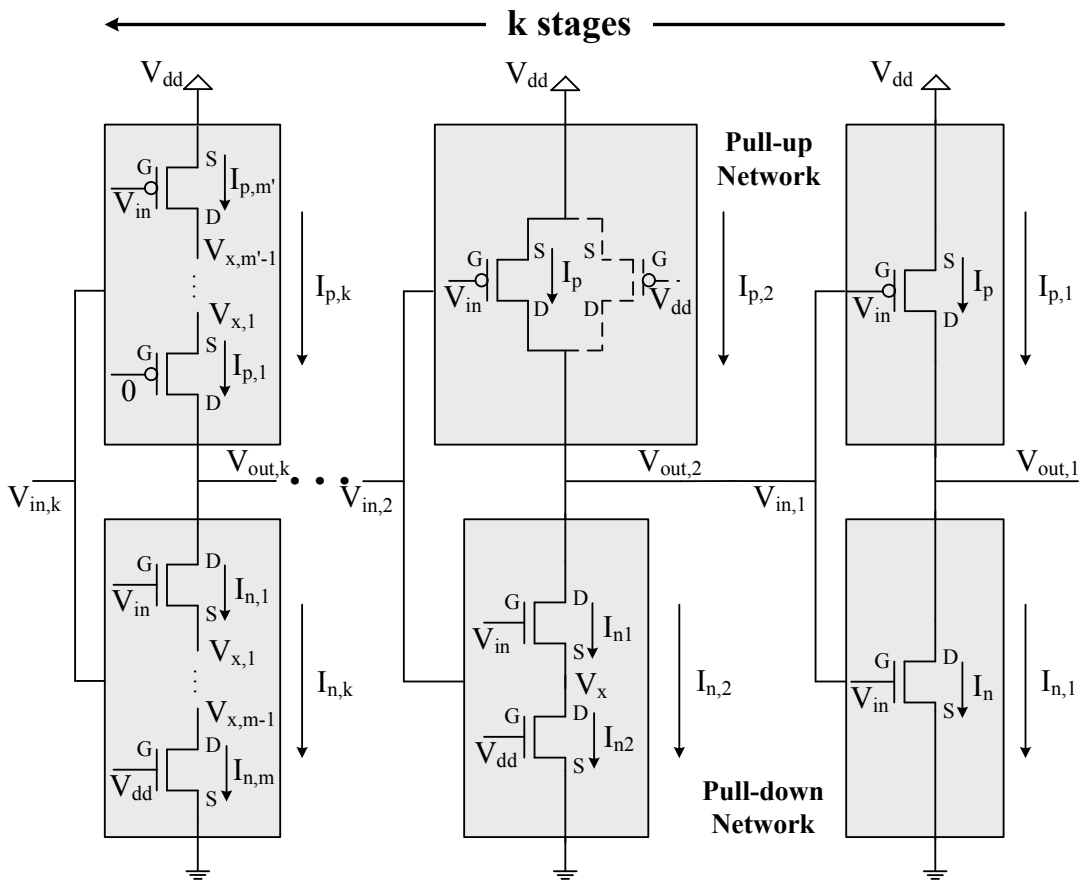


Figure 3.6: General framework for logic threshold voltage calculation

Logic threshold of each stage is calculated by approximating the input voltage ($V_{in,i}$) at which the currents through the pull-up (I_p) and pull-down (I_n) networks are equal, where output voltage ($V_{out,i}$) is known. The algorithm (Figure 3.7) first converts a (pull-up or pull-down) network into a number of series connected transistors by calculating the effective $\frac{W}{L}$ of parallel connected transistors ($\frac{W}{L} = \frac{W_1}{L_1} + \cdots + \frac{W_n}{L_n}$). The algorithm (Figure 3.7) approximates the logic threshold voltage between the two variables, V_{Max} and V_{Min}. It first assigns V_{dd} to V_{Max} and 0 V to V_{Min}, with each iteration it reduces the separation between V_{Max} and V_{Min} by half. The input voltage V_{in} is set to the midpoint between V_{Max} and V_{Min} and by comparing the currents through the pull-up and pull-down networks, where the currents are calculated through the algorithm shown in Figure 3.8. Each iteration reduces the separation between V_{Max} and V_{Min} either by reducing V_{Max} or increasing V_{Min} (step-17 to step-21 in Figure 3.7). These steps are based on the principle that for $V_{in} \geq L_{th}$, $I_n \geq I_p$ and at $V_{in} < L_{th}$, $I_n < I_p$. The algorithm terminates when the difference in I_n and I_p is smaller than $LIMIT$. In this work, $LIMIT$ is set to 1μA, which was decided through a trade-off analysis between accuracy and speed as shown in Table 3.1. Table 3.1 shows the results of calculating logic threshold voltage by using algorithm shown in Figure 3.7 and Figure 3.8 based on a 65-nm gate library. It shows the change of $LIMIT$ (first column) leading to the loss of accuracy (second column) and the improvement in simulation runtime (third column)

Input: V_{dd}
Output: V_{in}
1: Read the PTM model card.
// Model card is needed for parameters in Eq. (3.1)
2: $LIMIT = 1\ \mu$A
3: Divide gate structure into k (k ≥ 1) internal stages.
*// E*ach stage has a pull-up and pull-down network.
4: Each network is converted in to m ($m \leq 3$) transistors in series.
5: **for** $i = 1 \rightarrow k$ **do**
6: **if** $i = 1$ **then**
7: $V_{out,i} = V_{out} = V_{dd}/2$
8: **else**
9: $V_{out,i} = V_{in,(i-1)}$
10: **end if**
11: $V_{Max} = V_{dd}$, $V_{Min} = 0$
12: $V_{tmp} = (V_{Max} + V_{Min})/2$
// This loop is used to get $V_{in,i}$
13: **repeat**
14: $V_{in,i} = V_{tmp}$
15: $I_{n,i}$ can be calculated by using the algorithm shown in Figure 3.8 with input of $V_{in,i}$ and $V_{out,i}$
16: $I_{p,i}$ can be calculated by modifying the algorithm shown in Figure 3.8 with input of $V_{in,i}$ and $V_{out,i}$
17: **if** $I_{n,i} \geq I_{p,i}$ **then**
18: $V_{Max} = V_{tmp}$
19: **else**
20: $V_{Min} = V_{tmp}$
21: **end if**
22: $V_{tmp} = (V_{Max} + V_{Min})/2$
23: **until** $(|I_{n,i} - I_{p,i}| \geq LIMIT)$
24: **end for**
25: **return** $V_{in} = V_{in,k}$

Figure 3.7: Logic threshold voltage algorithm

Input: V_{in} and V_{out}
Output: I_n

Read the PTM model card.
$LIMIT = 1\ \mu$A
if Single NMOS **then**
  $I_n = I_{ds,n}(V_{ds} = V_{out},\ V_{gs} = V_{in})$
  // $I_{ds,n}$ *is calculated by using Eq.* (3.1)
else if Two NMOS transistors in series **then**
  $V_{Max} = V_{out},\ V_{Min} = 0$
  $V_x = (V_{Max} + V_{Min})/2$
  repeat
    $I_{n,1} = I_{ds,n}(V_{ds} = V_{out} - V_x,\ V_{gs} = V_{in} - V_x)$
    $I_{n,2} = I_{ds,n}(V_{ds} = V_x,\ V_{gs} = V_{dd})$
    if $I_{n,1} \geq I_{n,2}$ **then**
      $V_{Max} = V_x$
    else
      $V_{Min} = V_x$
    end if
    $V_x = (V_{Max} + V_{Min})/2$
  until $(|I_{n,1} - I_{n,2}| \geq LIMIT)$
  $I_n = (I_{n,1} + I_{n,2})/2$
else if Three NMOS transistors in series **then**
  $V_{Max,1} = V_{out},\ V_{Min,1} = 0$
  $V_{x,1} = (V_{Max,1} + V_{Min,1})/2$
  repeat
    $V_{Max,2} = V_{x,1},\ V_{Min,2} = 0$
    $V_{x,2} = (V_{Max,2} + V_{Min,2})/2$
    repeat
      $I_{n,2} = I_{ds,n}(V_{ds} = V_{x,1} - V_{x,2},\ V_{gs} = V_{dd} - V_{x,2})$
      $I_{n,3} = I_{ds,n}(V_{ds} = V_{x,2},\ V_{gs} = V_{dd})$
      if $I_{n,2} \geq I_{n,3}$ **then**
        $V_{Max,1} = V_{x,1}$
      else
        $V_{Min,1} = V_{x,1}$
      end if
      $V_{x,1} = (V_{Max,1} + V_{Min,1})/2$
    until $(|I_{n,2} - I_{n,3}| \geq LIMIT)$
    $I_{tmp} = (I_{n,2} + I_{n,3})/2$
    $I_{n,1} = I_{ds,n}(V_{ds} = V_{out} - V_{x,1},\ V_{gs} = V_{in} - V_{x,1})$
    if $I_{n,1} \geq I_{tmp}$ **then**
      $V_{Max,2} = V_{x,2}$
    else
      $V_{Min,2} = V_{x,2}$
    end if
    $V_{x,2} = (V_{Max,2} + V_{Min,2})/2$
  until $(|I_{n,1} - I_{tmp}| \geq LIMIT)$
  $I_n = (I_{n,1} + I_{n,2} + I_{n,3})/3$
end if
return I_n

Figure 3.8: L_{th} calculation algorithm for NMOS transistors in series.

Table 3.1: Impact of $LIMIT$ on accuracy and speedup

$LIMIT$	Average Error of L_{th}	Average Speedup
0.1μA	1.5%	97x
1μA	2.4%	257x
10μA	6.1%	403x

in logic threshold voltage calculation when comparing the results with HSPICE. Results show that logic threshold calculated with the $LIMIT$ of 1μA, has accuracy error of 2.4%, with speedup of 257 times, when compared with SPICE, which is considered to be a balance point between speed and accuracy. The algorithm converges using small (on average 15 or less) number of iterations with $LIMIT$ setting to 1μA for all reported results in Section 3.4. The current through the series connected transistors in pull-up and pull-down networks is calculated by using the algorithm (Figure 3.8), which can be used for up to three transistors in series ($m \leq 3$; where m is the number of transistors in series).

Next, how to approximate the current through the pull-down network (I_n) is explained, for the second stage of a gate shown in Figure 3.6, with two NMOS transistors in series ($m = 2$). The steps (Step-5 to Step-18) are shown in Figure 3.8. I_n is calculated by approximating the value of V_x across series connected transistors and it is used to calculate the currents through each of the two transistors in series (I_{n1} and I_{n2}). The algorithm first assigns the stage output voltage (V_{out}) to V_{Max} and 0 V to V_{Min}. It then assigns V_x the mid-point voltage of V_{Max} and V_{Min}. Through this value of V_x, it calculates the currents through each of the two transistors I_{n1} and I_{n2}, using Eq. (3.1). It then compares I_{n1} and I_{n2}, and reduces the separation between V_{Max} and V_{Min} until the difference between the currents I_{n1} and I_{n2} is less than $LIMIT$ (1μA). Once the current through the pull-down network is calculated, it is compared with I_p (current through the pull-up network of the same stage). This current is calculated using an algorithm for PMOS transistors, similar to Figure 3.8, where I_p will be calculated by assigning (V_{out}-V_{dd}) to V_{ds} and (V_{in}-V_{dd}) to V_{gs} (note as shown in Figure 3.6, the pull-up network of second stage has just one transistor switched on). The currents through the pull-up and pull-down networks are used to adjust the logic threshold voltage of each stage $V_{in,i}$ (step-17 to step-21 in Figure 3.7), until the difference is less than $LIMIT$ and at that point the algorithm returns the logic threshold voltage of the given gate.

3.3.2 Bridge Critical Resistance Calculation Algorithm

The critical resistance of a bridge is calculated through bridged net voltage approximation algorithm (Section 3.3.2.1 and Section 3.3.2.2) by using BSIM4 transistor model. The algorithm approximates the voltage on bridged nets (V_0 and V_1, Figure 3.1), while

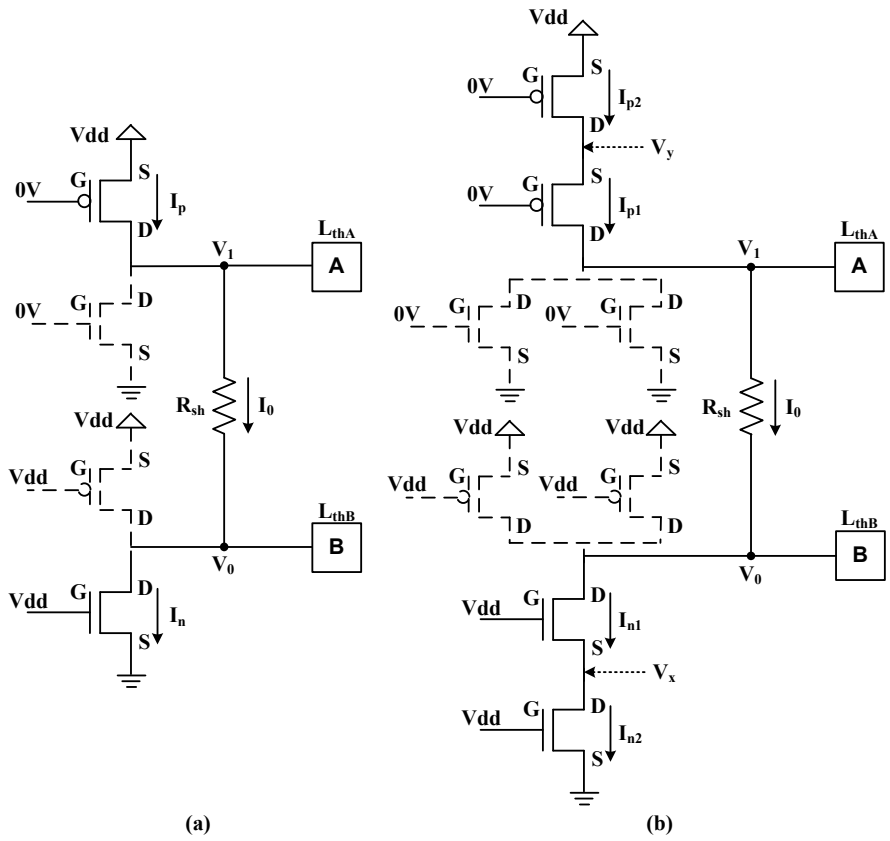


Figure 3.9: Bridge resistance examples: (a) A fault-site driven by two inverters; (b) A fault-site driven by 2-input NOR and 2-input NAND

considering both NMOS and PMOS transistors in gates driving the bridge (D_1 and D_2, Figure 3.1). The algorithm presented in Section 3.3.2.1 approximates the voltage on bridged nets based on linear search algorithm, while the algorithm proposed in Section 3.3.2.2 improves the algorithm in Section 3.3.2.1 by approximating the voltage based on binary search algorithm. The calculated bridged net voltages (V_0 and V_1, Figure 3.1) are used to calculate the critical resistance (R_{sh}, Figure 3.1) of a bridge fault-site. As an example Figure 3.9-(a) shows a fault-site, where two inverters are driving a bridge (R_{sh}) and I_0 is the current through the resistor. The transistors drawn using dashed lines represent switched off transistors. The value of critical resistance (R_{sh}) shown in Figure 3.9 can be calculated by the following expression:

$$R_{sh} = \frac{(V_1 - V_0)}{I_0} \tag{3.3}$$

3.3.2.1 Linear Search Algorithm

This section explains the voltage approximation algorithm based on linear search method for calculating the voltage on bridged nets (V_0 and V_1) for Eq. (3.3) by using the algorithm shown in Figure 3.10 for both NMOS and PMOS transistors. The fault-site

Input: I_0
Output: V_0 or V_1
1: Read the PTM model card.
   // Model card is needed for parameters in Eq. (3.1)
2: $V_{tmp} = 0$, STEP = 0.0005
3: $I_{tmp} = 0$, LIMIT = 0.005
4: **if** NMOS **then**
5: **repeat**
6: $V_{tmp} = V_{tmp}$ + STEP
7: $I_{tmp} = I_n(V_{tmp})$; with $V_{bs} = 0$ and $V_{gs} = V_{dd}$
   // I_{tmp} is calculated by using Eq. (3.1)
8: **until** $\left[\frac{|I_0 - I_{tmp}|}{I_0} \geq LIMIT\right]$
9: **return** (V_{tmp})
   // $V_0 = V_{tmp}$
10: **else**
11: **repeat**
12: $V_{tmp} = V_{tmp}$ - STEP
13: $I_{tmp} = I_p(V_{tmp})$; with $V_{bs} = 0$ and $V_{gs} = -V_{dd}$
   // I_{tmp} is calculated by using Eq. (3.1)
14: **until** $\left[\frac{|I_0 - I_{tmp}|}{I_0} \geq LIMIT\right]$
15: **return** $(V_{dd} + V_{tmp})$
   // $V_1 = V_{dd} + V_{tmp}$
16: **end if**

Figure 3.10: Bridged net voltage linear search algorithm for N/P transistor.

Input: V_0 or V_1
Output: I_0

1: Read the PTM model card.
   // Model card is needed for parameters in Eq. (3.1)
2: **if** NMOS **then**
3: $V_x = \frac{V_0}{2}$
   // Figure 3.9-(b) shows V_x
4: **repeat**
5: $I_{n1} = I_n(V_0 - V_x)$;
   with $V_{bs} = -V_x$ and $V_{gs} = V_{dd} - V_x$
6: $I_{n2} = I_n(V_x)$; with $V_{bs} = 0V$ and $V_{gs} = V_{dd}$
   // I_{n1} *and* I_{n2} *are calculated by using Eq.* (3.1)
7: $I_0 = \frac{(I_{n1}+I_{n2})}{2}$
8: $V_x = I_n^{-1}(I_0)$; with $V_{bs} = 0V$ and $V_{gs} = V_{dd}$
   // Using Algorithm shown in Figure 3.10
9: **until** $(|I_{n1} - I_{n2}| \geq 1\mu A)$
10: **else**
11: $V_y = \frac{(V_1+V_{dd})}{2}$
   // Figure 3.9-(b) shows V_y
12: **repeat**
13: $I_{p1} = I_p(V_1 - V_y)$;
   with $V_{bs} = V_{dd} - V_y$ and $V_{gs} = -V_y$
14: $I_{p2} = I_p(V_y - V_{dd})$; with $V_{bs} = 0V$ and $V_{gs} = -V_{dd}$
   // I_{p1} *and* I_{p2} *are calculated by using Eq.* (3.1)
15: $I_0 = \frac{(I_{p1}+I_{p2})}{2}$
16: $V_y = I_p^{-1}(I_0)$; with $V_{bs} = 0V$ and $V_{gs} = -V_{dd}$
   // Using Algorithm shown in Figure 3.10
17: **until** $(|I_{p1} - I_{p2}| \geq 1\mu A)$
18: **end if**
19: **return** (I_0)

Figure 3.11: I_0 approximation algorithm for two transistor in series.

shown in Figure 3.9-(a) is used as an example. Using the logic threshold voltage (L_{thA}, obtained through the algorithm shown in Figure 3.7 and Figure 3.8 in Section 3.3.1) of the driven gate "A" (Figure 3.9), V_1 is L_{thA}, which can be used to calculate I_0 through the I-V_{ds} relationship shown in Eq. (3.1), i.e., $I_0 = I_p(V_{ds,p})$, where $V_{ds,p} = L_{thA} - V_{dd}$. The only unknown variable left in Eq. (3.3) is V_0, which can be approximated by using the algorithm shown in Figure 3.10 (since $I_0 = I_n(V_0)$, which implies $V_0 = I_n^{-1}(I_0)$). In Figure 3.10 the value of V_0 is gradually incremented (step-6) until the relative difference of I_0 and I_n (I_n is represented by I_{tmp}) is smaller than the specified limit, as determined by step-8 of the algorithm. The value of V_0 is then used together with the other two variables (V_1 and I_0) to calculate R_{sh} using Eq. (3.3). The same procedure can be repeated for PMOS transistor, starting with the value of V_0 as the logic threshold of gate "B", i.e., L_{thB}. In case of transistors in parallel, the $\frac{W}{L}$ for the equivalent transistor is calculated by using Eq. (3.4) before starting the algorithm.

$$\left(\frac{W}{L}\right)_{equ} = \sum_{i=1}^{k} \left(\frac{W}{L}\right)_i \tag{3.4}$$

Next, how to approximate the value of I_0 in the case where two transistors are in series (Figure 3.9-(b)) is described. In this case, R_{sh} (Eq. (3.3)) is calculated starting with the logic threshold voltage of gate "B", i.e. $V_0 = L_{thB}$. V_0 is used to calculate I_0 using the algorithm shown in Figure 3.11 for (NMOS or PMOS) transistors in series. It can be seen that the currents through the two NMOS transistors (I_{n1} and I_{n2}) are calculated by approximating the value of V_x, starting with $\frac{V_0}{2}$ as shown in step-3. This is used to generate the intermediate values of I_0, which further improve the approximation of V_x (step-8). This process is repeated until the difference in I_{n1} and I_{n2} is smaller than 1μA, which usually requires very small (5 or less) number of iterations. The difference of less than 1μA provides a close approximation of I-V_{ds} when compared with the HSPICE results. The value of I_0 is then used to calculate V_1 using the algorithm shown in Figure 3.10. Finally, all three variables (V_0, V_1 and I_0) are used to calculate R_{sh} using Eq. (3.3). A similar algorithm is used for PMOS transistors (in series, by step-11 to step-17). It can be extended to calculate I_0 for more than 2 transistors in series, for example in case of 3 transistors, step-3 is changed with $\frac{V_0}{3}$ and approximating currents through each transistor i.e. I_{n1}, I_{n2} and I_{n3} using their respective V_{ds} voltages to calculate the value of I_0. Note that previously [94] has used Eq. (3.2) and a V_{ds} approximation method to calculate the critical resistance of a bridge in nominal operating conditions, but that work does not include the effect of process variation, which is described in Section 3.3.3.

3.3.2.2 Binary Search Algorithm

The voltage on bridged nets is calculated using the algorithm shown in Figure 3.10 by linearly increasing the voltage value (step-6) until the relative difference between the

Input: I_0
Output: V_0 or V_1
1: Read the PTM model card.
   // *Model card is needed for parameters in Eq.* (3.1)
2: $V_{tmp} = 0$
3: $I_{tmp} = 0$, LIMIT $= 0.005$
4: **if** NMOS **then**
5: $V_{Max} = V_{dd}$, $V_{Min} = 0$
6: **repeat**
7: $V_{tmp} = (V_{Max} + V_{Min})/2$
8: $I_{tmp} = I_n(V_{tmp})$; with $V_{bs} = 0$ and $V_{gs} = V_{dd}$
   // I_{tmp} *is calculated by using Eq.* (3.1)
9: **if** $I_{tmp} \geq I_0$ **then**
10: $V_{Max} = V_{tmp}$
11: **else**
12: $V_{Min} = V_{tmp}$
13: **end if**
14: **until** $\left[\frac{|I_0 - I_{tmp}|}{I_0} \geq LIMIT\right]$
15: **return** (V_{tmp})
   // $V_0 = V_{tmp}$
16: **else**
17: $V_{Max} = 0$, $V_{Min} = -V_{dd}$
18: **repeat**
19: $V_{tmp} = (V_{Max} + V_{Min})/2$
20: $I_{tmp} = I_p(V_{tmp})$; with $V_{bs} = 0$ and $V_{gs} = -V_{dd}$
   // I_{tmp} *is calculated by using Eq.* (3.1)
21: **if** $I_{tmp} \geq I_0$ **then**
22: $V_{Min} = V_{tmp}$
23: **else**
24: $V_{Max} = V_{tmp}$
25: **end if**
26: **until** $\left[\frac{|I_0 - I_{tmp}|}{I_0} \geq LIMIT\right]$
27: **return** $(V_{dd} + V_{tmp})$
   // $V_1 = V_{dd} + V_{tmp}$
28: **end if**

Figure 3.12: Binary search algorithm for calculating voltage on bridged nets

currents is smaller than the specified limit (step-8). This section presents an algorithm shown in Figure 3.12 to improve the linear search algorithm shown in Figure 3.10 by replacing the linearly step (step-6 in Figure 3.10) to a binary search method (step 7 to step 13 in Figure 3.12).

As an example, the bridge fault-site shown in Figure 3.9-(a) is also used here. To calculate the critical resistance using Eq. (3.3), the value of V_1 is equal to L_{thA} (obtained through the algorithm discussed in Section 3.3.1) and it can be used to calculate I_0 is by using Eq. (3.1). The only unknown variable left in Eq. (3.3) is V_0, which can be approximated by using the algorithm shown in Figure 3.12 (since $I_0 = I_n(V_0)$, which implies $V_0 = I_n^{-1}(I_0)$). The algorithm shown in Figure 3.12 assigns V_{dd} to V_{Max} and 0-V to V_{Min} respectively. It then assigns the mid-point voltage value (between V_{Max} and V_{Min}) to V_0. This is used to generate I_n (represented by I_{tmp}). If $I_n \geq I_0$, that means the value of V_0 is between V_{tmp} and 0 V; otherwise it is between V_{dd} and V_{tmp} (as ideally, $I_n = I_p = I_0$). This process is repeated until the relative difference between I_0 and I_n is smaller than the specified limit, as determined by step-14 of the algorithm. In this work, $LIMIT$ is set to 0.005, as it was determined empirically that this value provides high accuracy when compared with HSPICE (similar to analysis shown in Table 3.1). The algorithm converges quickly and requires only small number (15 or less) of iterations. The value of V_0 is then used together with the other two variables (V_1 and I_0) to calculate R_{sh} using Eq. (3.3). The same procedure can be repeated for PMOS transistor, starting with the value of V_0 as the logic threshold of gate "B", i.e., L_{thB}. This approximation algorithm (Figure 3.12) is an improvement over the one presented in Figure 3.10 and is on average 41 times faster, while achieving the same accuracy. In case of transistors in parallel, the effective $\frac{W}{L}$ is calculated by using Eq. (3.4) before starting the algorithm. For transistors in series (Figure 3.9-(b)), the algorithm shown in Figure 3.11 is used.

3.3.3 Incorporation of PVT Variation

The first two stages of the proposed modelling technique (Section 3.3.1, Section 3.3.2) are used to calculate critical resistance of a bridge fault-site in nominal operating conditions. This section explains how the effect of process, voltage and temperature variation is incorporated in the proposed modelling technique. The variation effects are incorporated in transistor model, for example Eq. (3.1), and therefore affect logic threshold voltage of the driven gates and voltages on bridged nets, leading to change in logic fault behaviour of the bridge fault-site.

The effect of process variation is considered for both die-to-die variation (D2D) and within-die variation (WID). The D2D variation is modelled by using mutually independent parameter fluctuations (without spatial correlation) while the WID variation is modelled by spatial correlation effects on transistor length. A recent study describes the parameter extraction technique (for process variation) using a 65-nm CMOS library

Table 3.2: Varied process parameters in die-to-die variation

Parameter	Mean (μ)	Std. Deviation (σ)
L	60 nm	$\pm$4% (2.4 nm)
V_{thn}	0.423 V	$\pm$5% (21.15 mV)
V_{thp}	-0.365 V	$\pm$5% (18.25 mV)
μ_{effn}	491 cm^2/V.s	$\pm$21% (103.1 cm^2/V.s)
μ_{effp}	57.4 cm^2/V.s	$\pm$21% (12.05 cm^2/V.s)

with a PTM model [10, 21]. Three transistor parameters are recognised as the leading sources of process variation, which include: gate length (L), threshold voltage (V_{th}), and mobility (μ_{eff}). These parameters follow Gaussian distribution (Eq. 1.1, Chapter 1) with standard deviations of 4% for L, 5% for V_{th} and 21% for μ_{eff} (Figure 1.1 in Chapter 1). Negligible spatial correlation is found in between these parameters, i.e., they can be treated as independent random variables following Gaussian distribution. These results are validated by comparing with the measured data using a fabricated device. Simulation in this work is based on a ST Microelectronics 65-nm gate library using the same PTM model cards that are used in [10], which is why this work has also assumed the same parameter fluctuations. The mean and standard deviation for both NMOS/PMOS transistors are shown in Table 3.2. The calculated standard deviation of V_{th} is also compared with the σV_{th} value using the relationship presented in Eq. (3.5) [138], and only a small difference (around 5 mV) is found for both the NMOS and PMOS transistors. Recent research has shown that it is sufficient to consider $\pm 3\sigma$ variation of process parameters, when modelling process variation for logical part of the design [2, 139], and higher variation effects ($\pm 6\sigma$ or more) are considered for (SRAM and Flash) memories [12]. This work also deals with the logical part of the design, which is why this work have also considered $\pm 3\sigma$ variation effects.

$$\sigma V_{th} = 3.19 \times 10^{-8} \frac{\left(t_{ox} \cdot N_A^{0.4}\right)}{\sqrt{L_{eff} \cdot W_{eff}}} [V] \tag{3.5}$$

where N_A, L_{eff} and W_{eff} are the average channel doping, effective channel length and width respectively.

The effect of within die variations are analysed, by varying only the gate length of different transistors using a spatial correlation model [22]. As pointed out by several publications, gate length is a leading source of process variation and it has shown correlated variation effects due to lithography [23, 24, 25, 140]. The spatial correlation model that correlates the gate length of different transistors within the same die is given by the following relationship:

$$\rho = \begin{cases} 1 - \dfrac{x}{X_L}\left(1 - \rho_B\right), & x \leq X_L \quad (3.6) \\ \rho_B, & x \geq X_L \quad (3.7) \end{cases}$$

where ρ is the correlation coefficient that relates the gate length of different transistors, X_L is the correlation length, ρ_B is the correlation baseline and x is the separation between transistors.

Temperature variation is incorporated in Eq. (3.1) by using temperature dependent models of threshold voltage, mobility and saturation velocity, as described in BSIM4 transistor model [86]. The temperature dependent threshold voltage model is given by Eq. (3.8):

$$\begin{aligned} V_{th}(T) = V_{th}(TNOM) + \\ \left(KT1 + \frac{KT1L}{L_{eff}} + KT2 \cdot V_{bseff}\right) \cdot \\ \left(\frac{T}{TNOM} - 1\right) \end{aligned} \quad (3.8)$$

where $V_{th}(T)$ is the temperature dependence of threshold voltage, $V_{th}(TNOM)$ is the nominal transistor threshold voltage, T is the circuit temperature, $TNOM$ is the transistor model reference temperature and its nominal value is 25^oC, $KT1$ is the temperature coefficient for threshold voltage, $KT1L$ is the channel length dependence of the temperature coefficient for threshold voltage, $KT2$ is the body-bias coefficient of threshold voltage temperature effect, V_{bseff} is the effective body bias voltage. The temperature dependent mobility model is given by Eq. (3.9):

$$U0\left(T\right) = U0\left(TNOM\right) \cdot \left(T/TNOM\right)^{UTE} \quad (3.9)$$

where $U0(T)$ is the temperature dependence of mobility, $U0(TNOM)$ is the nominal transistor mobility, UTE is the mobility temperature exponent. The temperature dependent model of saturation velocity is given by Eq. (3.10):

$$VSAT\left(T\right) = VSAT\left(TNOM\right) - AT \cdot \left(T/TNOM - 1\right) \quad (3.10)$$

where $VSAT(T)$ is the temperature dependence of saturation velocity, $VSAT(TNOM)$ is the nominal transistor saturation velocity, AT is the temperature coefficient for saturation velocity. The temperature dependent models mainly rely on the ratio of circuit temperature (T) to model reference temperature $(TNOM)$, for example as shown in

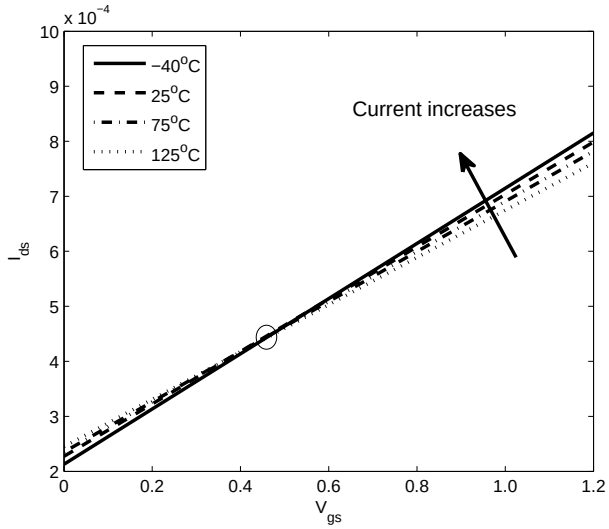
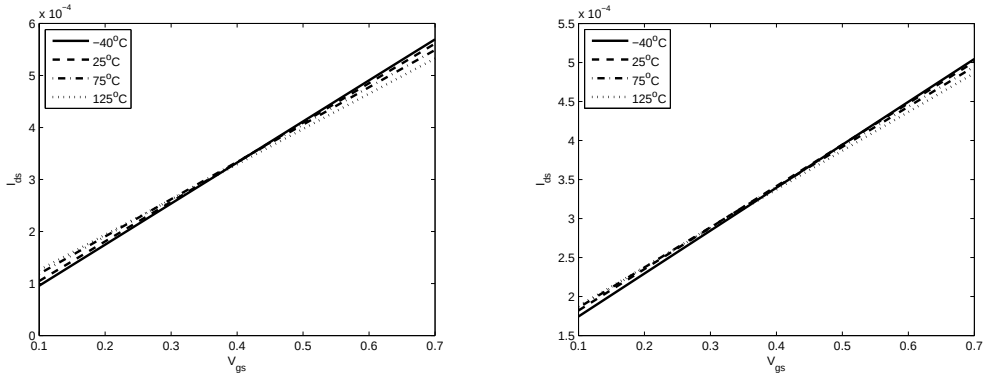


Figure 3.13: Temperature dependence of NMOS transistor drain current (I_{ds}) in 65-nm technology



(a) 45-nm technology

(b) 32-nm technology

Figure 3.14: Temperature dependence of NMOS transistor drain current (I_{ds}).

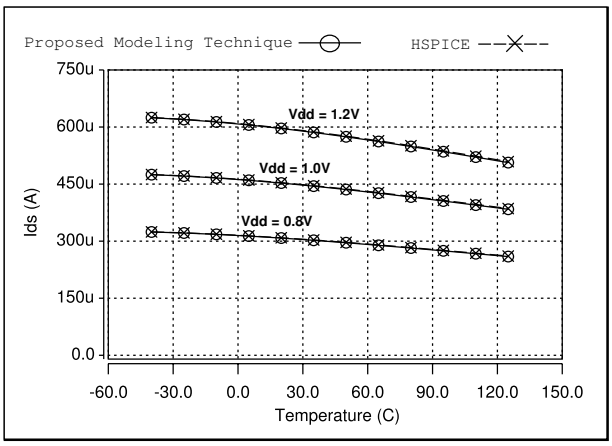


Figure 3.15: Drain current under different supply voltages and temperatures using proposed model and HSPICE

Eq. (3.8). When $T \neq TNOM$, additional values are calculated according to these equations and are used for calculating transistor drain current using Eq. (3.1).

As discussed at the beginning of this section, for a given fault-site, transistor drain current of the driving gate and logic threshold voltage of the driven gates are the two important parameters for calculating the bridge critical resistance. The effect of temperature variation on critical resistance calculation is analysed by simulating the change in transistor drain current and logic threshold voltage with the change in temperature. Figure 3.13 shows the effect of temperature variation on drain current of 65-nm NMOS transistor, while keeping V_{ds}=V_{dd}=1.2 V and increasing V_{gs} from 0 V to V_{dd}. As can be seen, at lower values of V_{gs}≤0.45 V, current increases with temperature, however this trend reverses at higher values of V_{gs} and current reduces with increase in temperature. The crossing point (marked in Figure 3.13) is also called Zero Temperature Coefficient (ZTC) and its effect is examined in several publications, see [141] for more details. Similarly, when considering logic threshold voltage of a gate, it was found that it also reduces as temperature increases. For 65-nm ST Microelectronics gate library, average reduction in logic threshold voltage is about 75 mV, when the temperature increases from -40^oC to 125^oC, operating at 1.2 V V_{dd}. Temperature also affects metal resistance as it increases with temperature. To analyse the effect of temperature variation on detectable resistance range, an experiment at 0.8 V V_{dd} using 350 fault-sites at -40^oC and 125^oC is conducted. It was found that about 86% fault-sites show higher detectability (higher detectable resistance range) at 125^oC and about 14% fault-sites show better detectability at -40^oC. This means that at higher temperatures, reduction in transistor current of driving gates and logic threshold voltages of the driven gates increases the detectable resistance range of majority of fault-sites. The behaviour of transistor drain current is also analysed by using 45-nm and 32-nm NMOS transistors (PTM model cards) and similar behaviour is found as shown in Figure 3.14, which means that this trend of detectable resistance range will continue for these technologies as well.

The variation in supply voltage is modelled by Eq. (3.1) in a straight forward manner because V_{ds} and V_{gs} change with supply voltage. Figure 3.15 shows the drain current under different supply voltages and temperatures using the proposed model and HSPICE using 65-nm technology. The temperature varies from -40^oC to 125^oC and the voltage varies from 0.8 V to 1.2 V, which are the operating temperatures and voltages for 65-nm ST Microelectronics gate library. As can be seen, it shows excellent correlation with HSPICE results.

3.4 Simulation Results

This section describes the simulation results using the proposed modelling technique presented in Section 3.3. A bridge fault simulation flow using the modelling technique is introduced in Section 3.4.1. A simulation of validating the logic threshold voltage calculation algorithm is considered in Section 3.4.2, while a validation of the critical resistance calculation algorithm is discussed in Section 3.4.3. These two algorithms are separately validated to determine the loss of accuracy due to each algorithm. Finally Section 3.4.4 validates the complete modelling technique when operating in nominal conditions and under the influence of process variation, which includes die-to-die and within-die variation. The effect of voltage and temperature variation is also considered.

3.4.1 Fault Simulation Flow

All simulations are conducted using a 65-nm ST Microelectronics gate library[3] and PTM transistor model card[4] [21] on Intel Xeon Quad Core 2.7 GHz processor with 12 GB RAM. The gate library consists of a variety of gates including simple (NAND, NOR, INV) and compound gates (AO22, OA22 etc.), each with different drive strengths. For illustration purposes 1.2 V and 25°C is used as the nominal operating voltage and temperature in all simulations. The proposed modelling technique is based on BSIM4 transistor model that provides detailed sets of equations for calculating each transistor parameter. The input value to each equation is provided by PTM transistor model card and the gate library. The proposed model is compared with HSPICE and to avoid any discrepancies, the same gate library and transistor model card are used with both techniques (HSPICE and proposed). HSPICE also uses BSIM4 transistor model as noted on HSPICE data sheet [83]. A logic fault simulation flow of bridge fault-site by using the proposed modelling technique is shown in Figure 3.16. The flow inputs are gate library and respective transistor models and the output is logic fault values (second part of Figure 2.2 in Chapter 2) of the bridge fault-site in the presence of process, voltage and temperature (PVT) variation.

[3] Appendix B shows SPICE description of three gates from the gate library.
[4] Appendix C shows SPICE description of the PTM transistor model card.

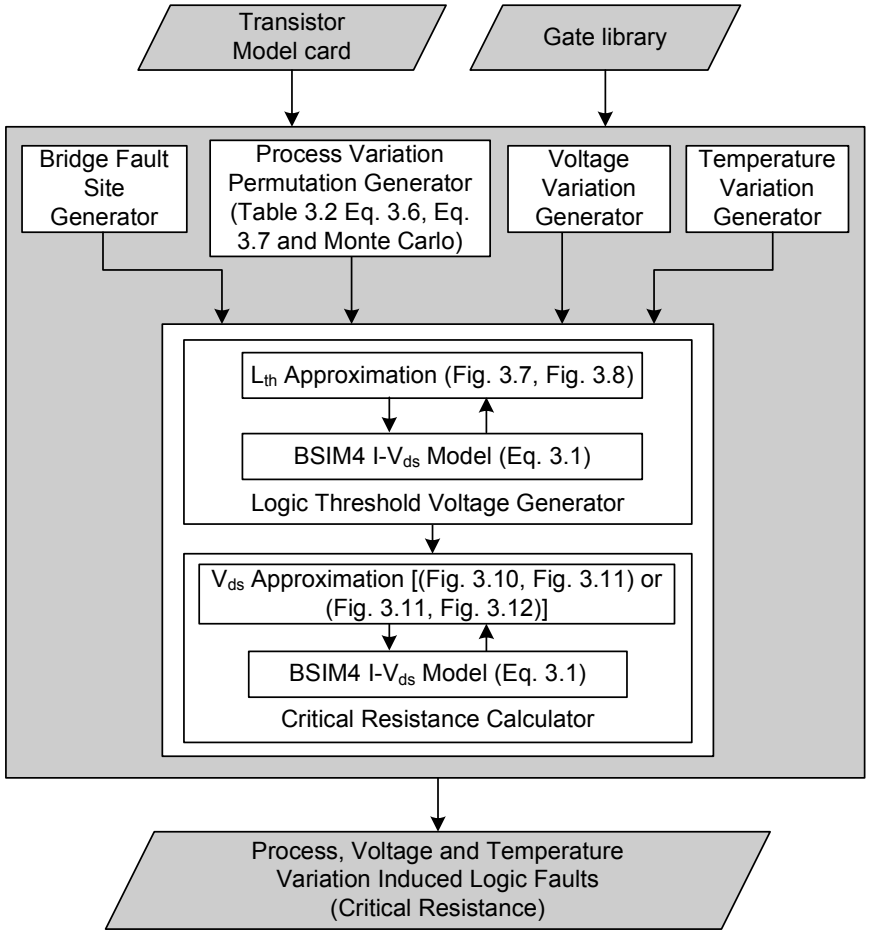


Figure 3.16: Proposed variation-aware bridge fault simulation flow

The flow has five main blocks as shown in Figure 3.16. The bridge fault-site is generated by randomly selecting (driving and driven) gates from the gate library, using n driven gates per fanout, where $n \in [1, 5]$ and only non-feedback and inter-gate bridges are generated by the bridge fault-site generator. Each of the driving gate is assigned a random input, while ensuring that the two nets are driven at opposite logic values to activate the bridge fault. This setup uses 350 fault-sites for each simulation because it was shown in [2] that the average number of fault-sites per design is less than 300 with coupling capacitance based layout extraction of bridges using ISCAS 85, 89 benchmarks. The effect of process variation is incorporated by the process variation permutation generator. Die-to-die variation is modelled by varying three parameters (L, V_{th} and μ_{eff}) using Gaussian distribution with mean and standard deviation as shown in Table 3.2. In total 600 permutations per fault-site are generated through Monte-Carlo simulation. The number of permutations are based on a recent study, which shows that the probability of generating a unique logic fault follows the law of diminishing returns, as it reduces significantly after 500 permutations [3]. Within-die variation is modelled by varying the gate length (L) through the correlation coefficient (Eq. (3.6) and Eq. (3.7)) with 600 permutations per fault-site. The effect of voltage variation is incorporated by varying supply voltages from 0.8 V to 1.2 V with the step size of 0.1 V using the voltage variation generator. The temperature variation generator generates three temperature values: -40°C, 25°C and 125°C, which are the minimum, nominal and maximum working temperatures for 65-nm ST gate library. These voltage and temperature values are used for demonstration purposes and the same flow can also be used for other values just as well. The outputs of these four blocks are fed to the main block which includes logic threshold voltage generator and critical resistance calculator. The logic threshold voltage generator uses BSIM4 transistor drain current model (Eq. (3.1)) and the L_{th} calculation algorithm (Section 3.3.1) to generate logic threshold voltages of a given fault-site, while including the effect of PVT variation. The output of these voltage values are fed to the critical resistance calculator. The critical resistance calculator uses Eq. (3.1) and voltage approximation algorithm (linear search in Section 3.3.2.1 or binary search in Section 3.3.2.2) to generate all PVT variation induced logic faults of bridge fault-site.

The fault simulation flow shown in Figure 3.16 has been implemented as a prototype software tool using C/C++. The software tool analyses the gate structure (transistor in series or in parallel) from given gate library and read parameter values both from gate library and transistor model card. This information (gate structure and parameter values) is used in the five main blocks shown in Figure 3.16. Bridge fault site generator selects (driving and driven) gates from the gate library randomly to build activated bridge fault-site. Process variation permutation generator generates random parameter values following Gaussian distribution. This is implemented based on parameters with mean and standard deviation as shown in Table 3.2, Eq. 1.1 in Chapter 1 and the random value generating function *rand*() in C/C++. Voltage and temperature variation generator generate voltage and temperature values from specific range. The logic

threshold voltage generator based on the algorithms shown in Figure 3.7 and Figure 3.8 uses BSIM4 I-V_{ds} equation Eq. (3.1) to build a data structure using C/C++ for each gate in the gate library. This is about 3000 lines of code in C/C++. The critical resistance calculator is built by using linear search algorithm (Figure 3.10 and Figure 3.11) and binary search algorithm (Figure 3.11 and Figure 3.12), and consists of more that about 2800 lines of code in C/C++. The software tool flow shown in Figure 3.16 is built without using any SPICE approximation algorithm.

3.4.2 Logic Threshold Voltage Calculation Algorithm Validation

The logic threshold voltage calculation algorithm (Section 3.3.1) is validated by comparing the results with HSPICE in nominal operating conditions (1.2 V and 25°C) and in the presence of PVT variation. When operating at nominal conditions, the comparison (accuracy and speed) of various gates is shown in Table 3.3. Table 3.3 shows gates that are expected to cause higher approximation error; higher error is expected in gates with transistors in series and compound gates. It can be seen that using the proposed logic threshold calculation algorithm, the error varies from 0.1% (INV "Inverter"; 0.7 mV error) to 3.1% (input-B of NAND4 "4-input NAND gate"; 20.1 mV error) when compared with HSPICE. Error is highest in case of NAND4 gate (four-input NAND) as it consists of 2 two-input NAND gates connected to a two-input NOR gate followed by an INV, therefore the error in logic threshold calculation is accumulated with each stage.

Furthermore, the effect of leakage current contributed by transistors that are switched off is investigated. For all the gates shown in Table 3.3, error due to leakage current is less than 2.1 mV (0.44%) and on average it is 0.7 mV (0.13%). To analyse further on the deviation of calculated logic threshold voltages from HSPICE results, the impact of body effect due to transistor stacking on threshold voltage is also investigated by using 20 different transistor configurations. For all configurations, using the proposed model, the difference is less than 0.98 mV (0.42%) with average difference of 0.86 mV (0.31%) in comparison to HSPICE. The last column of Table 3.3 shows the relative runtime improvement when comparing the proposed logic threshold calculation algorithm and HSPICE ($\frac{HSPICE}{LG}$). In comparison to HSPICE, the maximum improvement is in case of INV (2730 times), and least improvement is in case of NAND3 gate (3-input NAND), which is 5 times. This is because NAND3 has three transistors in series in the pull-down network and I_n approximation (Figure 3.8) for each transistor is needed to compute logic threshold voltage of each gate input. When considering all the gates shown in Table 3.3, average improvement is 197 times in comparison to HSPICE and in general considering all gates in the gate library, it was found that average improvement is 257 times.

Next, the influence of process variation on the accuracy of the proposed logic threshold voltage calculation algorithm is examined. This simulation uses process variation permutation generator (Figure 3.16) and the generated results are compared with HSPICE

Table 3.3: Logic threshold voltage generator in comparison with HSPICE in nominal operating conditions

Gate	Node	Lvth (V)			Time
		LG	HSPICE	Err (%)	$\frac{HSPICE}{LG}$
INV		0.5393	0.54	0.1	2730x
NAND3	A	0.566	0.5738	1.4	5x
	B	0.5976	0.6108	2.2	5x
	C	0.6222	0.6332	1.7	6x
NAND4	A	0.5648	0.5511	2.5	84x
	B	0.6216	0.6417	**3.1**	251x
	C	0.5668	0.5514	2.8	196x
	D	0.6268	0.6465	3.0	135x
NOR3	A	0.5535	0.5441	1.7	14x
	B	0.5343	0.5225	2.3	9x
	C	0.5121	0.5045	1.5	6x
NOR4	A	0.5343	0.5493	2.7	198x
	B	0.4869	0.4726	3.0	288x
	C	0.5414	0.5495	1.5	188x
	D	0.5201	0.5345	2.7	251x
AND3	A	0.5801	0.5708	1.6	8x
	B	0.6238	0.6403	2.6	7x
	C	0.6375	0.6513	2.1	16x
AND4	A	0.5695	0.5652	0.8	215x
	B	0.6141	0.6318	2.8	133x
	C	0.5777	0.5677	1.8	138x
	D	0.6217	0.6373	2.4	165x
OR3	A	0.5653	0.5534	2.2	8x
	B	0.546	0.5457	0.1	9x
	C	0.5261	0.5327	1.2	8x
OR4	A	0.5414	0.5428	0.3	167x
	B	0.5121	0.523	2.1	142x
	C	0.5343	0.5426	1.5	161x
	D	0.505	0.5038	0.2	167x
Avg. Speedup					197

* LG → Proposed Logic Threshold Voltage Calculation Algorithm

Table 3.4: Error (%) in logic threshold voltage calculation under the influence of process variation in comparison to HSPICE

Gate	Node	L_{th} Error (%)		
		Min	Max	Avg
INV		0.01	0.58	0.14
NAND3	A	0.26	3.54	1.88
	B	0.59	4.97	2.27
	C	0.64	4.66	2.11
NAND4	A	0.01	5.18	2.60
	B	0.66	5.57	4.17
	C	0.02	5.15	3.74
	D	0.20	5.19	4.31
NOR3	A	0.57	4.06	2.89
	B	0.47	3.45	2.37
	C	0.50	3.76	1.81
NOR4	A	0.01	**5.74**	3.88
	B	0.63	5.69	**4.58**
	C	0.14	5.19	4.00
	D	0.04	5.21	4.39
AND3	A	0.05	4.84	2.36
	B	0.04	4.89	2.78
	C	0.59	4.80	2.52
AND4	A	0.47	4.61	2.90
	B	0.19	5.06	3.49
	C	0.02	4.30	2.11
	D	0.10	4.26	2.88
OR3	A	0.22	5.21	3.25
	B	0.46	3.79	2.43
	C	0.08	4.25	1.90
OR4	A	0.04	5.52	2.06
	B	0.40	5.36	3.45
	C	0.02	4.73	1.90
	D	0.07	5.22	2.54

Table 3.5: Average error (%) in logic threshold voltage calculation under PVT variation in comparison to HSPICE

Temperature \ Voltage	0.8 V	0.9 V	1.0 V	1.1 V	1.2 V
-40^{o}C	2.31%	2.23%	2.18%	2.13%	2.13%
25^{o}C	3.11%	2.77%	2.50%	2.36%	2.23%
125^{o}C	3.27%	3.08%	2.96%	2.82%	2.80%

to examine the relative accuracy. In this simulation only die-to-die variation using three un-correlated parameters (L, V_{th} and μ_{eff}) are considered. The results are shown in Table 3.4 with minimum (Min), maximum (Max) and average (Avg) error per gate-input of each gate. The least error is observed in case of the simplest gate (INV) with 0.14% average error over 600 permutations. Highest deviation of 4.58% is observed in case of NOR4 gate (4-input NOR), because it consists of two NOR2 gates, connected to NAND2 gate, followed by an INV and error is accumulated with each stage. In any permutation over all gates, maximum observed error is 5.74% as in case of input-B of NOR4 gate.

Finally, the logic threshold voltage generator is validated under the effect of PVT variation by using the process variation permutation generator, voltage and temperature variation generators (Figure 3.16). The simulation is conducted by using 600 permutations of process variation at different voltage and temperature settings, when considering each gate-input per gate and average error is shown in Table 3.5. At a given temperature, the average error in logic threshold voltage calculation decreases as voltage increases. Similarly, at a given voltage, average error in logic threshold calculation increases as temperature increases from -40°C to 125°C. This is because transistor drain current reduces with supply voltage, and at a given voltage it reduces further with increase in temperature as shown in Figure 3.15. Since the algorithm (Section 3.3.1) terminates when the difference in currents through the pull-up and pull-down networks of a gate is less than 1-μA, this 1-μA difference in current becomes a bigger proportion of transistor currents at lower voltage (higher temperature) setting leading to higher accuracy error. A trivial change in termination criteria, for example reducing it further from 1-μA (see Table 3.1 in Section 3.3.1) can improve the accuracy at lower voltage and higher temperature setting.

3.4.3 Bridge Critical Resistance Calculation Algorithm Validation

The bridge critical resistance calculation uses the calculation algorithm discussed in Section 3.3.2.1 or Section 3.3.2.2. The following sub-sections (Section 3.4.3.1 and Section 3.4.3.2) validates these two algorithms by using the flow shown in Figure 3.16 without using the logic threshold voltage generation algorithm, where for a given bridge

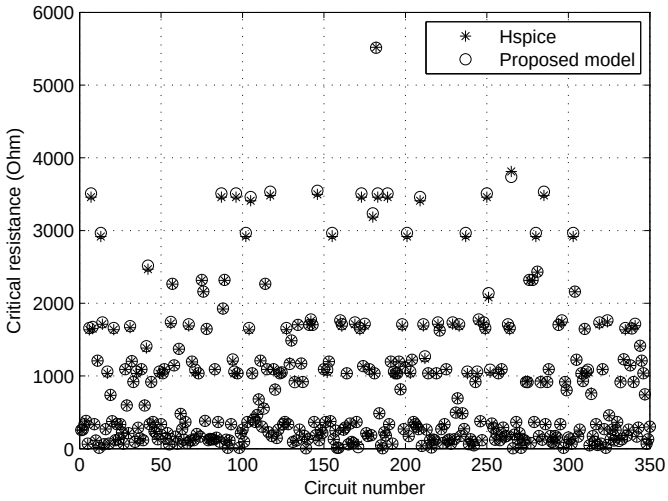


Figure 3.17: Nominal operating conditions: proposed technique and HSPICE.

fault-site, logic threshold voltage of the driven gates is calculated by using HSPICE to examine the calculation error of the proposed algorithm.

3.4.3.1 Linear Search Algorithm Validation

The critical resistance calculator shown in Figure 3.16 uses the bridged net voltage approximation algorithm with linear search method proposed in Section 3.3.2.1 is validated by comparing the results with HSPICE in nominal operating conditions and then under the influence of process variation by using un-correlated parameter fluctuations (die-to-die variation) of three parameters (L, V_{th} and μ_{eff}).

For the nominal operating conditions, the linear search algorithm (Figure 3.10 and Figure 3.11) is used to calculate the critical resistance (Eq. (3.3)) and the results are compared with HSPICE for 350 different fault-sites. The outcome is shown in Figure 3.17. It can be seen that the proposed linear search algorithm performs very well and high accuracy is achieved in comparison to HSPICE for all 350 fault-sites. These results are further elaborated in Table 3.6, which shows the calculated critical resistance for 6 fault-sites. These fault-sites include a number of cases where high approximation error is expected due to transistors in series. From Table 3.6, it can be seen that the difference in calculated critical resistance varies from 4Ω to 12Ω, when compared with HSPICE, leading to calculation error of 0.1% to 0.8% respectively. The maximum difference is in case of a bridge driven by a 3-Input NOR gate and a 3-Input NAND gate (0.8%). This is because 3 transistors are in series in the pull-up and pull-down networks of the gates driving high and low respectively, and each transistor requires voltage approximation using the method shown in Figure 3.11, to calculate the critical resistance of the bridge. This particular case also shows the maximum difference in critical resistance calculation, out of all the fault-sites considered in nominal operating conditions. This shows

Table 3.6: Nominal operating conditions: HSPICE results in comparison to the proposed technique.

Driving Gates (D_1, D_2)	Boolean Input(s)		R_{crit}, Ω		Error %
			HSPICE	Proposed Algorithm	
INV, INV	0	1	1661	1657	0.24
INV, 2-Input NAND	0	1, 1	1232	1224	0.65
2-Input NAND, 2-Input NAND	0, 0	1, 1	3752	3748	0.1
2-Input NOR, 2-Input NAND	0, 0	1, 1	2104	2093	0.52
2-Input NOR, 3-Input NAND	0, 0	1, 1, 1	1655	1647	0.48
3-Input NOR, 3-Input NAND	0, 0, 0	1, 1, 1	1500	1512	**0.8**

that the proposed linear search algorithm is accurate in nominal operating conditions with maximum difference of 0.8% in critical resistance calculation when compared with HSPICE.

The proposed linear search algorithm is validated under the influence of process variation by considering variations of the three un-correlated parameters (L, V_{th}, μ_{eff}) by using the process variation integrator as shown in Figure 3.16. The results are shown in Table 3.7 for the same set of fault-sites as in Table 3.6. It shows the calculated critical resistance under the influence of process variation for the two techniques (PLA (proposed linear search algorithm) and HSPICE). The low and high values of R_{crit} represent the minimum and maximum values of critical resistance, as a result of variation. It can be seen that the minimum and maximum differences are 1.4 Ω and 104 Ω respectively, which is also the maximum difference observed for all 350 fault-sites. It should be noted that bridge fault is detected over a range of resistance values and a test is not for a specific (discrete) resistance, as shown in [50]. For example, all bridges with resistance values $R_{sh} \in [0, R_{I3}]$ (Figure 2.2) are detectable with the same test. This means that the difference in resistance values (Table 3.6, Table 3.7) in between the proposed technique and HSPICE does not necessarily mean loss of test coverage. Next, to observe the effect of process variation on critical resistance of a bridge, a fault-site is used with two inverters as driving gates and only one gate as a driven gate. The change in critical resistance is shown in Figure 3.18, as can be seen the proposed technique achieves very high accuracy when compared with HSPICE.

Table 3.7: HSPICE results in comparison to the proposed technique using 3σ variations of un-correlated parameters (L, V_{th}, μ_{eff}).

Driving Gates		3σ Variation R_{crit} (Ω)		
(D_1, D_2)	Mechanism	Low	High	Time (s)
INV,	PLA	95	7557	15.7
INV	HSPICE	99	7574	292.4
INV,	PLA	410	9757	21.2
2-Input NAND	HSPICE	406	9762	311.2
2-Input NAND,	PLA	2548	14004	73
2-Input NAND	HSPICE	2569	13900	315.2
2-Input NOR,	PLA	61.6	11998	33.3
2-Input NAND	HSPICE	56	12080	309.1
2-Input NOR,	PLA	60	13544	33.4
3-Input NAND	HSPICE	67	13440	296
3-Input NOR,	PLA	75	13159	38.8
3-Input NAND	HSPICE	73.6	13090	329.4

* PLA → Proposed Linear Search Algorithm

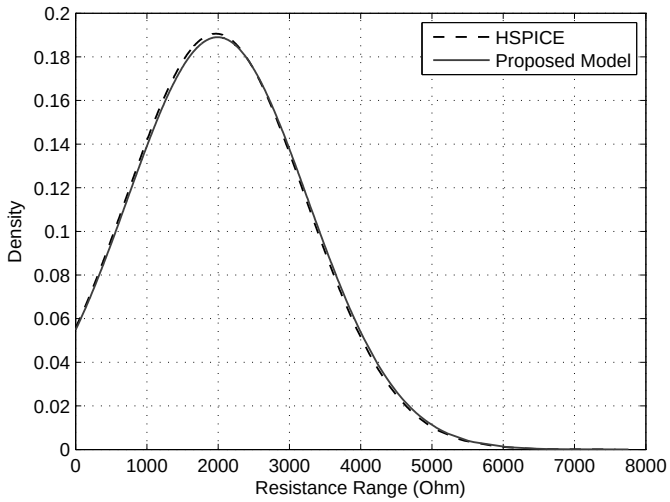


Figure 3.18: The effect of process variation on the critical resistance of a bridge driven by two inverters.

Table 3.8: HSPICE results in comparison with the critical resistance calculation using the binary search algorithm at nominal operating conditions and under the influence of process variation

Driving Gates (D_1, D_2)	Tech.	Nom (Ω)	$\pm 3\sigma$ Variation		Time (s)	Nom Err. %
			L (Ω)	H (Ω)		
INV (0)	PBA	1658	95	7559	0.37	
INV (1)	HSP	1661	99	7574	301	0.18
IVN (0)	PBA	1223	408	9758	0.74	
NAND2 (1,1)	HSP	1232	406	9762	307	0.73
NAND2 (0,0)	PBA	3732	2550	14001	0.82	
NAND2 (1,1)	HSP	3752	2569	13900	312	0.53
NOR2 (0,0)	PBA	2094	62	11998	1.20	
NAND2 (1,1)	HSP	2104	56	12080	319	0.47
NOR2 (0,0)	PBA	1649	59	13540	1.50	
NAND3 (1,1,1)	HSP	1655	67	13440	317	0.36
NOR3 (0,0,0)	PBA	1511	76	13157	1.86	
NAND3 (1,1,1)	HSP	1500	73.6	13090	316	0.73

* PBA → Proposed Binary Search Algorithm

HSP → HSPICE, L → Low, H → High, Nom → Nominal

The last column of Table 3.7 shows the simulation time of the two techniques. To demonstrate the relative speedup in critical resistance calculation using the proposed linear search algorithm, the simulation time of logic threshold generation is excluded as it is the same for both, either by using pre-computed database or HSPICE at runtime. It can be seen that the proposed technique is approximately 9 times faster than HSPICE and in general, 7 times faster for 350 fault-sites. These results clearly show that the proposed technique is sufficiently accurate and significantly faster than HSPICE. When considering the simulation time of logic threshold voltage by using HSPCIE together with the calculation time of critical resistance, the runtime improvement reduces from 7 times to only 10% faster.

3.4.3.2 Binary Search Algorithm Validation

The critical resistance calculator shown in Figure 3.16 can also be used with the binary search algorithm (Figure 3.11 and Figure 3.12), Section 3.3.2. The generated results using the proposed binary search algorithm is compared with HSPICE in nominal operating conditions and using un-correlated parameter fluctuations (die-to-die variation) of three parameters (L, V_{th} and μ_{eff}). The logic threshold voltage in this simulation is calculated by using HSPICE to examine the calculation error of the proposed algorithm.

Table 3.9: Average error (%) in critical resistance calculations under PVT variations in comparison to HSPICE

Temperature \ Voltage	0.8 V	0.9 V	1.0 V	1.1 V	1.2 V
-40^{o}C	0.65%	0.61%	0.59%	0.57%	0.56%
25^{o}C	0.69%	0.69%	0.66%	0.62%	0.60%
125^{o}C	1.73%	1.47%	1.31%	1.19%	1.11%

Table 3.8 shows the results in nominal operating conditions and when considering the influence of die-to-die variation on critical resistance calculation of a bridge fault-site. The fault-sites shown in Table 3.8 include a number of cases where high approximation error is expected due to gates with transistors in series, which is the same set of fault-sites as in Table 3.6. At nominal operating conditions, the difference in critical resistance varies from 0.18% (fault-site driven by INV-INV with 3Ω difference) to 0.73% (fault-site driven by 3-input NOR and 3-input NAND gates with 11Ω difference) when compared with HSPICE. Generally, the difference in critical resistance calculation increases with higher number of transistors in series in the pull-up and pull-down networks as in case of fault-site driven by NOR3-NAND3 gates. This is because each transistor requires voltage approximation algorithm (Figure 3.12) discussed in Section 3.3.2 to calculate critical resistance of the bridge. The error contribution due to leakage current is also analysed and for all the fault-sites shown in Table 3.8, max difference is 0.13% leading to 2Ω difference in critical resistance calculation, as in case of a fault-site driven by NOR3-NAND3 gates. This clearly demonstrates the accuracy of the critical resistance calaulation using the proposed binary search algorithm in nominal operating conditions.

Table 3.8 also shows the minimum (low) and maximum (high) values of bridge critical resistance, as a result of process variation across $\pm 3\sigma$ range. The minimum and maximum differences are 2Ω (fault-site driven by INV-NAND2) and 101Ω (fault-site driven by NAND2-NAND2) respectively, which is also the maximum difference observed for all 350 fault-sites. The second last column of Table 3.8 shows the simulation time using the two methods (Proposed and HSPICE). The proposed method is approximately 376 times faster than HSPICE and in general, 287 times faster for 350 fault-sites. This is a significant speedup in comparison to the algorithm proposed in Section 3.3.2.1, which is on average 7 times faster than HSPICE. Note that the simulation time of logic threshold generation is excluded to examine the relative speedup, in comparison to HSPICE, using the proposed algorithm for critical resistance calculation.

The combined effect of (die-to-die) process, voltage and temperature (PVT) variation using the proposed bridge critical resistance calculation algorithm is considered next and the results are shown in Table 3.9. This shows the average error in bridge critical resistance calculation using the proposed algorithm and HSPICE when considering 350 fault-sites and it shows the same trend as observed from Table 3.5 and related discussion

(on last paragraph of Section 3.4.2), at a given temperature, the average error in bridge critical resistance calculation decreases as voltage increases and at a given voltage, the average error increases as temperature increases.

3.4.4 Proposed Modelling Technique Validation

This section shows simulation results to validate the complete modelling flow (Figure 3.16) and compare the results with HSPICE to examine the net effect on accuracy and speed. In this simulation, the proposed algorithm for logic threshold (L_{th}) voltage generation is used for each fault-site and L_{th} is then used to calculate the critical resistance of each of 350 different bridge fault-sites. The calculation of critical resistance uses the binary search algorithm (Section 3.3.2.2) which is about 41 time faster than the linear search algorithm (Section 3.3.2.1).

3.4.4.1 Die-to-die Variation

The results at nominal operating conditions (1.2 V, 25°C) are shown first and then the effect of die-to-die process variation is incorporated and discussed. The results are shown in Table 3.10 for a selected number of fault-sites for which high approximation error is expected due to gates with transistors in series. For all fault-sites, Table 3.10 shows the comparison at nominal operating conditions (marked "Nom"), it can be seen that when compared with HSPICE, the difference in critical resistance varies from 8Ω (bridge driven by INV-INV) to 31Ω (bridge driven by NOR3-NAND3) leading to calculation error of 0.48% to 2.07% respectively. High error (in case of NOR3-NAND3) is because 3 transistors are in series in the pull-up and pull-down networks of the two gates driving the bridge, and each transistor requires voltage (V_{ds}) approximation using the algorithm proposed in Section 3.3.2, for bridge critical resistance calculation. Error due to leakage current for all the fault-sites shown in Table 3.10 is also analysed and max difference is 0.2% leading to 3Ω difference in critical resistance calculation, as in case of a fault-site driven by NOR3-NAND3 gates.

When considering the effect of die-to-die process variation, Table 3.10 shows the minimum (low) and maximum (high) values of bridge critical resistance (R_{crit}). The minimum and maximum differences are 3Ω (fault-site driven by INV-NAND2) and 132Ω (fault-site driven by NOR3-NAND3) respectively, which is also the maximum difference observed for all 350 fault-sites. Figure 3.19 shows the effect of process variation on critical resistance calculation of a bridge fault-site driven by two NAND2 gates (with inputs [0,0] and [1,1]). It can be observed from these results (Table 3.10, Figure 3.19) that the proposed modelling technique achieves high accuracy (worst case error of 2.07%) when compared with HSPICE. When considering the combined effect of PVT variation over

Table 3.10: HSPICE results in comparison with the proposed technique at nominal operating conditions and under the influence of die-to-die process variation

Driving Gates (D_1, D_2)	Tech.	Nom (Ω)	$\pm 3\sigma$ Variation		Nom Err. %
			Low (Ω)	High (Ω)	
INV (0)	PM	1653	92	7555	
INV (1)	HSPICE	1661	99	7574	0.48
IVN (0)	PM	1223	409	9754	
NAND2 (1,1)	HSPICE	1232	406	9762	0.73
NAND2 (0,0)	PM	3732	2551	14022	
NAND2 (1,1)	HSPICE	3752	2569	13900	0.53
NOR2 (0,0)	PM	2080	63	11991	
NAND2 (1,1)	HSPICE	2104	56	12080	1.14
NOR2 (0,0)	PM	1631	62	13548	
NAND3 (1,1,1)	HSPICE	1655	67	13440	1.45
NOR3 (0,0,0)	PM	1531	79	13222	
NAND3 (1,1,1)	HSPICE	1500	73.6	13090	**2.07**

* PM → Proposed Modelling Technique

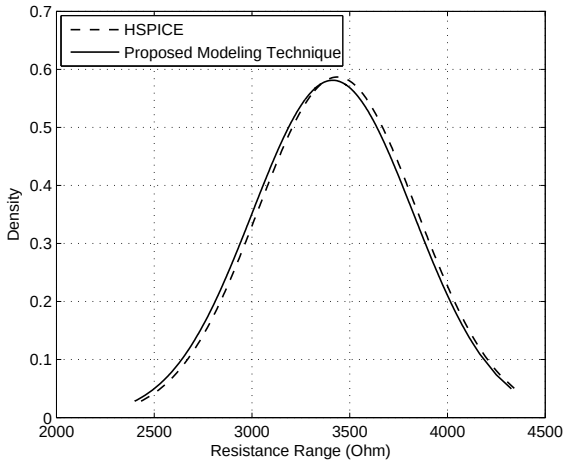


Figure 3.19: The effect of process variation on the critical resistance of a bridge driven by two NAND2 gates

all (350) fault-sites (detials are shown in Section 3.4.4.3), the worst case error is 2.64% when operating at 0.8 V and 125°C.

From results presented in this section, the following two observations are observed that can further improve the proposed modelling technique. Firstly, since the proposed technique utilises BSIM4, the upper bound in accuracy is that of BSIM4 transistor model. However, this technique (Section 3.3) does not depend on a specific transistor model and can be updated using a more accurate transistor model to achieve higher accuracy.

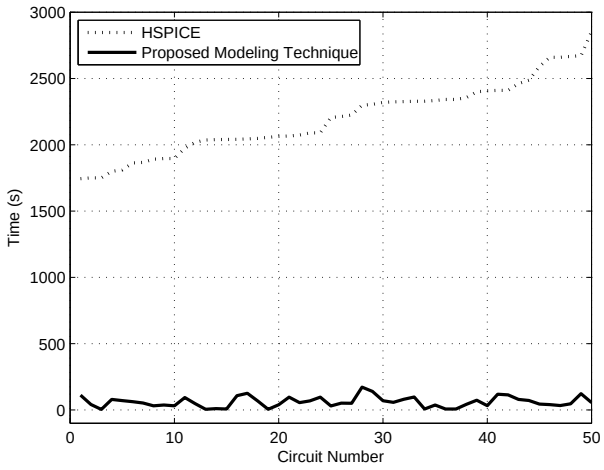


Figure 3.20: Computation time improvement: proposed technique vs HSPICE

Secondly, it is observed from the results that in comparison to HSPICE, the accuracy of the proposed technique reduces with higher number of transistors in series. For example, the difference in critical resistance calculation increases (compared to HSPICE) with higher number of transistors in series in the pull-up and pull-down networks as in case of fault-site driven by NOR3-NAND3 gates (last row of Table 3.10). The accuracy can be improved further by reducing the value of the parameter $LIMIT$ (Figure 3.7, Figure 3.8 and Figure 3.12), however that will increase the computation time of the algorithm.

To demonstrate the runtime improvement of the proposed modelling technique in comparison to HSPICE, Figure 3.20 shows the simulation time of the two modelling techniques (Proposed and HSPICE) using 50 randomly generated fault-sites with 600 permutations of un-correlated parameter fluctuations. The minimum and maximum time for the proposed technique is 5.2-s and 171.7-s respectively. In case of maximum simulation time (171.7-s) the fault-site comprises of three NAND3 (3-input NAND) gates as the fanout gates, which requires longer logic threshold calculation time. Table 3.3 shows that logic threshold calculation is slowest in case of NAND3 gate when compared to other gates and it is only 5 times faster than using HSPICE. However, using HSPICE the minimum and maximum time for critical resistance calculation is 1743.9-s and 2853.5-s respectively. In general, when considering 350 fault-sites, the proposed technique is 53 times faster than HSPICE. A recent study has reported that it took nearly a week with 8 computers working in parallel to generate a database for ISCAS 85, 89 benchmarks [3]. Using the proposed technique, we were able to re-generate the database for the same set of benchmarks in just over 3 hours with approximately the same accuracy. These results clearly show that the proposed technique is fast and accurate when compared with HSPICE. The proposed technique can be incorporated into an ATPG process through database generation, for example as in [50]. However, instead of generating database through HSPICE, the proposed technique can be used for efficient database generation. The only downside of database generation is that it is technology specific and a new

Table 3.11: Effect on gate output voltage (V_0) and logic threshold voltage (L_{th}) due to 1σ un-correlated parameter fluctuations.

Parameter	Range	V_{th}, L and μ_{eff}	
		Low	High
V_0 (R_{sh} = 0 Ω)	Min.	-3%	2.8%
	Max.	-24.8%	30%
	Avg.	-17.6%	18.0%
Logic Threshold	Min.	-0.5%	0.2%
	Max.	-13.3%	13.2%
	Avg.	-4%	3.7%

database will be needed for every technology node. Similarly, for fault simulation, the proposed technique can be used instead of HSPICE for very efficient fault simulation.

As described in Section 2.3.1, the newly generated logic faults (leading to loss of fault coverage) are due to change in gate output voltages (V_0 and V_1, Figure 3.1) of the driving gates, which is the voltages on bridged nets, and logic threshold voltages of the driven gates. The effect of die-to-die process variation on these two parameters is analysed and the results are shown in Table 3.11 for 1σ variation around mean. The effect of logic threshold variation is calculated by using the logic threshold voltage calculation algorithm (Section 3.3.1) for each gate input of all the gates in the gate library. The change in gate output voltages of the driving gates is calculated by using the proposed binary search algorithm (Section 3.3.2.2) with R_{sh} = 0Ω (V_0=V_1) on 350 fault-sites. On average, the logic threshold voltage varies in the range of [-4%, 3.7%] around the mean, and it is lower than the gate output voltage variation, which on average varies in the range of [-17.6%, 18%]. The results discussed in this section clearly demonstrate the efficiency (speed and accuracy) of the proposed modelling technique in comparison to HSPICE.

3.4.4.2 Within-die Variation

Gate length is a leading source of variation that shows correlated variation effects due to lithography [22] and its effect on bridge critical resistance calculation is examined in this section. The effect of within die spatial correlation using the proposed model[5] is investigated to compare its effect with die-to-die un-correlated parameter fluctuations on resistive bridges. This is analysed by varying (3σ) the gate length "L" using a spatial correlation model described in Section 3.3.3, with 600 permutations per fault-site. The correlation coefficient ρ is calculated using Eq. (3.6) and Eq. (3.7) respectively, with

[5]Section 3.4.4.1 confirmed the accuracy of the proposed model, which is why HSPICE is not used in this simulation.

Table 3.12: Spatial correlation of "L" using 3σ variation.

Driving Gate (D1, D2)	R_{crit} (Ω)	L with SC $\rho_B = 0.2$, $\rho \leq 0.8$	Un-correlated L
INV, INV	Low	251	237
	High	3724	4020
INV, 2-Input NAND	Low	674	537
	High	4396	5009
2-Input NAND, 2-Input NAND	Low	2855	2808
	High	5841	6311
2-Input NOR, 2-Input NAND	Low	331	216
	High	4760	5210
2-Input NOR, 3-Input NAND	Low	270	146
	High	5697	6006
3-Input NOR, 3-Input NAND	Low	243	122
	High	5639	5954

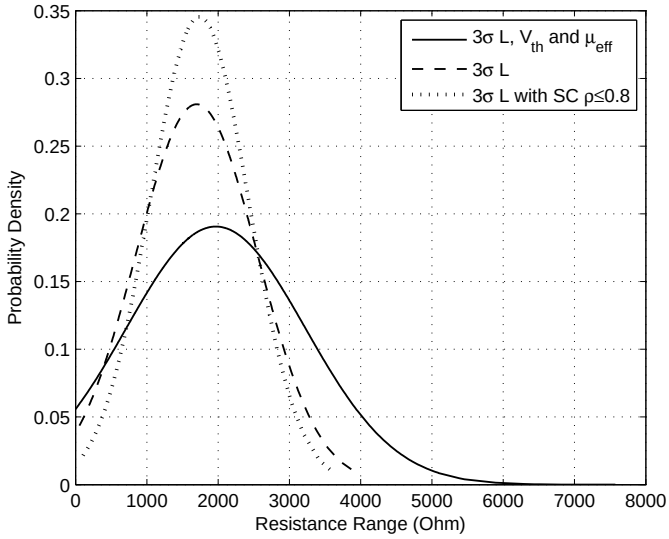


Figure 3.21: Comparison of spatially correlated and un-correlated parameter fluctuations on the critical resistance of a bridge driven by two inverters.

$\rho \leq 0.8$ and the baseline correlation ρ_B is 0.2. This is compared with un-correlated 3σ variation of "L". Table 3.12 shows the results for the same set of fault-sites as in Table 3.6. As can be seen the un-correlated variation of "L" results in wider spread than spatially correlated variations. This is further shown in Figure 3.21, which shows the distribution spread of a bridge critical resistance. Two inverters are driving this bridge with only one driven gate, and three different types of variations are used for this purpose: "L" with spatial correlation ($\rho \leq 0.8$ and $\rho_B = 0.2$), "L" without spatial correlation and by

using three un-correlated parameters (L, V_{th} and μ_{eff}). As expected, the 3σ variation of L, V_{th} and μ_{eff} results in a much wider spread of critical resistance than the other two types of variations. From this simulation it can be observed that the effect of spatial correlation is covered by considering variations due to three un-correlated parameter (L, V_{th} and μ_{eff}). From test generation point of view, it means that considering the three un-correlated parameter variations is likely to cover the complete logic fault domain due to spatially correlated parameter variation.

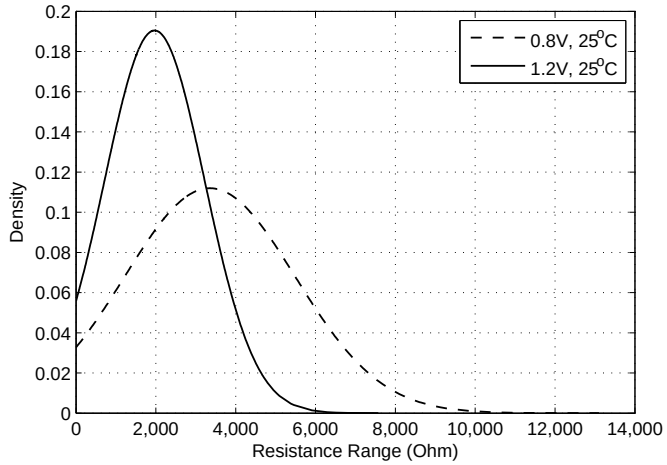
In this chapter, the effect of uncorrelated (die-to-die) and correlated (within-die) parameter variation is modelled by using [10] and [22] respectively. The proposed technique (Figure 3.16) is independent of any specific variation model and can be used to incorporate the effect of other parameters using their respective distribution models, for example as used in [2] and [139].

3.4.4.3 Process, Voltage and Temperature Variation

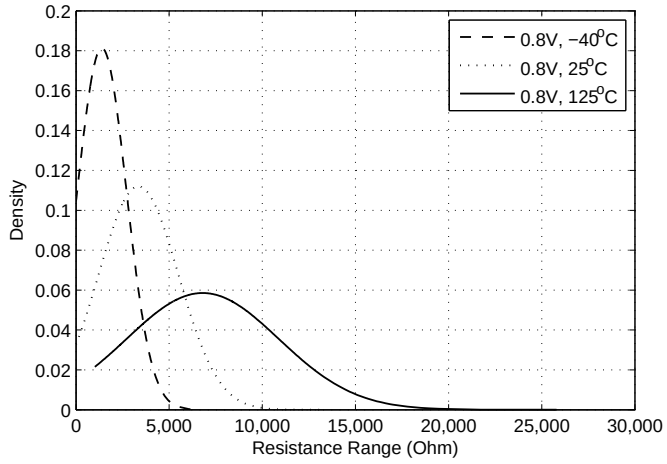
Next, the effect of process, voltage and temperature variations on resistance range coverage of bridge defect is shown. Result in Section 3.4.4.2 shows that considering three un-correlation parameters (die-to-die) variation is likely to cover the complete logic fault domain of within-die spatially correlated parameter variation, therefore in this simulation only three un-correlation parameters variation is considered. Figure 3.22(a) shows the effect of process and voltage variation on critical resistance of a bridge driven by two inverters. It can be seen that higher resistance range is detectable at lower voltage, which is in line with results reported in recent publications, for example see [95, 50] for more details, note that temperature is constant at both voltage settings. For the same fault-site, the lowest voltage (0.8 V that covers highest resistance range) is used and the temperature is changed to observe the effect of temperature variation. The simulation results are shown in Figure 3.22(b), which shows the effect of process and temperature variation on critical resistance of a bridge. It can be seen that maximum resistance is covered at highest temperature and lowest voltage setting, which is in line with the discussion in Section 3.3.3.

3.5 Concluding Remarks

This chapter has presented a fast and accurate technique to model the effect of process, voltage and temperature variation on resistive bridge defects by employing the BSIM4 (I-V_{ds}) transistor model. The effect of process variation is considered for both die-to-die variation and within-die variation. Die-to-die variation is modelled by using three un-correlated transistor parameters: L, V_{th} and μ_{eff}, using Gaussian distribution and the within-die variation is modelled by using spatially correlated gate length (L) variation.



(a) Effect of voltage variation



(b) Effect of temperature variation

Figure 3.22: Resistance range coverage for a fault-site driven by two inverters.

Variation in supply voltage is modelled by varying the supply voltage and temperature dependent transistor models are used to model the effect of temperature variation. The effect of voltage and temperature variation is incorporated by varying their respective values within prescribed (gate library) operating ranges. The proposed modelling technique employs two efficient algorithms: one is the calculation algorithm (Section 3.3.1) for logic threshold calculation of gates' inputs driven by the bridge fault-site and the other one is to calculate the critical resistance by using either of the two voltage approximation algorithms, linear search algorithm (Section 3.3.2.1) or binary search algorithm (Section 3.3.2.2). The proposed modelling technique is extensively validated through comparison with HSPICE when operating at nominal (1.2 V and 25°C) conditions. It is shown that the worst-case error for logic threshold generation algorithm is 3.1%. When calculating the critical resistance through the bridged net voltage approximation algorithm, it is shown that the binary search algorithm (worst-case error 0.73%) is 41 times

faster than the linear search algorithm (worst-case error 0.8%). Using the two calculation algorithms (logic threshold voltage calculation algorithm and critical resistance calculation algorithm based on binary search) together for resistive bridges under the influence of PVT variation over 350 fault-sites, it is found that the worst-case error is 2.64%, when compared with HSPICE. In terms of run-time improvement, it is shown that on average, over 350 fault-sites, the proposed modelling technique is 53 times faster than HSPICE (Figure 3.20). The effect of spatially correlated gate length (L) variation is found to be covered by the variation effects of the three un-correlated parameters: L, V_{th} and μ_{eff}. Therefore a test for considering die-to-die variation of three parameters is likely to cover all the logic faults due to within die spatial correlation. The proposed modelling technique has been demonstrated on a 65-nm gate library, and it can be used for evaluating the impact of process, voltage and temperature variation on bridge defect using other technology nodes. The modelling flow (Figure 3.16) will require a gate library with respective transistor model card, appropriate values of mean and standard deviation for the three transistor parameters (Table 3.2) and voltage and temperature variation ranges through the gate library.

Chapter 4

Accelerated Delay Fault Simulation of Resistive Bridge Defects

Chapter 3 developed a fast and accurate variation-aware fault modelling technique of resistive bridge defects when using logic test. As discussed in Section 2.1.3.2, Chapter 2, bridges with large resistance value cannot be detected by using logic test. Bridges with higher resistance values can be detected by using delay test. Delay fault simulation using SPICE is accurate but it requires a long computation time to simulate additional delay faults when considering process variation as discussed in Section 4.2. Recent publications [2, 79, 80] also observed that fault simulation using SPICE takes a long time to simulate the fault behaviour when considering the influence of process variation. This chapter addresses the problem of long simulation time of delay faults by analysing the delay behaviour of resistive bridges defect in the presence of process variation to identify the influential variables needed to accurately compute delay faults and then by reducing their computation time. A key identified variable is transient gate output voltage. An accelerated delay fault simulation methodology is developed by employing a three-step strategy to speed up the calculation of the transient gate output voltage without compromising accuracy. Results show that the proposed methodology is on average 17.4 times faster with 5.2% error in accuracy, when compared to SPICE. This chapter also analyses the delay behaviour of three transition delay test classes under the influence of process variation and results show that any class of delay test covers higher resistance coverage than logic test. Class-I delay test has the largest coverage among the three different classes both in nominal operating conditions and under process variation.

4.1 Introduction

The impact of fabrication process variation on integrated circuit performance and manufacturing test cannot be ignored [10, 6, 29, 142, 133, 143, 144]. As discussed in Section 1.1, Chapter 1, fabrication process variation is mainly due to sub-wavelength lithography, random dopant distribution, line edge roughness and stress engineering [10]. Resistive bridges represent a major class of defect in deep submicron CMOS. Using a 45-nm technology library, an increase in the number of logic faults is observed, and tests generated for nominal operating conditions without considering process variation led to as much as 10% loss of fault coverage [2]. As highlighted in Section 2.3, Chapter 2, a bottleneck in simulating additional faults due to process variation with SPICE, is long computation time as observed in recent publications [2, 79, 80]. The reported studies [2, 79, 80] have addressed this problem of long computation time through parallel processing by utilising cluster computing and storing fault simulation data in a database. The database is generated with SPICE per technology node for subsequent use, with automatic test pattern generation, fault simulation or diagnosis [2, 79, 80]. However, because of using SPICE, database generation requires a long computation time. In one reported study [2], this took nearly a week with 8 computers working in parallel to generate a database for ISCAS 85, 89 benchmarks. Similarly, for the study reported in [79, 80], it took 10 days with a 32-node cluster to generate a database for simple logic gates. The work presented in Section 4.4 shows that, using SPICE with a 65-nm design library, for a design with 3734 gates and 1194 bridge fault-sites, it was found that SPICE requires more than 32 days to complete delay fault simulation when using a Quad-Core 2.7 GHz processor with 12 GB RAM.

Chapter 3 addressed the problem of long computation time by developing an efficient variation-aware fault modelling technique for logic fault of resistive bridge defects, which is 53 times faster than SPICE (Synopsys HSPICE) with worst-case accuracy deviation of 2.6%. When considering transition delay fault test, there is no efficient process variation-aware delay fault simulation methodology as highlighted in Section 1.5, Chapter 2, which is the aim of this chapter. A fast and accuracte delay fault simulation methodology is proposed in this chapter. The methodology is accurate because it uses the most recent transistor model (BSIM4.7: Berkeley Short-Channel IGFET Model) [86]. It is fast because it employs a three-step strategy to accelerate the computation of transient gate output voltage, which is used to compute delay fault per resistive bridge fault-site. The first step analyses and identifies transistor models that affect the calculation accuracy of transient gate output voltage. The second step eliminates unnecessary electrical parameters within the retained transistor models obtained from the first step. The final step appropriately adjusts step size during different stages of transient analysis when calculating gate output voltage. The effect of process variation is modelled by incorporating fluctuations in three un-correlated transistor parameters: gate length (L), threshold voltage (V_{th}) and effective mobility (μ_{eff}), using Gaussian distribution [10].

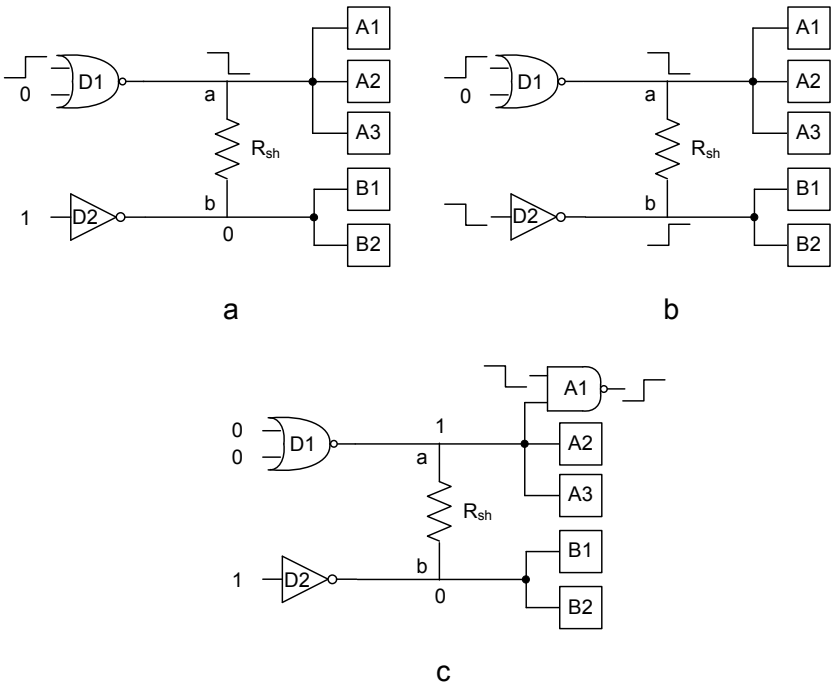


Figure 4.1: Transition delay test classification: (a) Class-I; (b) Class-II; (c) Class-III.

Simulations are conducted using a 65-nm gate library, and results are compared with SPICE. Results show that the proposed methodology is on average 17.4 times faster, with 5.2% error in accuracy (worst-case is 8.8%), when compared with SPICE.

This chapter is organised as follows: Section 4.2 analyses the delay behaviour of resistive bridge defects to show that the detectability is affected due to additional delay faults when considering process variation. It also identifies the key variables for computing delay faults. Section 4.3 presents the proposed delay fault simulation methodology. Section 4.4 discusses the results and Section 4.5 concludes this chapter.

4.2 Preliminaries

Transition delay test on resistive bridge defects is classified into three classes: Class-I, Class-II and Class-III [42] as mentioned in Section 2.1.3.2, Chapter 2. In this section, an example of Class-I is used to explain how delay faults are computed for a given bridge fault-site. Furthermore, when considering process variation, the increase in fault domain leading to test escapes is explained. This section identifies the most influential variables for computing delay faults with the aim to reduce their computation time.

The transition delay test of resistive bridge defects can be classified into three classes [42], which are shown in Figure 4.1. In Figure 4.1, R_{sh} represents the resistive bridge, D_1 and D_2 are the gates driving the bridge nets while A_1 to A_3 and B_1, B_2 are the driven gates.

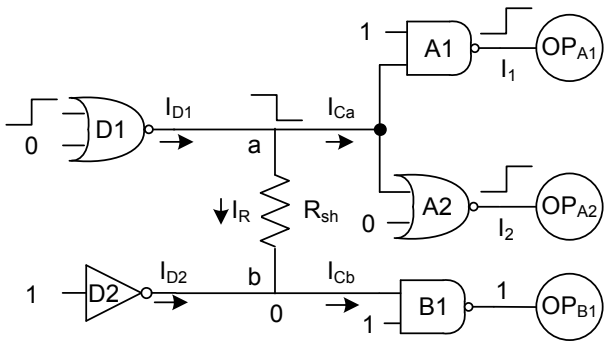
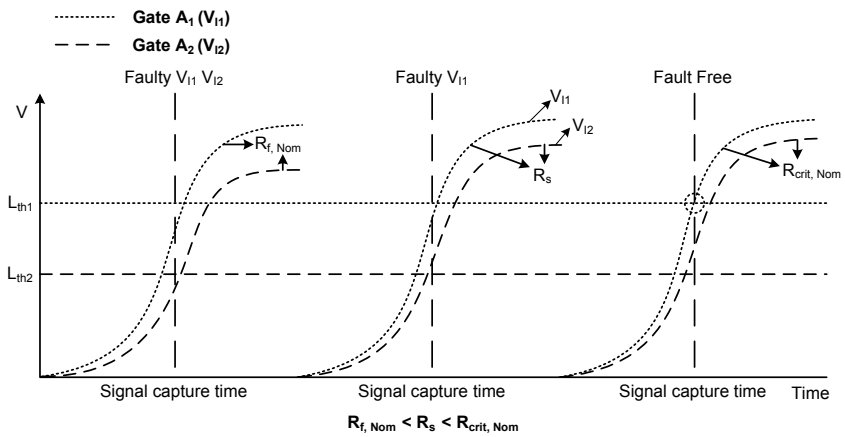


Figure 4.2: Resistive bridge "R_{sh}" forming a potential circuit fault.

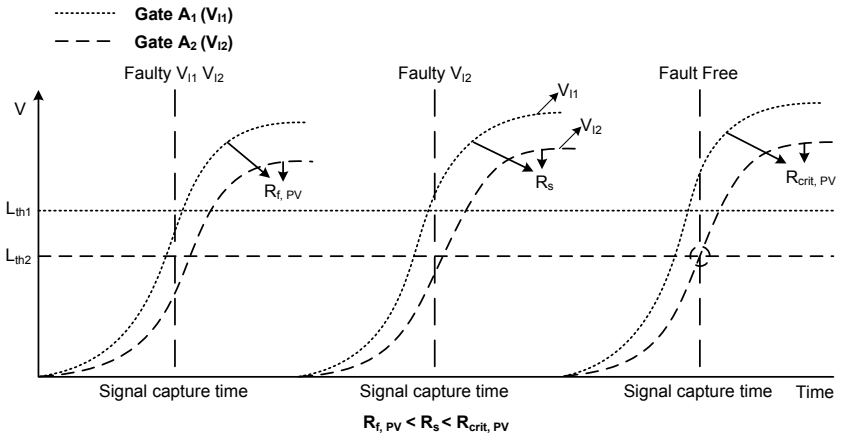
Class-I is shown in Figure 4.1(a), as can be seen there is only 1 transition at faulty node (node a) while the other node (node b) is held at a constant value, and the transition fault is detected through the transition delay at the output of driven gate A_1, or A_2, or A_3. Class-II is shown in Figure 4.1(b), which is due to two transitions at faulty nodes (both node a and b) and the transition fault is detected as in case of Class-I. Class-III test is shown in Figure 4.1(c), which is due to transition at the input of the driven gate (gate A_1) connected to the faulty node (node a) and the transition fault is detected at the output of the same gate (gate A_1). This work analyses the effect of these 3 types of delay test to determine the most suitable type for detecting resistive bridge faults in nominal operating conditions and under the influence of process variation.

Class-I transition delay test shown in Figure 4.1(a) is redrawn in Figure 4.2 to explain how delay faults are computed for a given bridge fault-site. Figure 4.2 shows a resistive bridge forming a potential bridge fault-site, where D_1 and D_2 are the gates driving the bridged lines a and b, R_{sh} is the resistive bridge, and A_1, A_2 and B_1 are the driven gates. The transition signals are applied to the inputs of driving gates and fault effect is observed at the observation points OP_{A1}, OP_{A2} and OP_{B1}. Bridge resistance (R_{sh}) is a continuous parameter which is not known in advance. For each value of bridge resistance $R_{sh} \in [0, \infty)$, the logic values read by observation points OP_{A1} and OP_{A2} can be determined by comparing the transition signal (V_{I1} and V_{I2}) with corresponding logic threshold voltages (L_{th1} and L_{th2}) at signal capture time. This is used to determine if a certain value of bridge resistance leads to a delay fault. The L_{th} voltage of a gate input is the input voltage at which the output voltage reaches half of the supply voltage, while other gate input(s) are at non-controlling value(s) [145].

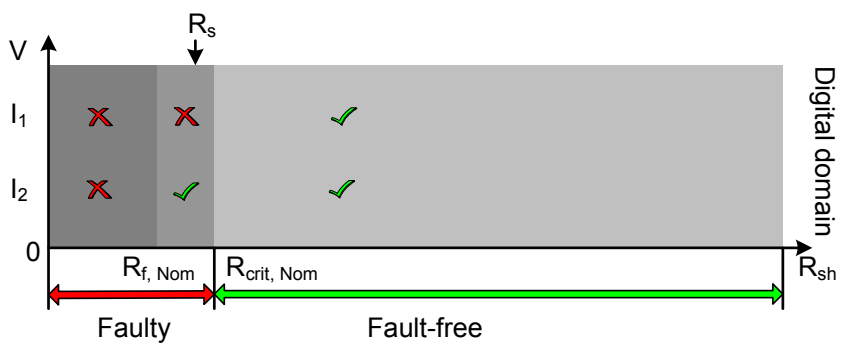
Figure 4.3(a) shows the transition delay behaviour of a resistive bridge (R_{sh}) in analog domain, when operating in a nominal condition. The vertical line represents the signal capture time, which is needed to compute delay faults. The horizontal lines L_{th1} and L_{th2} are logic threshold voltages of OP_{A1} and OP_{A2} respectively. The rising transition signals (V_{I1} and V_{I2}) are transient gate output voltages of gates A_1 and A_2 and input to OP_{A1} and OP_{A2} respectively. These rising transition signals are considered as fault-free, if for example, V_{I1} crosses L_{th1} before the signal capture time, which means that the logic



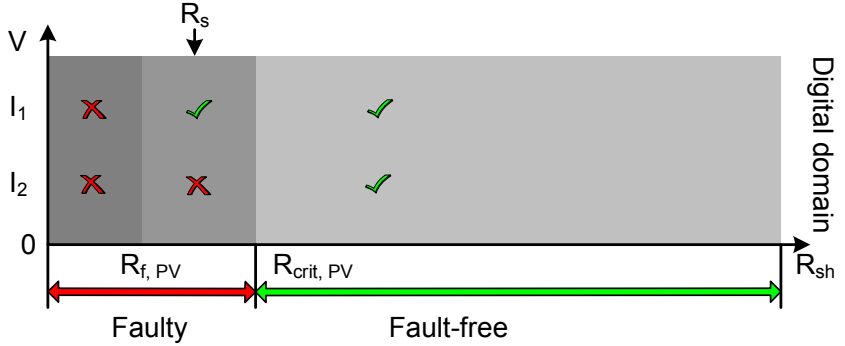
(a) Nominal Condition, Analog Domain: Transition delay behaviour of resistive bridge R_{sh}.



(b) Process Variation, Analog Domain: Change in delay behaviour of resistive bridge R_{sh} due to change in V_{I1}, V_{I2} and logic threshold voltages of driven gates' (L_{th1} and L_{th2}), when compared to Figure 4.3(a).



(c) Nominal Condition, Digital Domain: Delay faults showing timing critical resistance computed through V_{I1}, signal capture time and logic threshold voltages of driven gates' (L_{th1} and L_{th2}).



(d) Process Variation, Digital Domain: Additional delay fault and timing critical resistance, when compared to Figure 4.3(c) leading to increase in fault domain.

Figure 4.3: Transition delay behaviour of resistive bridge.

value (logic-1) is correctly observed at OP_{A1}. This behaviour is shown in Figure 4.3(a), marked as "fault-free". On the other hand, rising transition signals are considered faulty, if for example, V_{I1} crosses L_{th1} after the signal capture time, which means that the logic value (logic-0), observed at OP_{A1} is incorrect (Figure 4.3(a), marked as "Faulty I_1" and "Faulty I_1, I_2"). For a given signal capture time, the timing critical resistance (R_{crit}) of a bridge fault is defined as the resistance value of R_{sh} that corresponds to the crossing point between faulty and fault-free circuit behaviour. The timing critical resistance is the critical resistance that depends on required timing (signal capture time) [142]. Therefore, for a given signal capture time, the circuit behaviour is assumed to be fault free, for all values of $R_{sh} > R_{crit}$.

Using signal capture time (test frequency), the aim of delay fault simulation is to identify unique delay faults corresponding to $R_{sh} \in [0, \infty)$, and maximum detectable resistance of a given fault-site is referred as timing critical resistance (R_{crit}). This is demonstrated by the fault behaviour shown in Figure 4.3(a) (simulated through HSPICE using 65-nm STMicroelectronics library), when operating in nominal "Nom" conditions. As can be seen, depending on the value of R_{sh}, three bridge resistance intervals can be observed. To compute delay faults corresponding to each resistance interval, Figure 4.3(c), is used to convert analog fault behaviour (Figure 4.3(a)) to digital domain. Crosses are used to mark faulty logic values and ticks to mark the correct ones. Bridge delay fault with $R_{sh} \in [0, R_{f,Nom}]$, all observation points (OP_{A1} and OP_{A2}) read faulty logic value. This is because, $V_{I1} < L_{th1}$, and $V_{I2} < L_{th2}$, at signal capture time (Figure 4.3(a)). In the second interval, for $R_{sh} \in [R_{f,Nom}, R_{crit,Nom}]$, OP_{A1} reads faulty value, while OP_{A2} reads correct value ($V_{I2} > L_{th2}$). Finally, $R_{sh} \in [R_{crit,Nom}, \infty)$ leads to all inputs reading correct logic values. Consequently, each interval $[R_a, R_{a+1}]$ corresponds to a distinct logic behaviour occurring at the bridge fault site, referred as delay fault. The R_{sh} value corresponding to $R_{crit,Nom}$ is the timing critical resistance (R_{crit}) of a resistive bridge fault-site, which is detected through gate A_1 and observed at OP_{A1} (Figure 4.3(a)). SPICE Simulation results of delay measurement by sweeping the value of bridge resistance form 0Ω to 1MΩ based on a 65-nm gate library with 1500 bridge fault-sites indicate that in general delay reduces significantly with the increase of bridge resistance, which is shown Figure 4.4. Therefore, different values of R_{sh} have the following relationship: $R_{f,Nom} < R_S < R_{crit,Nom}$.

Due to process variation, the behaviour of resistive bridge fault deviates from that in nominal conditions as shown in Figure 4.3(a). Process variation affects transistor parameters leading to change of gate logic threshold voltage(s) (L_{th}), drive strength(s) (I_{ds}), and output voltage (V_{ds}), as observed in [2, 143]. This change in L_{th}, I_{ds}, and V_{ds} is explained next, which introduces additional delay faults. Figure 4.3(b) shows the change in output transition signals (V_{I1}, V_{I2}) and logic threshold voltages (L_{th1}, L_{th2}) from Figure 4.3(a) due to process variation. Figure 4.3(d) shows corresponding delay faults. From Figure 4.3(b), it can be seen that due to process variation, transition

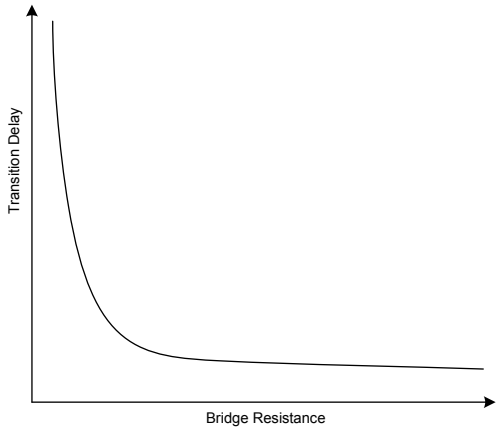
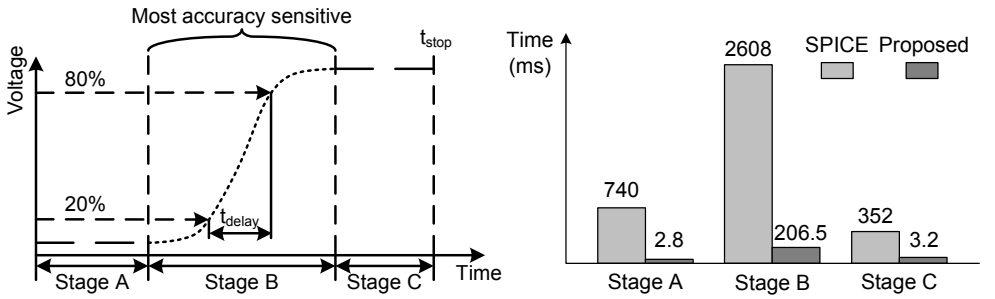


Figure 4.4: Bridge resistance vs. delay.

signals and logic threshold voltages have slightly changed. Because of that change, an additional delay fault is introduced, because maximum resistance is now detectable through I_2 instead of I_1 as shown in Figure 4.3(a). From test generation (ATPG) point of view, a test generated to propagate the fault effect through I_1 (observed at OP_{A1} as in the case of nominal scenario) may lead to a test escape, due to missing a resistance interval $R_{sh} \in [R_{f,PV}, R_{crit,PV}]$ ("PV" is process variation). This is because test through I_1 can only detect resistance value from $[0, R_{f,PV}]$ and an additional test is needed to propagate the fault effect through I_2 to cover additional fault. In case of logic test, it was shown in [2] that on average these additional faults can lead to 10% loss of fault coverage.

4.3 Accelerated Delay Fault Simulation Methodology

Figure 4.3 identifies three most influential variables for computing delay faults of a given bridge fault-site. They are transient gate output voltage (V_{I1}, V_{I2}), logic threshold voltage (L_{th1}, L_{th2}) and signal capture time. This observation is valid for both nominal operating conditions and under process variation. Signal capture time is an input variable to the delay fault simulator. Logic threshold voltage (L_{th}) of a gate can be calculated using SPICE, however it was found that this is a time consuming process. A fast logic threshold voltage calculation algorithm is proposed in Section 3.3.1, Chapter 3, which is on average 257 times faster with 3.1% accuracy deviation, when compared with SPICE. The proposed methodology in this chapter used the same logic threshold voltage calculation algorithm discussed in Section 3.3.1, Chapter 3 to compute delay faults of a given bridge fault-site. Delay faults are computed using an efficient methodology to calculate transient gate output voltage, which is described in Section 4.3.1. As shown in Figure 4.3, delay faults can be represented in term of the bridge resistance intervals $[R_a, R_{a+1}]$, where the timing critical resistance (R_{crit}) represents the maximum resistance coverage of a bridge fault-site. Section 4.3.2 presents the calculation methods of



(a) Transient signal

(b) Relative Speedup: SPICE vs Proposed Methodology

Figure 4.5: Transient signal analysis.

the timing critical resistance by using an interpolation method. Finally, Section 4.3.3 show how the effect of process variation is incorporated into the proposed methodology.

4.3.1 Transient Gate Output Voltage Calculation

The proposed methodology is as accurate as SPICE because it employs the BSIM4 transistor model for calculating transistor parameters, which are used by SPICE to compute transient gate output voltage. The BSIM4 transistor model is valid across all operating (active and saturation) regions. It accurately relates different electrical parameters with device structure and takes into account various inter-dependencies between different transistor parameters. Therefore it is well-suited to model the effect of process variation [86].

To compute delay faults per fault-site, transient gate output voltages (V_{I1}, V_{I2}) need to be calculated for each fanout gate of a bridge fault-site (Figure 4.2). Figure 4.5(a) shows a typical rising transition signal (V_{I1}, Figure 4.2) generated through transient simulation using SPICE. This can be divided into three parts. Stage-A is to initiate transient simulation, where SPICE iterates over a given fault-site to confirm initial voltages on all nodes. Detailed transient analysis is carried out in Stage-B, which uses a transistor output voltage (V_{I1}, Figure 4.2) approximation algorithm that balances out transistor currents at each circuit node, and then it approximates next transient value of V_{I1}. This iterative process continues at each transient step (marked as a dot in Figure 4.5(a)). Stage-C continues transient simulation until the end of simulation time t_{stop}. These three stages are carefully analysed to achieve a trade-off between speed and accuracy. Clearly, Stage-B is the most accuracy-sensitive and time consuming. To speed up these three stages without compromising accuracy, a three-step strategy is employed to compute transient gate output voltage (V_{I1}, Figure 4.2) that consists of the following: transistor models elimination, transistor electrical parameters elimination and step size adjustment.

Table 4.1: Transistor models used in the proposed methodology

S. No.	Transistor Models	Model Used	Parameter Used
1	Drain Current	√	92%
2	Channel Charge & Subthreshold	√	75%
3	Gate Direct Tunneling Current	√	90%
4	Capacitance	√	80%
5	Threshold Voltage	√	100%
6	Body Current	√	100%
7	Temperature Dependent	√	100%
8	Layout-Dependent Parasitics	√	100%
9	Asymmetric Junction Diode	X	
10	New Material	X	
11	Stress Effect	X	
12	Well Proximity Effect	X	
13	High-Speed RF	X	
14	Noise	X	

4.3.1.1 Transistor Models Elimination

Table 4.1 shows all fourteen transistor models used in BSIM4.7 [86]. All these models are analysed to study their influence on transient gate output voltage calculation. It was found that six out of fourteen models (Table 4.1, marked with "X") have negligible effect on transient gate output voltage calculation, which can be removed to save computation time of all variables within each of six eliminated transistor models. The remaining 8 transistor models were retained (Table 4.1, marked with "√") since they affect the accuracy when calculating transient gate output voltage. To explain why some transistor models have negligible effect on calculation accuracy, consider the contribution of Asymmetric MOS Junction Diode Model to transient gate-output voltage calculation in Stage-B (Figure 4.5(a)), it was found that eliminating this model leads to negligible accuracy deviation when compared with SPICE. It has negligible effect because it is used to model situations when a MOS transistor acts as a diode (gate-to-drain connection). The New Materials Model was removed because it is used to model the effect of non-silicon gate insulator, non-poly-silicon gate and non-silicon channel. The gate library used in this work does not employ such materials, which is why this model was not used. The Stress effect and Well Proximity Effect models adjust transistor threshold voltage and mobility due to band structure modification and change in doping profile, as well as deep well implants can effect devices at mask edge. In this work, these two transistor parameters (Effective Mobility μ_{eff} and threshold voltage V_{th}) are varied by directly using the study reported in [10] to model the effect of process variation, which

is why both these models were replaced and removed. The High-Speed RF and Noise Models are needed to model current and noise at high frequencies in analog RF circuits. Since this work deals with digital logic, therefore both these models were removed.

4.3.1.2 Transistor Electrical Parameter Elimination

The second step to accelerate delay fault simulation involves eliminating unnecessary electrical parameters within the retained eight transistor models (Table 4.1, marked as a "√"). In order to identify the unnecessary parameters from each of the transistor model, this work considered data from 100,000 transient simulations of gate output voltage and eliminated electrical parameters from each model using open-source SPICE (NGSPICE [85]). The contribution of each electrical parameter per transistor model was evaluated by comparing with SPICE in terms of transient gate output voltage. In this step, accuracy of transient gate output voltage is maintained to $\leq 3\%$, as it was emprically found to achieve a good trade-off between speed and accuracy. Overall, when considering each of the eight transistor models (Table 4.1), it is possible to eliminate 30 out of 380 (8%) parameters from Drain Current Model, 20 out of 80 (25%) from Channel Charge and Subthreshold Swing Model, 9 out of 90 (10%) from Gate Direct Tunneling Current Model and 79 out of 394 (20%) from Capacitance Model. The remaining four retained transistor models, marked with "100%" in Table 4.1, use all electrical parameters. This count is from NGSPICE [85] and include temporary variables and overlapped parameters that are shared between different models. For example V_{gsteff}, which is the effective (V_{gs}-V_{th}) used to describe the channel charge densities from sub-threshold to strong inversion [86], is used in Capacitance and Drain Current Models and is counted once in both models (Table 4.1).

The elimination of transistor electrical parameters is explained next. In case of the Drain Current model is used to calculate transistor drain current in both linear and saturation region. Eq. (4.1) shows the drain current model. All electrical parameters have their usual meanings, see [86] for details. This model requires 63 sub-equations and 380 electrical parameters, which are calculated by using device parameters (per transistor) through a transistor model card [21] and BSIM4 transistor model equations [86]. To demonstrate how electrical parameters are eliminated without compromising accuracy, consider V_{ADIBL} shown in Eq. (4.1), which represents Early voltage due to Drain Induced Barrier Lowering. Eq. (4.2) shows the equation for calculating V_{ADIBL}. One of the parameter in Eq. (4.2) is V_{gsteff}. Eq. (4.3) is used to calculate V_{gsteff}. All parameters in Eq. (4.3) are analysed and it is found that $NOFF$ and $VOFFCV$ have default values of 1 and 0 respectively. Furthermore, the analysis of transient gate output voltage including transistor drain current (Eq. (4.1)), V_{ADIBL} (Eq. (4.2)) showed negligible effect without these two parameters. This means that these two parameters can be removed without compromising transient gate output voltage calculation. Therefore Eq. (4.3) can be

rewritten as Eq. (4.4). Note that V_{gsteff} is used not only in Eq. (4.2), but is also needed in a number of other transistor electrical parameters. For example, bulk charge (Q_b), source charge (Q_s), drain charge (Q_d), bulk charge effect (A_{bulk}), saturation voltage (V_{dsat}) and early voltage (V_{ASAT}). Therefore simplifying Eq. (4.3) contributes towards faster calculation of these parameters as well. Note, A_{bulk}, V_{dsat}, V_{ASAT} and V_{ADIBL} are included in the drain current equation (Eq. (4.1)).

$$\begin{aligned} I_{ds} = & \frac{I_{ds0} \cdot NF}{1 + \frac{R_{ds} \cdot I_{ds0}}{V_{dseff}}} \left[1 + \frac{1}{C_{clm}} ln \left(\frac{V_A}{V_{Asat}} \right) \right] \\ & \cdot \left(1 + \frac{V_{ds} - V_{dseff}}{V_{ADIBL}} \right) \cdot \left(1 + \frac{V_{ds} - V_{dseff}}{V_{ADITS}} \right) \\ & \cdot \left(1 + \frac{V_{ds} - V_{dseff}}{V_{ASCBE}} \right) \end{aligned} \tag{4.1}$$

$$\begin{aligned} V_{ADIBL} = & \frac{V_{gsteff} + 2v_t}{\theta_{rout} \left(1 + PDIBLCB \cdot V_{bseff} \right)} \\ & \cdot \left(1 - \frac{A_{bulk} V_{dsat}}{A_{bulk} V_{dsat} + V_{gsteff} + 2v_t} \right) \\ & \cdot \left(1 + PVAG \frac{V_{gsteff}}{E_{sat} L_{eff}} \right) \end{aligned} \tag{4.2}$$

$$V_{gsteff} = NOFF \cdot n \cdot v_t \cdot ln \left[1 + \exp \left(\frac{V_{gse} - V_{th} - VOFFCV}{NOFF \cdot nv_t} \right) \right] \tag{4.3}$$

$$V_{gsteff} = n \cdot v_t \cdot ln \left[1 + \exp \left(\frac{V_{gse} - V_{th}}{nv_t} \right) \right] \tag{4.4}$$

In Channel Charge and Subthreshold Swing Model (Table 4.1), the subthreshold swing n is calculated by using Eq. (4.5), which is needed for calculating V_{gsteff}. By analysing one of the parameters, $Cdsc\_Term$, shown in Eq. (4.5), which is calculated by using Eq. (4.6). All the parameters in Eq. (4.6) are analysed and it is found that $CDSCD$ and $CDSCB$ in Eq. (4.6) both have default value of 0, where $CDSCD$ is the body-bias sensitivity of $CDSC$ and $CDSCB$ is the drain-bias sensitivity of $CDSC$. Removing these two parameters and rewriting Eq. (4.6) as Eq. (4.7), simplified it and does not have any effect on transient gate output voltage calculation. Therefore speedup is achieved by simplifying Eq. (4.6) as Eq. (4.7).

$$n = 1 + NFACTOR \cdot \frac{C_{dep}}{C_{oxe}} + \frac{Cdsc\_Term + CIT}{C_{oxe}} \tag{4.5}$$

$$Cdsc\_Term = (CDSC + CDSCD \cdot V_{ds} + CDSCB \cdot V_{bseff}) \cdot \frac{0.5}{\cosh\left(DVT1\frac{L_{eff}}{l_t}\right) - 1} \tag{4.6}$$

$$Cdsc\_Term = CDSC \cdot \frac{0.5}{\cosh\left(DVT1\frac{L_{eff}}{l_t}\right) - 1} \tag{4.7}$$

In Gate Direct Tunneling Current Model, the gate-to-channel-to-source current I_{gcs} and gate-to-channel-to-drain current I_{gcd} are calculated by using Eq. (4.8) and Eq. (4.9) respectively. The analysis of transient gate output voltage calculation shows that calculation accuracy is not affected by removing Eq. (4.8) and Eq. (4.9). This is because the values of I_{gcs} and I_{gcd} are very small (1000 times smaller) when compared to the value of drain current I_{ds} (Eq. (4.1)). Therefore removing Eq. (4.8) and Eq. (4.9) also removes parameters that are only used in Eq. (4.8) and Eq. (4.9), for example $PIGCD$, which is the V_{ds} dependence of I_{gcs} and I_{gcd}, and contribute towards faster calculation of transient gate output voltage.

$$I_{gcs} = I_{gc0} \cdot \frac{PIGCD \cdot V_{dseff} + \exp\left(-PIGCD \cdot V_{dseff}\right) - 1 + 1.0\text{e} - 4}{PIGCD^2 \cdot V_{dseff}^2 + 2.0\text{e} - 4} \tag{4.8}$$

$$I_{gcd} = I_{gc0} \cdot \frac{1 - (PIGCD \cdot V_{dseff} + 1) \cdot \exp\left(-PIGCD \cdot V_{dseff}\right) + 1.0\text{e} - 4}{PIGCD^2 \cdot V_{dseff}^2 + 2.0\text{e} - 4} \tag{4.9}$$

In case of Capacitance Model, which has three modes of operation represented by "capMod" [86], where the third mode ($capMod = 2$) is the most accurate mode of operation. Therefore electrical parameters needed only in the other two modes of operation can be safely removed without affecting accuracy. For example, Flat band voltage (Vfbcv) is used only in the first mode ($capMod = 0$). More details of transistor electrical parameter elimination can be found in Appendix D.

4.3.1.3 Step Size Adjustment

SPICE employs convergence algorithms at every time-step of transient simulation, where currents and voltages are equated at each transistor terminal within the fault-site. A wide variety of gates, including simple and compound gates using 65-nm gate library are analysed to overall reduce convergence time spent at all three stages shown in Figure 4.5(a). For this analysis, the input transient signal is kept constant (rising/falling with rise/fall time) and only varied the step size when considering all three stages. As

expected stage-B is the most sensitive to step size adjustment. The acceptable margin is set to achieve $\leqslant 3\%$ accuracy when compared with Synopsys SPICE, which is a trade-off[1] point between speed and accuracy. In case of Stage-A and Stage-C, which are least sensitive to step sizes, only 2 steps are used for each stage. In case of Stage-A further improvement (about 4x) in computation time is achieved by directly applying initial conditions on all fault-sites nodes instead of iterating over initial conditions as in case of SPICE (Figure 4.5(a)). See [146, 147] for more details on initial condition approximation and convergence algorithms employed by SPICE. As an example, Figure 4.6 shows that SPICE employs convergence algorithms at transient time-step calculation. This transient gate output voltage is calculated through the retained transistor sub-models, which uses $V = \frac{Q}{C}$ relationship to calculate the rate of change of transistor terminal charges Q_g, Q_b, Q_s, and Q_d that are associated with the transistor gate, bulk, source, and drain terminals to determine time-dependent transistor terminal voltages V_g, V_b, V_s, and V_d. The change of transistor current in the BSIM4 model is calculated through Eq. 3.1. Detailed steps for calculating transistor output voltage (V_{I1}; Figure 4.2) are shown in Figure 4.6. It consists of the following 12 steps.

Firstly, operating points of the driving gate (Figure 4.2) are initialised by setting the value of V_{dd} and V_{in}. In step-2, transistor voltages (V_{ds}, V_{gs}) of each transistor in the driving gates (D_1 and D_2, Figure 4.2) are calculated. This is followed by step-3, which is used to calculate transistor currents in the pull-up and pull-down networks (I_p and I_n) of each transistor in the driving gates (D_1 and D_2) by using I_{ds} (Eq. 3.1) and leakage current equations (BSIM4.7 Manual [86]). Next, in step-4 the output current I_{D1}, I_{D2} and the output voltage V_a, V_b of the driving gates, which is also the voltages on the bridge nets are calculated, note where $V_{ds,n,D1}$ is the drain-to-source voltage of the pull-down network in driving gates D_1, $I_{p,D1}$ is the pull-up network current of driving gates D_1. By applying Kirchhoff's Current Law, which is the sum of currents in and out of a node is zero, it shows that I_{D1} - I_R - $I_{Ca} = 0$ for node a and I_{D2} + I_R - $I_{Cb} = 0$ for node b. I_{Ca} and I_{Cb} is the current that goes through the load capacitances C_a and C_b that depends on the driven gates (A1, A2 and B1). Because $I_R = (V_a+V_b)/R_{sh}$, $I_{Ca} = C_a\frac{dV_a}{dt}$ and $I_{Cb} = C_a\frac{dV_b}{dt}$, Eq. 4.10 can be derived. Here R_{sh} is the unknown variable of bridge resistance value, which is swept from 0Ω to a value that the circuit behaves as fault-free. This is described in Section 4.3.2. Step-5 uses the values of I_{D1}, I_{D2}, V_a and V_b to check the solution of Eq. 4.10 has converged within the error accuracy $\leqslant 3\%$. If not, the voltage values are adjusted in step-6 and the flow goes back to step-2. Otherwise in step-7, timing step size is dynamically adjusted per iteration. This is because, as shown in Figure 4.5(a), stage-B is the most accuracy sensitive stage, while timing step size can be dynamically adjusted in other two stages without affecting accuracy. To change timing information, time step Δt is adjusted dynamically by observing the slope of transition signal. Therefore, next timing point $t(n+1)$ is calculated by adding time step $\Delta t(n+1)$ to the current timing point $t(n)$. Step-8 is used to check if $t(n)$ has reached the

[1]Similar to the discussion of Table 3.1 in Chapter 3.

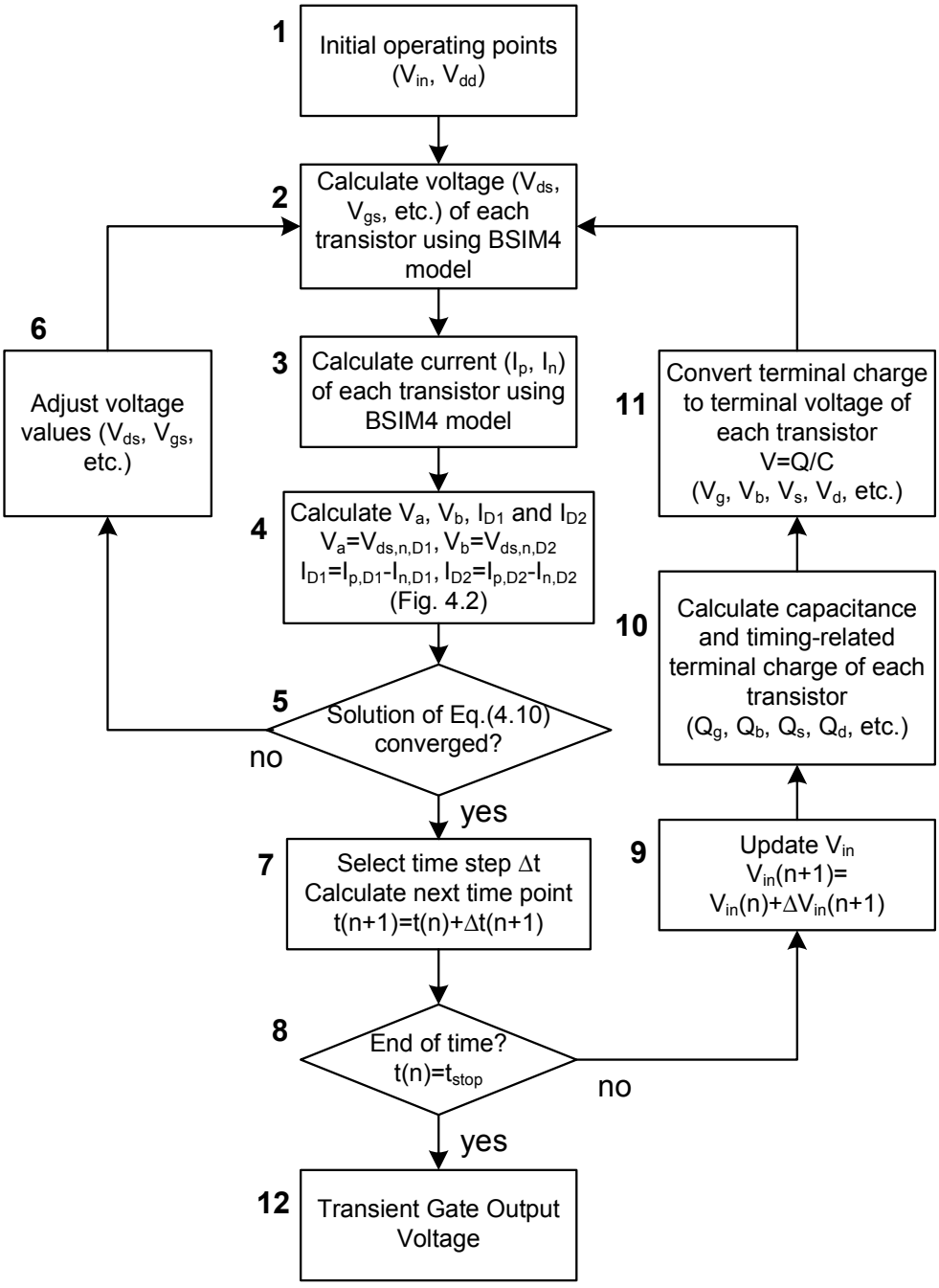


Figure 4.6: Simulation flow for transient gate output voltage calculation.

Table 4.2: Accuracy and speed breakdown of the methodology.

	Error (%)	Speedup
Models elimination (Step-1)	0.8	2.33
Parameters elimination (Step-1 and Step-2)	2.1	4.89
Step size adjustment (Step-3)	2.7	3.38
Combined effect	5.2	17.41

end of simulation time t_{stop}. If not, in step-9, V_{in}(n+1) is calculated by adding the next ΔV_{in}(n+1) to V_{in}(n). Note that ΔV_{in} changes with Δt. After updating the value of V_{in}, step-10 is used to calculate the terminal charges (Q_g, Q_b, Q_s, and Q_d) of each transistor by using the BSIM4 capacitance model with NGSPICE (open-source SPICE engine) approximation method [85]. Next these terminal charges are used to calculate terminal voltages (V_g, V_d, V_s, and V_d) of each transistor. Note after recalculating terminal voltages, the simulation flow goes back to step-2 and V_{ds} is re-calculated together with other transistor voltages. This flow continues until the current simulation time $t(n)$ is equal to the end of simulation time t_{stop}. The output transition voltages V_a (Figure 4.2) is used as the input transition voltage to the driven gates A1 to calculate V_{I1}.

$$\begin{cases} I_{D1} - \frac{V_a - V_b}{R_{sh}} - C_a \frac{\mathrm{d}V_a}{\mathrm{d}t} = 0 \\ I_{D2} + \frac{V_a - V_b}{R_{sh}} - C_b \frac{\mathrm{d}V_b}{\mathrm{d}t} = 0 \end{cases} \tag{4.10}$$

Figure 4.5(b) shows the relative speedup achieved by the proposed methodology when compared with SPICE in each of the three transient signal stages (Figure 4.5(a)). For this comparison, 1500 fault sites from 65-nm gate library were used. The results reported are, on average over 1500 fault sites when operating in nominal conditions. As can be seen, a significant speed up in each of the three stages shown in Figure 4.5(b) is possible through the proposed methodology. Table 4.2 shows the average contribution of each individual step of the proposed methodology on speed and accuracy. These results are based on 100 fault-sites with 600 permutations of process variation. It can be seen that each of the three steps improve fault-simulation time and the combined effect (last row) leads to on average 17.4 times improvement in comparison with SPICE.

4.3.2 Timing Critical Resistance Calculation

Figure 4.3(a) and Figure 4.3(c) show that for each fault-site, timing critical resistance (R_{crit}) distinguishes between faulty and non-faulty behaviour. R_{crit} is an unknown quantity, which is not known in advance. This section explains how accelerated transition gate output voltage calculation (Sec. 4.3.1) is used to calculate R_{crit}. The bridge resistance is swept from 0Ω to a value that the circuit behaves as fault-free, with a step size of 500Ω. To explain how R_{crit} of a fault-site (Figure 4.2) is calculated, the output rising

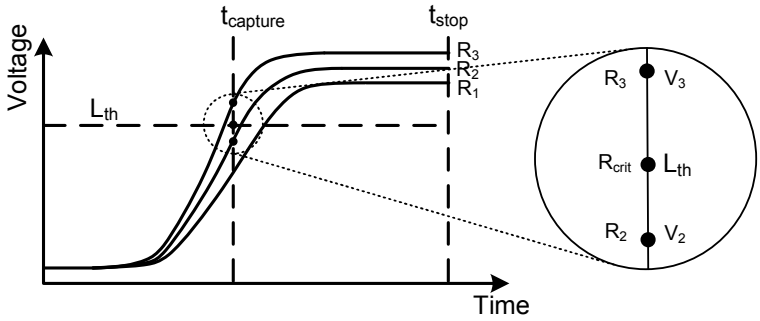


Figure 4.7: Bridge critical resistance calculation when considering delay test.

transition signal of gate A_1 shown in Figure 4.3(a) is re-drawn in Figure 4.7. When sweeping the value of bridge resistance R_{sh} (Figure 4.2) from R_1 to R_3 with a step size of 500Ω. It can be seen that the circuit behaves as faulty until $R_{sh} <= R_2$, however at $R_{sh} = R_3$, the circuit behaves as fault-free. The crossing point is enlarged for clarity. The timing critical resistance (R_{crit}) lies between R_2 and R_3. By applying a simple linear interpolation method, Eq. (4.11) can be obtained. Eq. (4.12) can be used to calculate timing critical resistance of a bridge fault-site which is derived by rewriting Eq. (4.11).

$$\frac{L_{th} - V_2}{R_{crit} - R_2} = \frac{V_3 - V_2}{R_3 - R_2} \tag{4.11}$$

$$R_{crit} = (L_{th} - V_2)\frac{R_3 - R_2}{V_3 - V_2} + R_2 \tag{4.12}$$

4.3.3 Incorporation of Process Variation

The effect of process variation is incorporated by using three un-correlated transistor parameters fluctuations which are recognised as the leading sources of process variation. These three transistor parameters include: gate length (L), threshold voltage (V_{th}), and mobility (μ_{eff}), which are discussed in Section 3.3.3, Chapter 3 already. These parameters follow Gaussian distribution ($\pm 3\sigma$ variation) with standard deviations of 4% for L, 5% for V_{th} and 21% for μ_{eff}. The mean and standard deviation of both NMOS and PMOS transistors of 65-nm technology are shown in Table 3.2. This chapter doesn't consider the effect of within-die variation, this is because of results shown in Section 3.4.4.2, Chapter 3 that show $\pm 3\sigma$ variation of L, V_{th} and μ_{eff} has a much wider spread of critical resistance than the gate length spatial correlation. That means the effect of spatial correlation is covered by considering variations due to three un-correlated parameters (L, V_{th} and μ_{eff}). From test generation point of view, it means that considering three un-correlated parameter variations are likely to cover the complete fault domain due to within-die spatially correlated parameter variation.

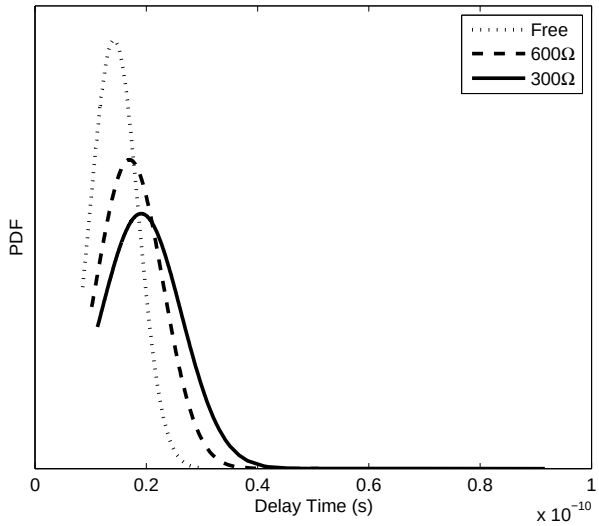


Figure 4.8: The effect of bridge resistance on delay behaviour under the influence of process variation.

Figure 4.8 uses the probability density function (PDF) of normal distribution to show the delay behaviour of a fault-site shown in Figure 4.2 under the influence of process variation by varying three parameters (L, V_{th} and μ_{eff}) using Gaussian distribution with $\pm 3\sigma$ variation. Recent research has shown that it is sufficient to consider $\pm 3\sigma$ variation of process parameters, when modelling process variation for logical part of the design, and higher variation effects ($\pm 6\sigma$ or more) are considered for (SRAM and Flash) memories [12]. For the example shown in Figure 4.8, two resistive bridge defects (R_{sh} is 300Ω and 600Ω) are inserted and the behaviour is compared with fault-free case. In Figure 4.8 the line $free$ represents the transition delay distribution of the fault-free case and the lines marked with 600Ω and 300Ω represent the transition delay obtained from faulty cases under the influence of process variation. It can be seen, as the value of resistance increases, the difference of transition delay in faulty and fault-free cases reduce and further higher values of bridge resistance ($R_{sh} > 600\Omega$) behave like a fault-free case as shown in Figure 4.7. This trend is found for all three classes (Figure 4.1) of transition delay faults.

4.4 Simulation Results

This section presents simulation results using the proposed accelerated delay fault simulation methodology outlined in Section 4.3. A bridge delay fault simulation flow using the proposed methodology is introduced in Section 4.4.1. Section 4.4.2 validates the methodology by comparing the transition gate output delay across a single gate with SPICE. Section 4.4.3 validates the complete simulation flow by using bridge fault-sites. Finally Section 4.4.4 uses the simulation flow to determine the most effective class of transition delay test for testing resistive bridge both in nominal operating conditions

and under process variation. It also compares the results between logic test at lower V_{dd} setting and delay test at nominal V_{dd} setting under the influence of process variation.

4.4.1 Fault Simulation Flow

Simulations are conducted using a 65-nm ST Microelectronics gate library[2] and PTM transistor model card[3] [21] on Intel Xeon Quad Core 2.7 GHz processor with 12 GB RAM. The gate library consists of a variety of gates including simple (NAND, NOR, INV) and compound gates (AO22, OA22 etc.), each with different drive strengths. For illustration purposes 1.2 V is used as the nominal operating voltage in all simulations. Figure 4.9 shows the proposed transition delay fault simulation flow for resistive bridges, when considering process variation. The flow inputs are gate library (65-nm ST Microelectronics) and respective transistor models (PTM model) and the output is transition delay faults (Figure 4.3) of a given fault-site. For each fault-site, 600 permutations are generated through Monte-Carlo simulation. The number of permutations are based on a recent study, which shows that the probability of generating a unique logic fault follows the law of diminishing returns, as it reduces significantly after 500 permutations [2].

The simulation flow shown in Figure 4.9 has seven main blocks. The effect of process variation is incorporated by the process variation permutation generator. It varies three parameters: gate length (L), threshold voltage (V_{th}), and mobility (μ_{eff}). These parameters follow Gaussian distribution ($\pm 3\sigma$ variation) with standard deviations of 4% for L, 5% for V_{th} and 21% for μ_{eff}. The fault-site generator is used to build bridge fault-site for each of the three classes (Class-I, Class-II and Class-III), which is generated by randomly selecting gates from the gate library, using n driven gates per fault-site, where $n \in [1, 5]$. In this work, 1500 fault-sites are used because beyond 1500 fault-sites, the time consumed by SPICE (needed for comparison) becomes prohibitively long as shown in Section 4.4.3. Each fault-site is tested by using an exhaustive transition test for fault propagation to the output of the fault-site, which means that every possible input vector is applied to the fault-site for three different classes. For example, in case of Class-I delay test shown in Figure 4.1(a), the transition signal is applied to every input of D_1 and D_2. The output of these three blocks shown in Figure 4.9 are fed to the transient gate output voltage calculator (Sec. 4.3.1) that uses BSIM4 transistor model and an open-source SPICE engine (NGSPICE [85]) to calculate transient gate output voltage (Figure 4.5(a)) of each fanout gate to generate transition delay faults (Figure 4.3). For illustration, the signal capture time used in this work is an output transition signal at 80% of V_{dd} for rising transition and at 20% of V_{dd} for falling transition ($t_{capture}$, Figure 4.7) in fault-free designs. For each fault-site, bridge resistance is swept with a step size of 500Ω, starting from 0Ω to a value such that the circuit behaves as fault-free. The delay faults generated from transient gate output voltage calculator are fed to the

[2] Appendix B shows SPICE description of three gates from the gate library.
[3] Appendix C shows SPICE description of the PTM transistor model card.

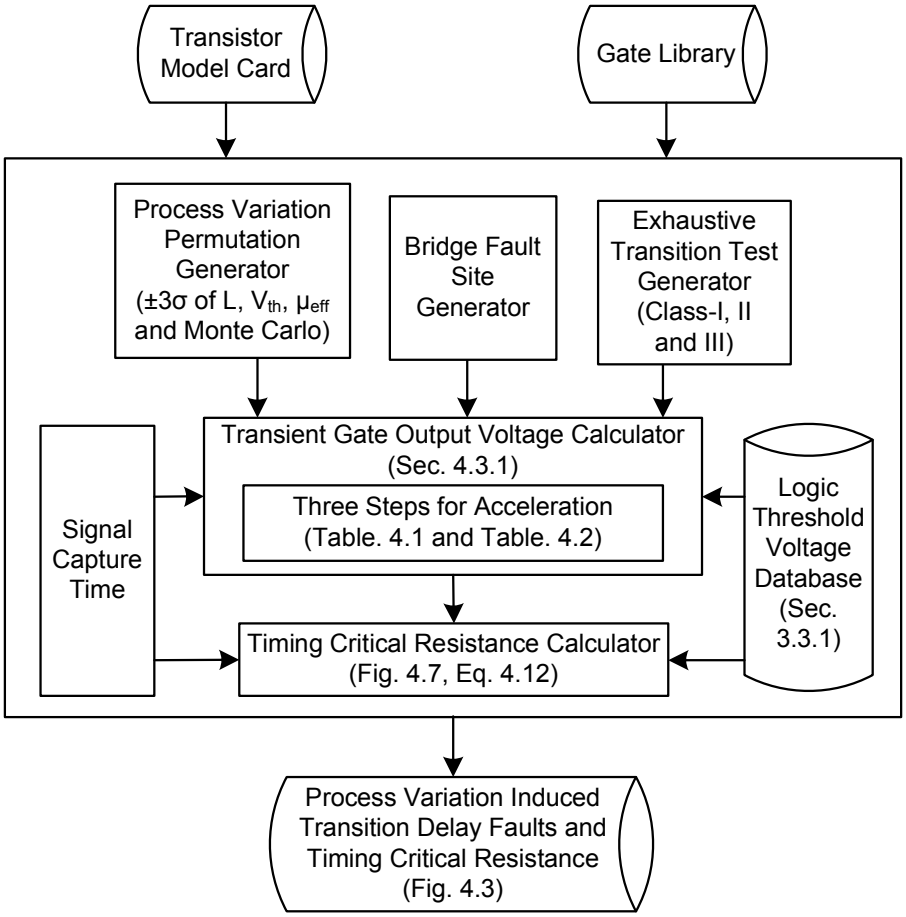


Figure 4.9: Accelerated delay fault simulation flow.

Table 4.3: Gate delay comparison: SPICE vs. proposed.

Gate	Delay (PS)				Worst case Error (%)	Speedup ($\frac{SPICE}{Proposed}$)
	SPICE		Proposed			
	Min	Max	Min	Max		
INV	3.8	14.7	3.7	14.5	2.6	39x
NAND4	4.1	17.4	4.2	17.8	3.1	31x
NOR4	3.9	20.5	4.0	19.9	3.2	29x
XOR3	4.9	18.1	4.8	18.6	**3.3**	28x
AND4	3.8	17.5	3.9	17.1	3.1	31x
OR4	3.8	19.6	3.9	20.1	3.2	29x

timing critical resistance calculator, that uses a linear interpolation method (Figure 4.7 and Eq. (4.12)) to calculate timing critical resistance. Using logic threshold voltage calculation algorithm proposed in Section 3.3.1, Chapter 3, a pre-computed logic threshold voltage database of the complete gate-library is employed, with both SPICE and the proposed methodology, to show relative speedup of the proposed methodology in comparison with SPICE. This setup is used to conduct three sets of simulations, including two sets of simulation validating the proposed simulation methodology and analysis of three transition delay test classes.

The fault simulation flow shown in Figure 4.9 has been implemented as a prototype software tool based on C/C++. The software tool analyses the gate structure (transistor in series or in parallel) from given gate library and read the parameter values from the gate library and transistor model card. This information (gate structure and parameter values) are used in the seven main blocks of the proposed simulation flow shown in Figure 4.9. Process variation permutation generator, bridge fault site generator and logic threshold voltage database have been discussed in Section 3.4, Chapter 3. The signal capture time is an input to the software tool. Exhaustive transition test generator is used with each gate of the gate library to generate test vector for three different classes (Class-I, II and III). The transient gate output voltage calculator is implemented based on transistor models elimination, transistor electrical parameter elimination and step size adjustment to modify the BSIM4 code [86] and NGSPICE code [85]. The transient simulation algorithm in NGSPICE is used as the basis for developing the program that is incorporated in the fault simulation flow shown in Figure 4.9. A Timing critical resistance calculator is built by using the interpolation method (Eq. (4.12) in Section 4.3.2).

4.4.2 Gate Output Transition Delay

Table 4.3 shows gate delay results to compare speed improvement and accuracy deviation of the proposed simulation methodology when compared with SPICE. For each

gate, the delay is calculated by taking the time difference between 20% to 80% of V_{dd} (t_{delay}, Figure 4.5(a)), as described in 65-nm STMicroelectronics gate library manual. To mimic Class-I test, single transition signal was applied to each gate input, while keeping all other inputs to non-controlling logic values. The first column in Table 4.3 shows a representative set of gates from 65-nm gate library that show highest calculation error in comparison with SPICE. For both methods, column two shows the minimum ("Min") and the maximum ("Max") gate delay "*Delay (PS)*" when considering 600 permutations of process variation. For each gate, third column shows the worst case deviation from SPICE results. The maximum (3.3%) error is obseved in case of XOR3 (3-input XOR) gate. The last column shows simulation runtime improvement ($\frac{SPICE}{Proposed}$) on average over all permutations. For the listed gates, maximum speedup of 39x is achieved in case of INV, while XOR3 shows minimum of 28x improvment. The difference in speedup is due to different number of transistors in INV and XOR3 gates, which require additional time for calculating current and voltage values. When considering all gates in the library, the proposed method is on average 29 times faster with 3.3% worst case deviation in accuracy.

4.4.3 Delay Fault Simulation

Using the accelerated simulation flow shown in Figure 4.9, Table 4.4 shows timing critical resistance results of 1500 fault-sites that exhibit best (B) and worst (W) case accuracy and runtime improvement. The first column in Table 4.4 shows driving gates of each fault-site. The second column shows selected timing critical resistance values from 600 permutations of process variation that show best case (B), or worst case (W) accuracy percentage error and relative speedup ($\frac{\text{SPICE}}{\text{Proposed}}$), when comparing the proposed methodology with SPICE. Note that fault-site permutations exhibiting the best case accuracy (0.9%) and speedup (21.3x) is in case of a fault-site driven by inverters. This is because, it is the simplest gate, requiring minimum number of voltage and current calculation steps. The worst case error (8.8%) is found in case of a fault-site driven by AND4-NOR3 gates which is also shown in Figure 4.10 when considering the effect of process variation, and minimum speedup (11.7x) is in case of a fault-site driven by XOR3-NAND3 gates. This is because of additional number of voltage and current calculation steps that led to these results. The last row of Table 4.4 shows average error (5.2%) and runtime improvement (17.4x) of timing critical resistance calculation over 1500 fault-sites. The runtime improvement on simulation time of 1500 delay fault-sites using 600 permutations is shown in Figure 4.11 using SPICE and the proposed methodology.

Table 4.5 shows delay fault simulation time comparison, when using SPICE and the proposed methodology on Intel Xeon Quad Core 2.7 GHz processor with 12 GB RAM. For comparison, five designs are synthesised using 65-nm STMicroelectronics gate library with Synopsys Design Compiler. The number of bridges are identified through Cadence

Table 4.4: Performance comparison over 1500 fault-sites.

Driving Gates (D1, D2)	R_{crit} (Ω)		Error (%)	Speedup ($\frac{SPICE}{Proposed}$)
	SPICE	Proposed		
INV-INV	4343.6	4304.2	**0.9** (B)	18.7X
AND4-NOR3	14578	15864	**8.8** (W)	16.6x
INV-INV	3865.2	3795.3	1.8	**21.3x** (B)
XOR3-NAND3	11598	12242	5.6	**11.7X** (W)
Average results			5.2	17.4x

* B → Best case W → Worst case

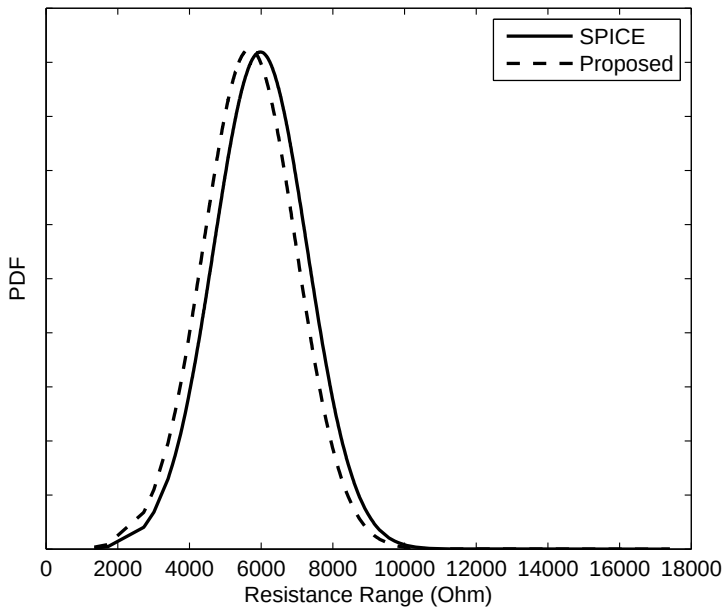


Figure 4.10: The effect of process variation on timing critical resistance of an bridge fault-site driven by AND4 and NOR3 gates.

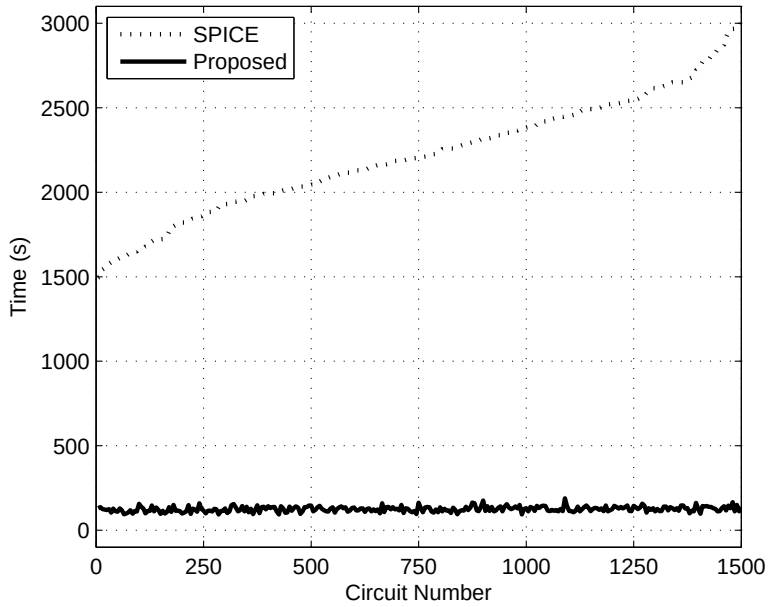


Figure 4.11: Computation time improvement: Proposed simulation methodology vs SPICE.

Table 4.5: Simulation time on SPICE vs. proposed model on Intel Xeon Quad Core 2.7 GHz processor with 12 GB RAM.

CKT No.	# Gates	# Bridges	Time (Days)	
			SPICE	PM
CKT 1	762	581	15.34	0.84
CKT 2	1153	363	11.19	0.62
CKT 3	1621	954	25.21	1.41
CKT 4	3734	1194	32.04	1.85
CKT 5	5251	2965	77.40 (Estimated)	4.65

"extractRC" tool, which is used to get all pair of bridges (fault list) through coupling capacitance based post-layout extraction. For each design, the number of gates and extracted bridges are shown in second and third columns respectively. The last column shows simulation time using SPICE and the proposed methodology. It can be observed that the proposed methodology significantly reduces delay fault simulation time when compared with SPICE. In case of CKT-5 with 5251 gates, and 2965 bridge fault sites, estimated time with SPICE is more than 77 days. It is estimated to avoid prohibitively long computation time. Estimation is made by using average runtime results shown in Table 4.4, over 1500 fault-sites when considering process variation. In comparison, delay fault simulation time by using the proposed methodology is only 4.65 days.

4.4.4 Analysis of Transition Delay Fault Behaviour

Section 4.4.2 and Section 4.4.3 validate the delay fault simulation methodology (Section 4.3), results show that proposed methodology is accurate (5.2% error) and fast (17.4x) when compared with SPICE. This section uses the proposed methodology to analyse the delay fault behaviour of three classes of transition delay test. Section 4.4.4.1 determines the most effective class of delay test for testing resistive bridge when in nominal operating conditions, while Section 4.4.4.2 compares the results under the influence of process variation. Section 4.4.4.3 compares the results of detectable resistance coverage between logic test at lower V_{dd} setting and delay test at nominal V_{dd} setting under the influence of process variation.

4.4.4.1 Delay Faults in Nominal Operating Conditions

For the nominal operating conditions, the simulation is conducted using the flow shown in Figure 4.9 without using process variation permutation generator for 1500 fault-sites using exhaustive test vectors per class. The results are shown in Table 4.6, which shows the timing critical resistance for the fault-site shown in Figure 4.1 using all three classes of delay test. The input vectors are chosen to ensure exhaustive transition tests for each class. The critical bridge resistance of bridge delay fault is calculated using the method shown in Figure 4.7 and Eq. (4.12) in nominal operating conditions, and the critical resistance from logic test is calculated by using the critical resistance calculation algorithm described in Section 3.3.2, Chapter 3. The column *Input* in Table 4.6 shows different input vectors for driving gates D_1 and D_2 (Figure 4.1). $\uparrow$ and $\downarrow$ represent the rising signal and falling signal. Results in Table 4.6 show that in nominal operating conditions, the critical resistance obtained from delay test (maximum of 2916.4Ω) is significantly higher than the one from logic test (399.8Ω). Table 4.6 also shows that using delay test with different input transition signals applied to different inputs (D_1 and D_2) the critical resistance changes significantly, for example, in case of Class-I delay test the value of R_{crit} varies from 901.1Ω to 2916.4Ω. Note that every class has different

Table 4.6: Critical resistance in logic test and delay test in nominal operating conditions.

Class	Input			R_{crit} (Ω)	
	D_1		D_2	Logic Test	Delay Test
I	↑	0	1	399.8	901.1
	↓	0	1		2087.9
	0	↑	1		1419.7
	0	↓	1		2916.4
	0	0	↑		1248.3
	0	0	↓		1598.6
II	↑	0	↓	399.8	897.8
	↓	0	↑		2122.1
	0	↑	↓		1377.8
	0	↓	↑		2277.7
III	0	0	1	399.8	967.6
	0	0	1		2050.9

Table 4.7: Percentage of critical resistance in nominal operating conditions using three classes of delay test.

	Class-I	Class-II	Class-III
Falling delay	45.3%	20.5%	34.2%
Rising delay	48.5%	19.4%	32.1%
Average	46.9%	19.9%	33.2%

maximum resistance coverage with different input vectors (timing critical resistance per class). The maximum resistance is detectable in case of Class-I test, which is 2916.4Ω compared to Class-II of 2277.7Ω and Class-III of 2050.9Ω. In general, when considering 1500 fault-sites with exhaustive test vectors in nominal operations, results are shown in Table 4.7. It shows that Class-I has the largest coverage, and up to 48.5% cases show maximum detectable resistance using Class-I, and on average Class-I has the largest coverage in 46.9% cases while Class-II has the lowest coverage (19.9% cases) when testing resistive bridge defects.

4.4.4.2 Delay Faults under Process Variation

Process variation permutation generator shown in Figure 4.9 is used to model the impact of process variation by considering variation of three un-correlated parameters (L, V_{th} and μ_{eff}). The process variation permutation generator generates 600 permutations of three parameters for each fault-site following Gaussian distribution within the range of

Table 4.8: Critical resistance in logic test and delay test under the influence of process variation.

Class	Input			Logic Test		Delay Test	
	D_1		D_2	Min (Ω)	Max (Ω)	Min (Ω)	Max (Ω)
I	↑	0	1	39.1	2968.7	0	9905
	↓	0	1			0	8645
	0	↑	1			0	7438
	0	↓	1			0	8313
	0	0	↑			0	7215
	0	0	↓			0	6957
II	↑	0	↓	39.1	2968.7	0	9410
	↓	0	↑			0	7825
	0	↑	↓			0	7700
	0	↓	↑			0	8438
III	0	0	1	39.1	2968.7	0	9354
	0	0	1			0	9625

Table 4.9: Percentage of maximum resistance range of each of the three classes of delay test under process variation.

	Class-I	Class-II	Class-III
Falling delay	38.4%	28.4%	33.2%
Rising delay	33.7%	34.2%	32.2%
Average	36.0%	31.3%	32.7%

$\pm 3\sigma$ using Monte-Carlo simulation. For the fault-site shown in Figure 4.1, the results are shown in Table 4.8. The "Min" and "Max" values in Table 4.8 represent the minimum and maximum values of critical resistance, as a result of process variation. It can be seen that in logic test the critical resistance varies from 39.1Ω to 2968.7Ω, but in delay test the critical resistance varies from 0Ω to 9905Ω, which is significantly higher. In this case, Class-I covers highest resistance range in comparison with the other two classes. Figure 4.12 shows the change in critical resistance for three different resistive bridge transition test classes shown in Figure 4.1 and using logic test, under the influence of process variation. It can be seen that Class-I covers maximum detectable resistance range (9905Ω). Next, 1500 fault-sites are used to show the maximum resistance coverage by each class using exhaustive test under the influence of process variation. Results are shown in Table 4.9. It can be seen that on average Class-I has the largest coverage (36.0%) and it is up to 38.4% and Class-II has the lowest (31.3%) when testing resistive bridge under the influence of process variation. These results clearly indicate that Class-I has the highest coverage range among three classes of delay test, and the coverage of

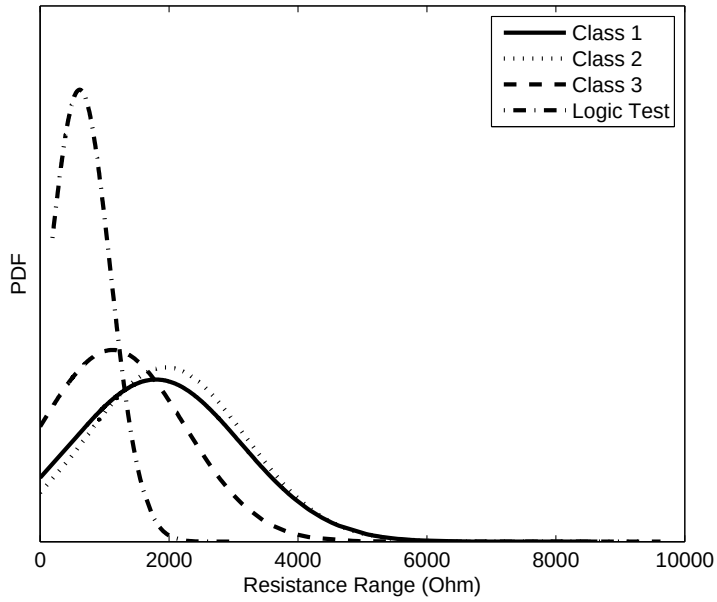


Figure 4.12: Critical resistance range for logic test and different classes of delay test under the influence of process variation.

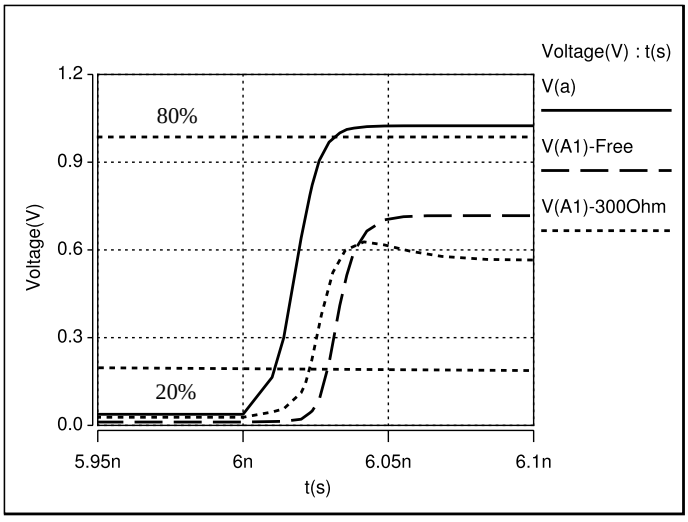


Figure 4.13: Transition signal behaviour due to worst case process variation and resistive bridge defect.

Table 4.10: Average critical resistance of logic test at 0.8 V V_{dd} setting and delay test of Class-I at 1.2 V V_{dd} setting in nominal operating conditions and under process variation.

		Nom	$\pm 3\sigma$ Variation	
Method	Voltage	(Ω)	Min (Ω)	Max (Ω)
Logic Test	0.8 V	1118.8	247.2	5447
Delay Test	1.2 V	2353.3	120.3	10269

each class is always better than logic test.

Next, an example to show the effect of worst case process variation (variation performed at -3σ from the mean) on delay behaviour of a fault-site as shown in Figure 4.1(a), without considering the effect of a resistive bridge (R_{sh} = 0Ω). The results is shown in Figure 4.13, where the design is operating at 1.2 V and a transition single is applied at the input of gate D1 and is observed at the output of gate D1 (node a) and gate A1 (node A1). It can be seen that the voltage V(a) is reduced (slightly above 80% V_{dd}) because of the decreased drive strength of gate D1 due to the worst case process variation. Furthermore, the logic threshold voltage and drive strength of the driven gate (A1) are also affected by worst case process variation leading to faulty behaviour (V(A1) $<$ 80% V_{dd}) as observed at the output of gate A1. Figure 4.13 also shows the delay behaviour of the same fault-site with a resistive bridge (R_{sh} = 300Ω). It can be seen that the delay increases further and the fault observed at the output of gate A1 behaves like a stuck-at fault, which can be detected through both logic and delay tests.

4.4.4.3 Comparison between Delay Test and Logic Test

Results in Section 4.4.4.1 and Section 4.4.4.2 show the resistance coverage using delay test is significantly larger than logic test both in nominal operating conditions and under the influence of process variation while using the same supply voltage (V_{dd} = 1.2 V). Previous research shows that lowering supply voltage setting can achieve higher resistance coverage both in case of logic and delay test [50, 71]. This section compares the results of logic test at lower supply voltage setting (V_{dd} = 0.8 V) while delay test at V_{dd} = 1.2 V using 1500 fault-sites both in nominal operating conditions and under the influence of process variation to compare the resistance coverage. For the fault-site shown in Figure 4.1, results are shown in Figure 4.14 by using the Class-I transition test and low V_{dd} logic test under the influence of process variation. Figure 4.14 shows that delay test at nominal operating voltage covers higher resistance range when compared with logic test at lower supply voltage. Results of average critical resistance from 1500 fault-sites are shown in Table 4.10. The value of "Nom" represents the critical resistance in nominal operating conditions. The values of "Min" and "Max" represent the minimum

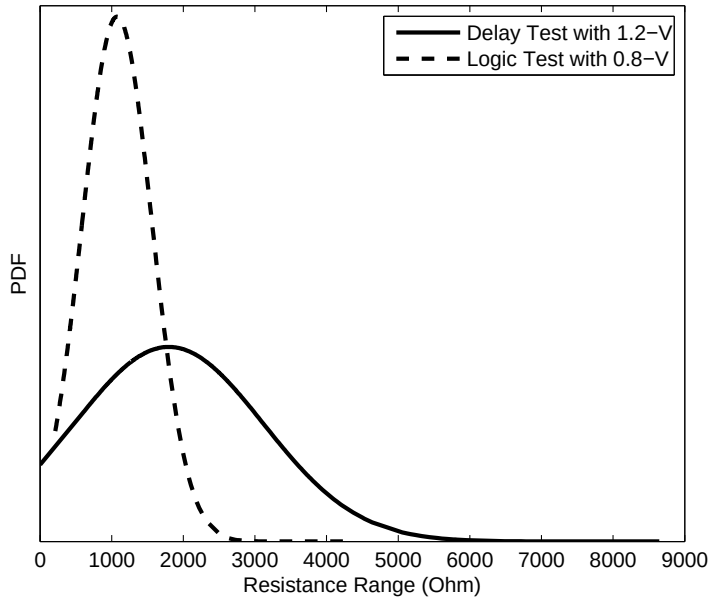


Figure 4.14: Critical resistance range for logic test at V_{dd} = 0.8 V and Class-I transition delay test at V_{dd} = 1.2 V under the influence of process variation.

and maximum critical resistances under the influence of process variation. On average from 1500 fault-sites, Class-I transition delay test has critical resistance value of 2353.3Ω in nominal operating conditions and varies from 120.3Ω to 10,269Ω under the influence of process variation which is significantly higher than the detectable resistance range at lower V_{dd} using logic test.

4.5 Concluding Remarks

This chapter analysed the delay behaviour of resistive bridge defects, and it has shown that their detectability is affected (Figure 4.3, Section 4.2) when considering the influence of process variation. To accelerate fault simulation under process variation, this chapter has presented an accelerated delay fault simulation methodology. The acceleration is achieved by employing a three-step strategy consisting of transistor model elimination, transistor electrical parameters elimination and step size adjustment, when computing transition gate output voltage. The methodology is on average 17.4 times faster with 5.2% error in accuracy, when compared to SPICE (Table 4.4, Section 4.4.3). Using a 65-nm gate library, a design with 3734 gates and 1194 bridge fault-sites, it was found that SPICE requires more than 32 days while the proposed methodology requires less than 2 days to complete delay fault modeling. Using the proposed methodology, the delay behaviour of resistive bridge defects is analysed using three delay test classes (Figure 4.1, Section 4.2). It is found that the resistance coverage achieved by using any class of delay test is significantly higher than logic test and this trend continues even with low-voltage

logic test and nominal voltage delay test. When comparing three different classes of delay test under the influence of process variation, it is found that maximum coverage is possible by using Class-I delay test. This trend continues in nominal operating conditions as well. The simulation methodology shown in Figure 4.9 has been demonstrated using a 65-nm gate library. It can be used with other technology nodes, which will require a gate library with respective transistor model card, and appropriate values of mean and standard deviation of transistor parameters (Table 3.2, Chapter 3).

Chapter 5

Accelerated Delay Fault Simulation of Resistive Open Defects

Resistive open defects are a major class of defects in deep submicron CMOS and as discussed earlier in Section 2.2, Chapter 2, there is a need to develop efficient variation-aware fault modelling and simulation methodology for this class of defects. Fault modelling and simulation using SPICE is the most accurate method of modelling defect behaviour. However, it requires a long computation time to model the delay behaviour (due to process variation) as discussed in Section 5.2. The problem of long computation time is observed in recent publications [2, 79, 80]. This chapter develops an accelerated delay fault simulation methodology for resistive open defects, which consists of two efficient algorithms to address the problem of long simulation time of delay faults. The first algorithm employs the methodology proposed in Section 4.3.1, Chapter 4 to accelerate calculation time of transient gate output voltage that is needed for generating delay faults. The second algorithm uses an approximation method to determine timing critical resistance of an open fault-site. Simulations are carried out using 65-nm and 45-nm gate libraries. It is shown that a synthesised ITC'99 benchmark design with 1465 gates and 1485 open fault-sites requires 24.06 days for delay fault simulation under process variation when using SPICE, while the proposed methodology only requires 0.50 days. Results show that the proposed methodology is on average 52 times faster with 4.43% error in accuracy, when compared with SPICE.

5.1 Introduction

Resistive open defects are due to broken interconnects in a manufactured design, which deviates the circuit behaviour from ideal. Open defects can be classified into full opens

and resistive opens [28]. Full opens exhibit a resistance range $\geq$ 10 MΩ and can be tested using static test (test patterns applied without timing consideration) because they lead to logic failures [127, 148, 149]. On the other hand, resistive opens exhibit resistance range $\leq$ 10 MΩ and are usually targeted by delay test because they show timing-dependent behaviour [28, 56, 149, 150]. When considering the effect of process variation, recent research has focused on modelling, simulation and test generation of deep submicron defects [2, 29, 79, 80, 142]. The main difficulty in simulating additional faults due to increase in size of fault domain, as pointed out in Section 2.3, Chapter 2 is long computation times, when using SPICE to model defect behaviour [2, 79, 80]. By analysing the delay fault simulation time of a 65-nm design with 1465 gates and 1485 open fault-sites in Section 5.4 of this chapter, it was found that SPICE requires 24.06 days to complete delay fault modelling, when using a Quad-Core 2.7 GHz processor with 12 GB RAM. Using these results, delay fault computation time of a slightly bigger netlist (5016 gates with 5773 open fault-sites) is estimated to require more than 93 days to complete fault simulation.

As described in Section 2.3, Chapter 2, there is no reported work on efficient process variation-aware delay fault simulation methodology for resistive opens, which is the aim of this chapter. In this chapter, the delay behaviour of resistive open defects is analysed and it is shown that transient gate output voltage is needed to compute delay fault of resistive opens (as for resistive bridges). Therefore the methodology proposed in Section 4.3.1, Chapter 4 to speed up the calculation of transient gate output voltage is used in this chapter as well to accelerate the computation time of resistive open delay faults. Furthermore, this chapter proposes an efficient algorithm based on the binary search method to reduce the time on calculating the timing critical resistance of an open fault-site. The proposed methodology in this chapter can be used to replace SPICE for efficient fault simulation of resistive open faults. Simulations are conducted using a 65-nm gate library, and results are compared with SPICE (Synopsys HSPICE). Results show that the proposed methodology is on average up to 52 times faster, with 4.2% error in accuracy (worst-case is 6.9%). The proposed methodology can be used with other technology nodes. Similarly, results using a 45-nm gate library show that the proposed methodology is on average 50 times faster that SPICE, with 4.43% error in accuracy.

This chapter is organised as follows: Section 5.2 illustrates the effect of process variation on detectability of resistive open faults and identifies the most influential variables needed to accurately compute delay faults. Section 5.3 presents an accelerated variation-aware delay fault simulation methodology for resistive open defects. Section 5.4 reports fault simulation flow and results, and finally Section 5.5 concludes the chapter.

5.2 Preliminaries

This section analyses the delay behaviour of resistive open faults in nominal operating conditions (referred as “Nom”) and under the influence of process variation (referred as

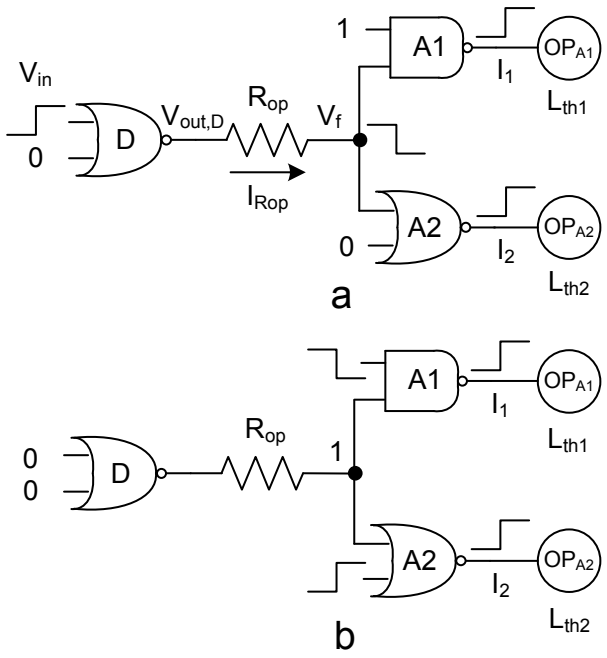
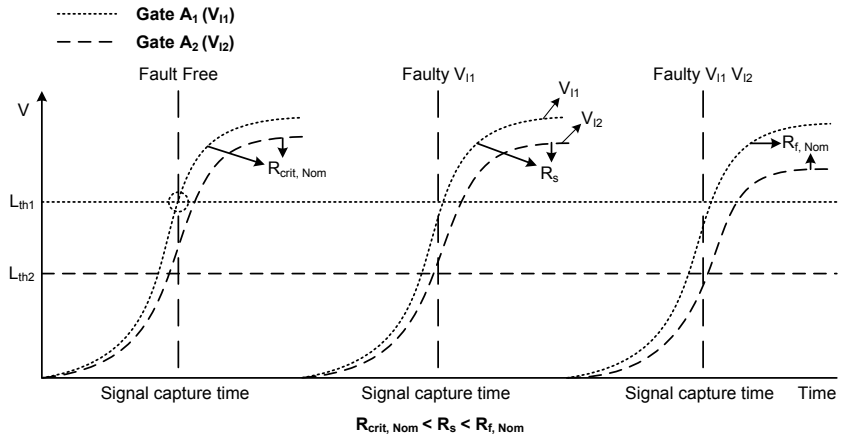


Figure 5.1: Transition delay test classification: (a) Class-I; (b) Class-II.

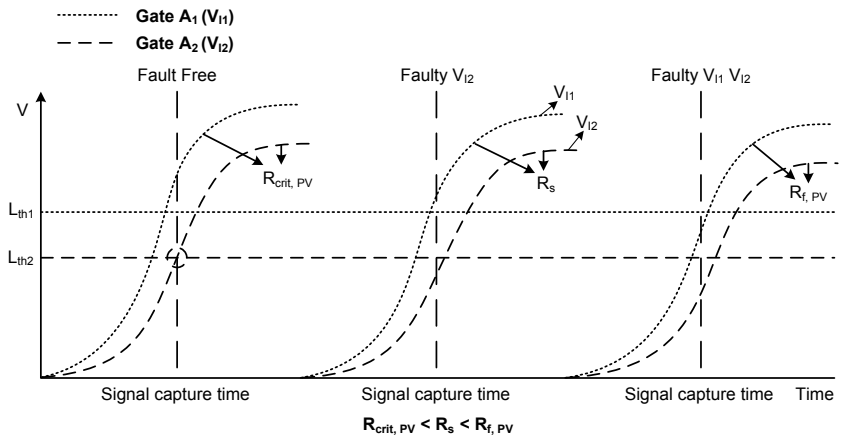
"PV") to demonstrate how detectability of delay test is affected under the influence of process variation and identify the most influential variables for computing delay faults with the aim to reduce their computation time.

The transition delay test of resistive opens can be classified into two classes. These two classes are shown in Figure 5.1, where R_{op} is the resistive open fault, D is the driving gate, A_1 and A_2 are the driven gates. Class-I is shown in Figure 5.1(a), as can be seen, the transition signal is applied to one of the inputs of the driving gate D, and the other input is kept at a constant value. The transition signal goes through the resistive open, and the fault is detected at the output of a driven gate A_1 or A_2. Class-II test is shown in Figure 5.1(b), which is due to signal transition at the input of the driven gates (A_1 or A_2) while the driving gate has stable inputs and the transition fault is detected at the output of the driven gates (A_1 or A_2).

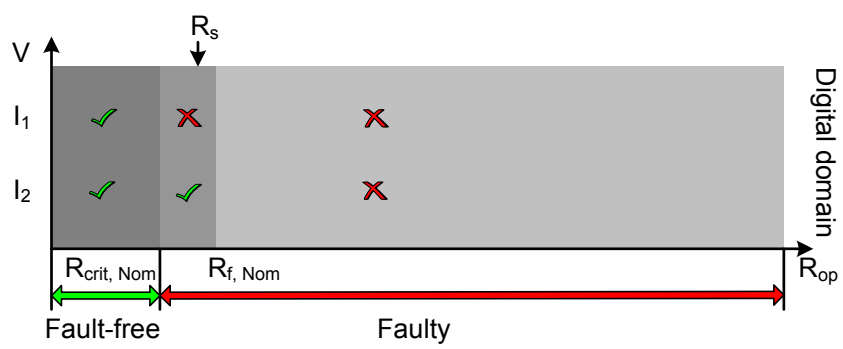
Figure 5.2(a) shows a typical resistive open transition delay fault behaviour in analog domain, when the fault-site shown in Figure 5.1(a) is operating in nominal operating conditions. The vertical dashed line represents the signal capture time. The horizontal lines marked with "L_{th1}" and "L_{th2}" are the logic threshold voltages of the driven gates that are connected to gates A_1 and A_2 respectively. The logic threshold of a gate input is defined as the input voltage at which the output reaches half of the supply voltage, while all other gate inputs are at non-controlling value(s) [50]. The rising transition signals are the gate output voltages of A_1 (dotted line) and A_2 (dashed line). If transition signal V_{I1} at the output of gate A_1 crosses L_{th1} before signal capture time, this means the driven gate connected to A_1 (OP_{A1}) reads correct logic value (logic-1) within signal capture time and it behaves as fault-free, for example the first dotted line shown in Figure 5.2(a)



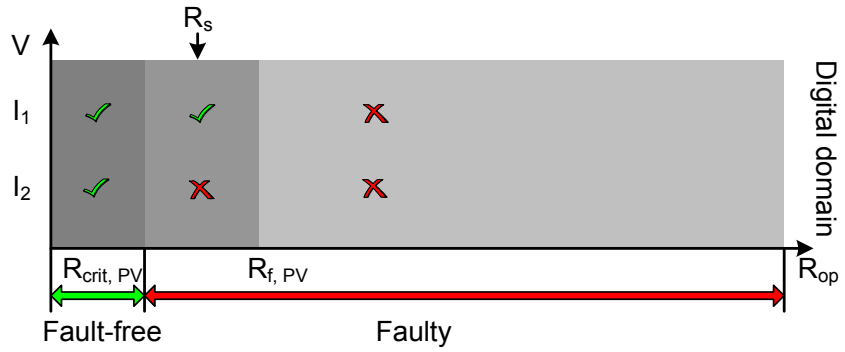
(a) Transition delay behaviour of resistive open R_{op} in nominal operating conditions.



(b) Change in delay behaviour of resistive open "R_{op}" due to change in V_{I1}, V_{I2} and logic threshold voltages of driven gates' (L_{th1} and L_{th2}), when compared with Figure 5.2(a).



(c) Delay faults showing timing critical resistance determined through V_{I1}, signal capture time and logic threshold voltages of driven gates' (L_{th1} and L_{th2}).



(d) Additional delay fault and timing critical resistance, when compared with Figure 5.2(c) leading to increase in fault domain.

Figure 5.2: Transition delay behaviour of resistive open.

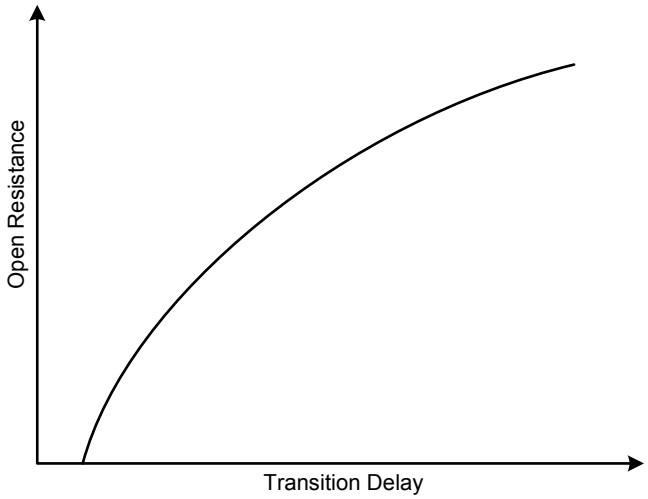


Figure 5.3: Open resistance vs. delay.

marked with $R_{crit,Nom}$. If the transition signal V_{I1} crosses L_{th1} after signal capture time, it is referred as a delay fault, for example, the second (R_S) and the third ($R_{f,Nom}$) dotted lines shown in Figure 5.2(a). $R_{crit,Nom}$, R_S and $R_{f,Nom}$ are different values of R_{op} $\in [0, 10M\Omega]$ shown in Figure 5.1(a). It was found in [28] that resistive opens exhibit a resistance range of $\leq$ 10 MΩ. Simulation results over 10,000 transient simulations using SPICE by sweeping the value of open resistance form 0Ω to 1MΩ based on a 65-nm gate library show that in general, the delay value of an open fault-site increases as the resistance value increases, which is shown in Figure 5.3. Therefore different values of R_{op} shown in Figure 5.2(a) have the following relationship: $R_{crit,Nom} < R_S < R_{f,Nom}$. For each value of open resistance, where $R_{op} \in [0, 10M\Omega]$, the logic values read by inputs I_1 and I_2 can be determined by comparing the transition gate output voltage signal with corresponding logic threshold voltage of the two observation points (OP_{A1} and OP_{A2}) at the signal capture time. These values are shown in Figure 5.2(c), which shows conversion of analog fault behaviour (Figure 5.2(a)) into corresponding digital values. Crosses are used to mark the faulty logic values and ticks to mark the correct ones. Based on the analog fault behaviour, there are three open resistance intervals that can be identified depending on the value of R_{op}. Opens with $R_{op} \in [0, R_{crit,Nom}]$ lead to all inputs reading correct logic value; opens with $R_{op} \in [R_{crit,Nom}, R_{f,Nom}]$, gate OP_{A1} reads faulty value, while OP_{A2} reads correct logic value. Finally, when $R_{op} > R_{f,Nom}$, both gates read faulty logic values. Consequently, each interval $[R_a, R_{a+1}]$ corresponds to a distinct logic behaviour occurring at the open fault site. This logic behaviour together with corresponding resistance intervals of an open fault-site can be used for fault simulation, test generation and diagnosis. The R_{op} value corresponding to $R_{crit,Nom}$ is the minimum detectable resistance of a resistive open fault-site, which is detectable through gate A_1 (indicated by a circle in Figure 5.2(a)) and referred as "timing critical resistance" (R_{crit}), as it represents the crossing point between faulty and correct logic behaviour.

Due to process variation, the behaviour of resistive open fault deviates from nominal

Table 5.1: Delay fault behaviour using Class-I test on 1500 resistive open fault-sites: Nominal ("Nom") and Process Variation ("PV").

R_{op}	Nominal (%)		PV (%)	
(Ω)	Fault-free	Faulty	Fault-free	Faulty
0	100.0	0.0	89.1	10.9
10k	68.3	31.7	56.1	43.9
50k	48.9	51.1	39.6	60.4
100k	35.2	64.8	25.8	74.2
500k	12.6	87.4	8.6	91.4
1M	3.7	96.3	2.3	97.7

operating conditions (Figure 5.2(a) and Figure 5.2(c)). Process variation affects transistor currents and voltages leading to change of gate's logic threshold voltage (L_{th}) and their drive strength affecting delay behaviour of output transition signal. This change introduces additional delay faults. Figure 5.2(b) shows change in transient gate output voltage and logic threshold voltages due to process variation and Figure 5.2(d) shows corresponding logic values in digital domain. From Figure 5.2(b), it can be seen that due to process variation, transition signals V_{I1} and V_{I2} at the output of gates A_1 and A_2 become slower while L_{th1} reduces and L_{th2} increases. It can be seen that in comparison to nominal operating conditions (Figure 5.2(a) and Figure 5.2(c)), the timing critical resistance is detectable through gate A_2 rather than A_1. From test generation point of view, a test generated to propagate the fault effect through gate A_1, as in the case of nominal operating conditions, will lead to a test escape, as the resistance interval $R_{op} \in [R_{crit,PV}, R_{f,PV}]$ will be missed. This is because test through gate A_1 can only detect resistance value from $[R_{f,PV}, 10\ M\Omega]$. Therefore in this case, an additional test is needed to propagate the fault effect through gate A_2 to cover this additional process variation induced delay fault. Figure 5.2 shows that for a given open fault-site the three most influential variables are transient gate output voltage (V_{I1}, V_{I2}), logic threshold voltage (L_{th1}, L_{th2}) and signal capture time.

To illustrate the effect of process variation on delay fault detectability, a simulation was conducted. When considering Class-I delay test (Figure 5.1(a)), the fault-site behaviour is simulated when assuming nominal operating conditions (referred as "Nom") and under the influence of process variation (referred as "PV"). The simulation was conducted on a 65-nm ST Microelectronics gate library using the PTM transistor models [21]. Results are summarised from 1500 resistive open fault-sites and for each fault-site with 600 permutations of process variation by varying three parameters (L, V_{th} and μ_{eff}) using Gaussian distribution with $\pm 3\sigma$ variation. See Section 5.4 for more details on simulation setup. For each fault-site 6 different resistance values were considered. Results are shown in Table 5.1. It can be observed that as the value of open resistance increases (1 MΩ), a higher percentage of fault-sites behave as faulty both in nominal operating conditions

(up to 96.3%) and under the influence of process variation (up to 97.7%). It can also be observed that under "PV", when considering a fault-free case (0 Ω), 10.9% of fault-sites behave as faulty, this phenomenon was earlier observed in [142]. These results clearly indicate that detectability of delay fault is effected by the influence of process variation.

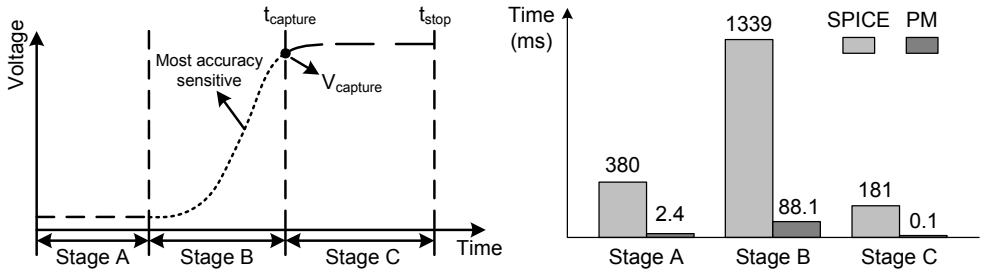
The available fault modelling and simulation techniques that use SPICE with Monte-Carlo simulation to simulate resistive opens under the influence process variation require long computation time. When considering resistive open faults, SPICE fault simulation time per fault site, on average over 1500 fault-sites is about 23.4 minutes (see Section 5.4 for details). Therefore an accelerated delay fault simulation methodology is needed, which is described in Section 5.3.

5.3 Accelerated Resistive Open Fault Simulation Methodology

From the analysis shown in Section 5.2, it is observed that the signal capture time, logic threshold voltages (L_{th1}, L_{th2}), and transient gate output voltages (V_{I1}, V_{I2}) are three variables needed to calculate delay faults and timing critical resistance of an open fault-site (Figure 5.2(b), Figure 5.2(d)). This observation is valid for both nominal operating conditions (Figure 5.2(a)) and under process variation (Figure 5.2(b)). Signal capture time is an input to the proposed methodology. Logic threshold voltage (L_{th}) of a gate is calculated by using the logic threshold voltage calculation algorithm proposed in Section 3.3.1, Chapter 3, which is on average 257 times faster with 3.1% accuracy deviation, when compared with SPICE. This section is organised as follow: Transient gate output voltage is calculated by employing an algorithm described in Section 5.3.1 to accelerate delay fault computation of an open fault-site. Delay faults can be represented in term of timing critical resistance as shown in Figure 5.2. Section 5.3.2 explains an approximation algorithm using the binary search method to calculate timing critical resistance of open fault-site rather than linearly sweeping the resistance value to find the timing critical resistance.

5.3.1 Transient Gate Output Voltage Calculation

Transient gate output voltage (V_{I1}, V_{I2}, Figure 5.2) is needed to accurately calculate delay faults at the output of each fanout gate of an open fault-site (Figure 5.1). To accelerate the calculation of transient gate output voltage without loosing accuracy, the three-step strategy that proposed in Section 4.3.1, Chapter 4 is used. These three steps are transistor models elimination, transistor electrical parameters elimination and step size adjustment. See Section 4.3.1 for details. The calculation of transient gate output voltage can be divided into three stages, which is shown in Figure 5.4(a) with a rising



(a) Transient signal (b) Relative Speedup: SPICE vs PM

Figure 5.4: Transient signal analysis.

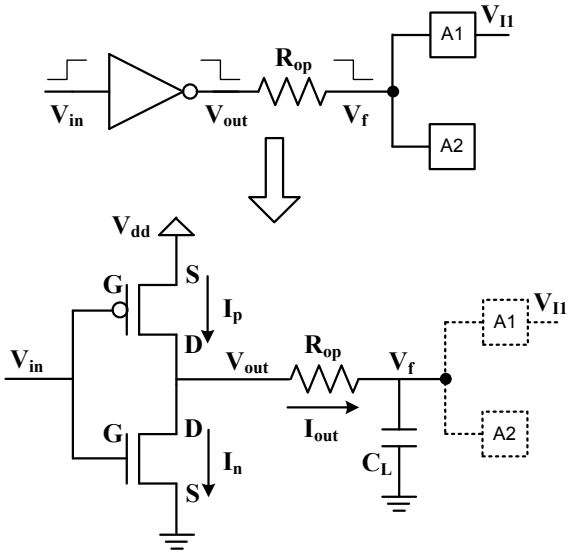


Figure 5.5: Transient analysis at the output of inverter.

transition signal generated through transient simulation. Figure 5.4(b) shows relative speedup achieved by the proposed methodology (three-step strategy) when compared with SPICE (Synopsys HSPICE) in each of the three transient signal stages as shown in Figure 5.4(a). For this comparison, 1500 fault sites from 65-nm STMicroelectronics design library were used. The results reported are, on average over 1500 fault sites when operating in nominal conditions. As can be seen, a significant speed up in each of the three stages is possible through the proposed methodology.

Next, the transient gate output voltage calculation of an open fault-site is discussed. The transient gate output voltage is calculated through the retained transistor sub-models (Table 4.1, marked with "$\surd$", Chapter 4), and uses $V = \frac{Q}{C}$ relationship to calculate the rate of change of transistor terminal charges Q_g, Q_b, Q_s, and Q_d that is associated with the transistor gate, bulk, source, and drain terminals to determine time-dependent transistor terminal voltages V_g, V_b, V_s, and V_d. The change of transistor current is calculated through the I_{ds} equation shown in Eq. (4.1) in Chapter 4 [86]. Figure 5.5 shows an example of Class-I transition test to calculate the transient gate output voltage of the an fault-site. V_{in}, V_{out}, V_f and V_{I1} in Figure 5.5 are the input

Input: Gate library, Transistor model card, Fault-site
Input: V_{in}, V_{dd}, t_{stop}
Output: V_{out}
1: Read gate library and transistor model card.
2: Using the gate library, read gate capacitance value of all fault-site gates.
3: Initial operating points according to V_{in} and V_{dd}
4: t(n) = 0
// t(n) is the current time value
5: **repeat**
6: Calculate capacitance and timing-related terminal charge of each transistor
// Terminal charge: Q_g, Q_b, Q_s, Q_d
7: Convert terminal charge to terminal voltage of each transistor by using V=Q/C
// Terminal voltage: V_g, V_b, V_s, V_d
8: Calculate voltage difference (V_{ds}, V_{gs}, etc.) of each transistor
// For example: V_{ds} = V_d - V_s
9: Calculate current (I_p, I_n) of each transistors using BSIM4 model
// I_p and I_n are calculated by using Eq. (4.1)
10: I_{out}=I_p - I_n
11: V_{out}=$V_{ds,n}$
12: Select time step Δt depending on different stages
// Three different stages are shown in Figure 5.4(a)
13: Calculate next time value t(n+1) = t(n) + Δt(n+1)
14: Update V_{in}: V_{in}(n+1) = V_{in}(n) + ΔV_{in}(n+1)
15: **until** $(t(n) \leqslant t_{stop})$
16: **return** V_{out}

Figure 5.6: Calculation of transient gate output voltage.

transition signal, output voltage of driving gate, faulty output voltage due to resistance open (R_{op}) and transient gate output voltage of an open fault-site. Detailed steps for calculating transient output voltage of an open fault-site (V_{I1}, Figure 5.5) are shown in Figure 5.6.

The algorithm starts by reading gate capacitance of all fault-site gates from the gate library. For a given fault-site, this algorithm operates from driving gates (Inverter, Figure 5.5) and works its way forward to the output of driven gate(s) (Gates "A1" and "A2", Figure 5.5). All transistor nodes are initialised in preparation of transient simulation. Next, the algorithm iterates over $t(n) \in [0, t_{stop}]$ to calculate gate output voltage at each time step $t(n)$ (step-5 to step-15, Figure 5.6). As part of this iterative process, it first calculates time-dependent terminal charges (Q_g, Q_b, Q_s, and Q_d) of each transistor by using the retained BSIM4 transistor sub-model (Table 4.1, marked with "$\surd$", Chapter 4) and NGSPICE (open-source SPICE engine) approximation method [85]. These terminal charges are used to calculate terminal voltages (V_g, V_b, V_s, and V_d), which are then used to compute terminal voltage differences (V_{ds}, V_{gs}) of each fault-site transistor. In the next step, currents in the pull-up (I_p) and pull-down (I_n) networks of each gate are calculated by using I_{ds} (Eq. (4.1), Chapter 4) and leakage current equations (BSIM4.7 Manual [86]). Note $V_{ds,n}$ is the drain-to-source voltage of the pull-down

network. This completes single iteration to compute output current I_{out} and output voltage V_{out}. In each iteration of $t(n)$, step size is dynamically adjusted. This is because stage-B is the most accuracy sensitive stage (Figure 5.4(a)), whereas timing step size can be dynamically adjusted in other two stages without affecting accuracy. Therefore time step Δt is adjusted dynamically by observing the slope of transition signal and the next timing point $t(n+1)$ is calculated by adding time step Δt(n+1) to the current time $t(n)$. Finally, V_{in}(n+1) is updated by adding the difference ΔV_{in}(n+1) to V_{in}(n), where ΔV_{in} changes with Δt. This iterative process continues until the end of simulation time ($t(n) > t_{stop}$). The output transition signal V_{out} (Figure 5.5) is used to calculate the input transition signal V_f to the driven gates "A1" and "A2". It is calculated by using Eq. (5.1) derived through equating current I_{Rop}, where C_L is the load capacitance of the driven gates (A1 and A2). R_{op} is the unknown variable of open resistance value, which is varied from R_{Max} to R_{Min} (resistance value) until timing critical resistance of the fault-site is determined. This is described in Section 5.3.2. By using V_f as the input transition signal to the driven gates and the algorithm shown in Figure 5.6, the transient gate output voltage of an open fault-site V_{I1} can be calculated.

$$\frac{V_{out} - V_f}{R_{op}} = C_L \frac{dV_f}{dt} \tag{5.1}$$

5.3.2 Timing Critical Resistance Calculation

This section shows an approximation algorithm to accelerate the calculation timing critical resistance, which is faster (about 3 times faster as discussed in Section 5.4.3.2) than linearly sweeping the resistance value to find the timing critical resistance. Figure 5.7 shows the approximation algorithm to calculate timing critical resistance of a resistive open fault-site. This algorithm is similar to the binary search (bisection) method, which is used to approximate the timing critical resistance between the two variables, R_{Max} and R_{Min}. For demonstration purposes R_{Max} is set to 1 MΩ and R_{Min} is set to 0 Ω. The algorithm reduces the separation between these two by half in each subsequent iteration. R_{op} (Figure 5.1) is set to the midpoint between R_{Max} and R_{Min} and the output voltage (for example, V_{I1} in Figure 5.1(a)) is calculated by using the simulation flow described in Sec. 5.3.1 (Figure 5.6). At signal capture time ($t_{capture}$; Figure 5.4(a)) the output voltage ($V_{capture}$) is compared with the logic threshold voltage of the driven gate(s), and each iteration reduces the separation between R_{Max} and R_{Min} either by reducing R_{Max} or increasing R_{Min} (step-9 to step-13 in Figure 5.7). These steps are based on the principle that for a rising output transition signal in an open fault-site, $V_{capture}$ reduces as R_{op} increases (Figure 5.3). Therefore if $V_{capture} \leqslant L_{th}$, $R_{op} \geqslant R_{crit}$, and if $V_{capture} > L_{th}$, it means $R_{op} < R_{crit}$. This concept is explained using an illustrative example shown in Figure 5.8. The output rising transition signal of gate A_1 (Figure 5.2(a)) is re-drawn in Figure 5.8, when considering the value of open resistance R_{op} from R_1 to R_3 for a

Input: L_{th}, t_{stop}
Output: R_{crit}
1: Read gate library and transistor model.
2: Generate resistive open fault-site.
// For example as shown in Figure 5.1
3: $LIMIT = 1$ mV
4: Set Initial value of R_{Max}, R_{Min}
// For demonstration purposes R_{Max} is set to 1 MΩ and R_{Min} is set to 0 Ω
5: $R_{tmp} = (R_{Max} + R_{Min})/2$
// This loop is used to get R_{crit} when the output voltage is a rising signal
6: **repeat**
7: $R_{op} = R_{tmp}$
8: $V_{capture} = V_{I1}$ (at $t_{capture}$)
// Using Figure 4.5(a) and Figure 5.6
9: **if** $V_{capture} \geqslant L_{th}$ **then**
10: $R_{Min} = R_{tmp}$
11: **else**
12: $R_{Max} = R_{tmp}$
13: **end if**
14: $R_{tmp} = (R_{Max} + R_{Min})/2$
15: **until** $(|V_{capture} - L_{th}| \geqslant LIMIT)$
16: **return** $R_{crit} = R_{op}$

Figure 5.7: Timing critical resistance approximation algorithm for resistive open faults.

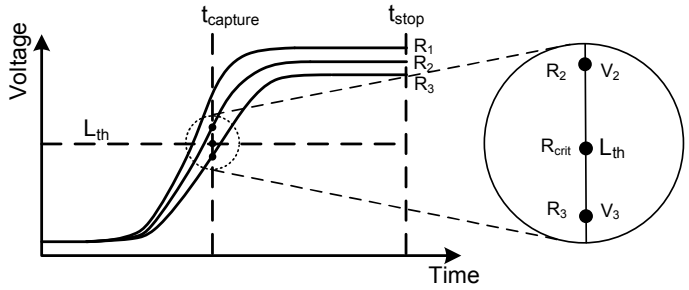


Figure 5.8: An example to illustrate the working of timing critical resistance approximation algorithm shown in Figure 5.7.

fault-site (Figure 5.1). The value of open resistance R_{op} is such that $R_1 < R_2 < R_3$. The crossing point between the signal capture time and the output transition signals is enlarged for clarity (Figure 5.8). It can be seen that the timing critical resistance (R_{crit}) is in between R_2 and R_3 and it can be calculated through a simple bisection method that interpolates between these two resistance values (Step-4 to Step-15, Figure 5.7). The algorithm terminates when the difference in $V_{capture}$ and L_{th} is smaller than the constant value of $LIMIT$. In this work, $LIMIT$ is set to 1 mV, which was found empirically[1] to achieve high accuracy. The algorithm converges using small (on average 13

[1]Similar to the discussion of Table 3.1 in Chapter 3.

or less) number of iterations for all reported results (Sec. 5.4). A similar algorithm can be derived when the gate output voltage is a falling transition signal.

5.4 Simulation Results

This section considers the simulation results using the proposed accelerated delay fault simulation methodology presented in Section 5.3. A delay fault simulation flow for resistive open defects using the proposed methodology is introduced in Section 5.4.1. This flow is used to conduct three sets of simulations. The first two sets of simulation validate the proposed methodology by comparing it with SPICE (Synopsys HSPICE) using a 65-nm gate library. First set of simulation (Sec 5.4.2) validates transient signal computation of gate output voltages. Second set of simulation (Sec 5.4.3) validates the complete simulation flow (Figure 5.9), in nominal operating conditions and under the influence of process variation. The last simulation (Sec 5.4.2) demonstrates that the proposed methodology can be used in different technology nodes by comparing simulation results with SPICE using a 45-nm gate library.

5.4.1 Fault Simulation Flow

Simulations are conducted using a 65-nm ST Microelectronics gate library[2] and PTM transistor model card[3] [21] on Intel Xeon Quad Core 2.7 GHz processor with 12 GB RAM. The gate library consists of 775 gates including simple (NAND, NOR, INV) and compound gates (AO22, OA22 etc.), where on average each gate is available in about 5 different drive strengths. For illustration purposes 1.2 V is used as the nominal operating voltage in all simulations. Figure 5.9 shows the proposed transition delay fault simulation flow for resistive opens, when considering process variation. It can be seen that the flow inputs are gate library (65-nm ST Microelectronics) and respective transistor models (PTM transistor model) and the output is PV-aware transition delay faults (Figure 5.2) for a given fault-site. For each fault-site, 600 permutations are generated through Monte-Carlo simulation.

Figure 5.9 shows the simulation flow, which has seven main blocks. The effect of process variation is incorporated by the Process Variation Permutation Generator. It varies three un-correlated parameters (die-to-die variation): gate length (L), threshold voltage (V_{th}) and mobility (μ_{eff}), which follow Gaussian distribution with $\pm 3\sigma$ variation and standard deviations (σ) of 4% for L, 5% for V_{th} and 21% for μ_{eff}. Within-die variation is not considered because it was shown in Section 3.4.4.2, Chapter 3 that die-to-die variation is likely to cover all the faults due to within-die variation. See more details

[2] Appendix B shows SPICE description of three gates from the gate library.
[3] Appendix C shows SPICE description of the PTM transistor model card.

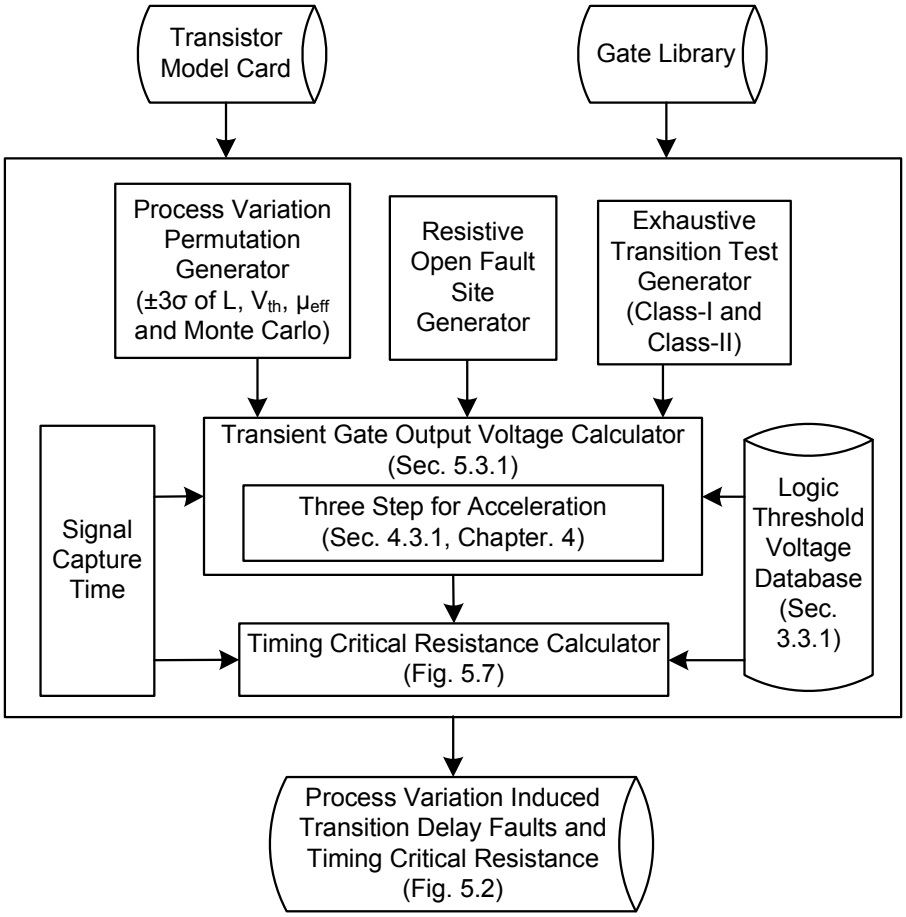


Figure 5.9: Accelerated delay fault simulation flow of resistive opens.

in Section 3.3.3, Chapter 3 for incorporating process variation. Resistive Open Fault-Site generator is used to build each fault-site, which is generated by randomly selecting (driving and driven) gates from the gate library, using n driven gates per fault-site, where $n \in [1, 5]$. In this chapter, 1500 fault-sites are used because beyond 1500 fault-sites, the time consumed by SPICE (needed for comparison) becomes prohibitively long. Each fault-site is tested by using exhaustive transition test (Class-I and Class-II, Figure 5.1) for fault propagation to the output of the fault-site. The output of these three blocks (Figure 5.9) are fed to the Transient Gate Output Voltage Calculator (Sec. 5.3.1) that uses the three-step strategy based on BSIM4 transistor model and an open-source SPICE engine (NGSPICE [85]) to calculate transient gate output voltage (Figure 5.4(a)) at each fanout gate to generate transition delay faults (Figure 5.2(b) and Figure 5.2(d)). For illustration, the signal capture time ($t_{capture}$, Figure 5.4(a)) used in this work is an output transition signal at 80% of V_{dd} for rising transition and at 20% of V_{dd} for falling transition in fault-free designs. The delay faults generated from Transient Gate Output Voltage Calculator are fed to the Timing Critical Resistance Calculator, that uses an approximation method (Figure 5.7) to calculate the timing critical resistance using signal capture time and logic threshold voltage (L_{th}) as inputs. For logic threshold voltage generation, the fast and accurate L_{th} calculation algorithm, which is on average 257 times faster with 3.1% worst case accuracy deviation, when compared with HSPICE (Section 3.3.1, Chapter 3) is used. To clearly demonstrate the speedup and accuracy achieved by the proposed methodology (Sec. 5.3) in comparison with SPICE (Synopsys HSPICE), a pre-computed logic threshold voltage database of the complete gate library is used with both techniques (Proposed and SPICE) to compute timing critical resistance per fault-site. The fault simulation flow shown in Figure 5.9 has been implemented as a prototype software tool with about 8000 lines of code using C/C++.

5.4.2 Transition Delay Calculation

In this section, the accelerated simulation methodology for calculating transient gate output voltage described in Section 5.3.1 is validated by comparing results of transition delay using a single gate (Sec. 5.4.2.1) and across multiple gates (Sec. 5.4.2.2). Results generated by the proposed methodology (PM) is compared with SPICE, in nominal operating conditions and under the influence of process variation. In this section, the transition delay for each gate is calculated by taking the time difference between 20% to 80% of Vdd (t_{delay}, Figure 4.5(a), Chapter 4), as described in 65-nm STMicroelectronics gate library manual.

5.4.2.1 Validation of Single Gate Delay Calculation

The comparison (accuracy) of various gates is shown in Table 5.2. Gates listed in the first column (Table 5.2) are shown because they exhibit highest calculation error when

Table 5.2: Gate delay comparison: Proposed methodology (PM) vs SPICE (Synopsys HSPICE).

Gate	Nominal		PV				Max Error (%)
	SPICE (ps)	PM (ps)	SPICE (ps)		PM (ps)		
			Min	Max	Min	Max	
Simple Gate							
INV	5.5	5.4	3.8	14.7	3.7	14.5	2.6
NAND2	7.0	7.1	4.2	15.6	4.1	15.2	3.0
NOR2	7.6	7.5	4.6	16.2	4.7	16.5	2.9
AND2	7.3	7.2	4.4	16.5	4.3	16.8	2.8
OR2	7.4	7.3	4.7	17.3	4.6	17.7	2.9
Complex Gate							
NAND4	8.4	8.6	4.1	17.4	4.2	17.8	3.1
NOR4	5.5	5.4	3.9	20.5	4.0	19.9	3.2
AND4	8.9	8.7	3.8	17.5	3.9	17.1	3.1
OR4	5.3	5.4	3.8	19.6	3.9	20.1	3.2
XOR2	9.2	9.0	4.3	18.7	4.2	19.2	3.2
XOR3	8.6	8.4	4.9	18.1	4.8	18.6	3.3
AO22	5.7	5.6	3.9	18.4	3.8	18.9	3.2
OA22	5.6	5.5	4.0	17.9	4.1	18.3	3.3

* PM → Proposed Delay Fault Simulation Methodology

compared with SPICE. Column two marked with "Nominal" shows gate delay values in nominal operating conditions, when simulating through SPICE and the proposed methodology "PM". Third column marked with "PV" shows minimum (Min) and maximum (Max) gate transition delays under the influence of process variation. Finally, the last column shows the maximum error found for each gate under the influence of process variation when comparing "PM" with SPICE. Table 5.2 shows that in nominal operating conditions, the highest error observed is 0.2-ps, for example in case of NAND4 (4-input NAND) gate, where error percentage is 2.4% in comparison with SPICE. For this gate (NAND4), output transistion signal comparison is shown in Figure 5.10. When compared with SPICE, it can be seen that the gate transition computation of the proposed methodology shows very close approximation. In case of process variation, the Process Variation Permutation Generator (Figure 5.9) is used for modeling the effect of process variation. Results in Table 5.2 show that under the influence of process variation, over 600 permutations, the maximum calculation error is 3.3%, as in case of XOR3 (3-Input XOR) and OA22 (2, 2-input OR gates connected by a NAND) gates. This accuracy error (3.3%) is highest in these gates because of higher number of transistors (on average about 5 times than inverter), which require additional output voltage approximation.

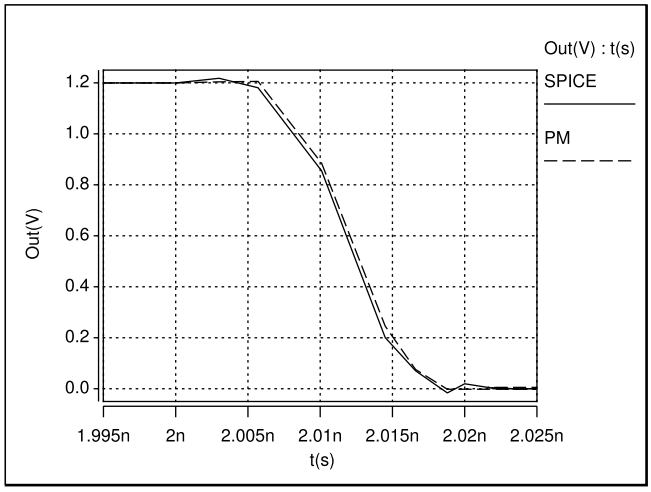


Figure 5.10: Transient gate output voltage calculation of a 4-Input NAND gate (NAND4) in nominal operating conditions: Proposed (PM) vs SPICE.

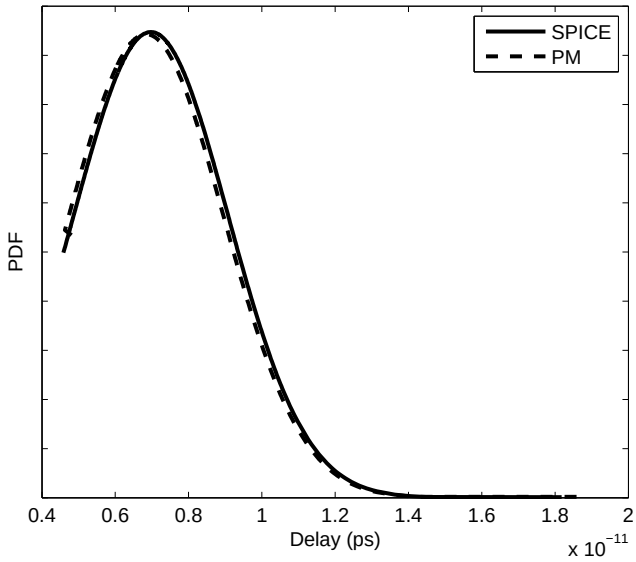


Figure 5.11: Transient gate output voltage comparison of a 3-Input XOR gate (XOR3) under the influence of process variation: PM vs SPICE.

Table 5.3: Gate delay calculation time comparison under the influence of process variation: PM vs SPICE.

Gate	Time (s)		Improvement ($\frac{SPICE}{PM}$)
	SPICE	PM	
INV	166	4.3	39x
NAND2	175	5.5	32x
NOR2	173	4.9	35x
AND2	172	5.2	33x
OR2	173	5.1	34x
NAND4	179	5.8	31x
AND4	180	5.8	31x
NOR4	180	6.2	29x
OR4	179	6.2	29x
XOR2	177	6.3	28x
XOR3	181	6.5	28x
AO22	179	6.2	29x
OA22	180	6.4	28x

* PM → Proposed Delay Fault Simulation Methodology

Gate delay computation for this gate (XOR3) is shown in Figure 5.11. When compared with SPICE, it can be seen that the computed delay of transient gate output voltage is well-captured by the proposed methodology.

Table 5.3 shows simulation runtime improvement of the proposed methodology, when considering the influence of process variation. It can be seen that the maximum runtime improvement is 39 times as in case of INV (Inverter) and the minimum improvement is 28 times as in case of XOR2 (2-input XOR), XOR3 (3-input XOR), and OA22 (2, 2-input OR gates connected by an AND) gates. The difference in simulation time between inverter and these compound gates is because on average, there are 5 times more transistors within these gates, which require additional time for calculating current and voltage values. When considering all gates in the gate library, the proposed method is on average 29 times faster than SPICE.

5.4.2.2 Validation of Multiple Gates Delay Calculation

This section demonstrates the accuracy of the proposed delay calculation algorithm across fault-sites without inserting any fault ($R_{op} = 0\ \Omega$). This is to evaluate error contribution without inserting any fault. Results are compared with SPICE for validation purposes. For these simulations, simulation flow (Figure 5.9) does not utilise "Critical

Table 5.4: Gate delay of fault-site (Figure 5.1) with $R_{op} = 0\ \Omega$ using SPICE and the proposed methodology.

Class	Input of Gate D		Nominal		PV				Max Error (%)
			SPICE (ps)	PM (ps)	SPICE (ps)		PM (ps)		
					Min	Max	Min	Max	
I	↑	0	4.9	4.8	1.3	17.8	1.3	17.4	3.1
	↓	0	5.8	5.8	2.0	19.3	2.0	18.8	3.0
	0	↑	5.6	5.5	1.5	18.2	1.5	18.7	3.2
	0	↓	5.5	5.6	2.2	20.5	2.2	21.2	3.5
II	0	0	6.2	6.3	1.6	24.5	1.6	25.2	3.5
	0	0	7.2	7.1	1.2	26.7	1.2	25.9	3.4

* PM → Proposed Delay Fault Simulation Methodology

Table 5.5: Delay calculation using the proposed methodology in comparison with SPICE over 1500 fault-sites (with $R_{op} = 0\ \Omega$) in nominal operating conditions and under the influence of process variation.

Class	Error (%)						Time Improvement ($\frac{SPICE}{PM}$)
	Nominal			PV			
	Min	Max	Avg	Min	Max	Avg	
I	1.3	3.7	2.1	1.6	5.5	3.1	20x
II	1.1	3.5	2.0	1.8	5.2	3.0	24x

Resistance Calculator" and fault-sites are generated by setting $R_{op} = 0\ \Omega$ by the "Fault Site Generator". Table 5.4 shows results for the fault-site shown in Figure 5.1 with $R_{op} = 0\ \Omega$. The column *Input* in Table 5.4 shows different input vectors for Gate "D". "↑" and "↓" represent rising and falling transition signals. Note that the input vectors of Gate "D" in Class-II are the same but different input vectors are applied to the driven gates. "Min" and "Max" represent the minimum and the maximum delay values obtained from 600 permutations of process variation. The last column of Table 5.4 shows the maximum error in delay calculation for each input combination, when considering nominal operating conditions and under the influence of process variation. It can be seen that the maximum error in delay calculation is 3.5% for both classes (Class-I and Class-II). When considering 1500 such fault-site ($R_{op} = 0\ \Omega$) under the influence of process variation, on average, deviation in delay calculation is $\leq 3.1\%$ for Class-I and $\leq 3\%$ for Class-II as shown in Table 5.5. The last column of Table 5.5 shows the average runtime improvement both in nominal operating conditions and under process variation when comparing SPICE to the proposed methodology (PM). It shows that the average runtime improvement is up to 24 times when comparing with SPICE. The difference in accuracy and runtime improvement between these two test classes is due to additional number of gates output transition signals that have to propagate through in Class-I,

Table 5.6: Timing critical resistance comparison of an open fault-site (Figure 5.1) in nominal operating conditions: PM vs SPICE.

Class	Input of Gate D		R_{crit} (Ω)		Error (%)	Speedup ($\frac{SPICE}{PM}$)
			SPICE	PM		
I	↑	0	1238	1276	3.1	48.2x
	↓	0	847	851	2.6	47.4x
	0	↑	1416	1462	3.2	47.6x
	0	↓	949	927	2.3	47.9x
II	0	0	4361	4265	2.2	56.7x
	0	0	3567	3638	2.0	55.8x

* PM → Proposed Delay Fault Simulation Methodology

Table 5.7: Nominal conditions: accuracy and speed comparison of the proposed methodology with SPICE when calculating timing critical resistance over 1500 fault-sites.

Class	Nominal (%)			Speedup ($\frac{SPICE}{PM}$)		
	Min	Max	Avg	Min	Max	Avg
I	1.7	4.1	2.3	37x	54x	50x
II	1.3	3.5	2.1	44x	58x	55x

which leads to smaller calculation error and faster computation on transition delay for Class-II delay test.

5.4.3 Timing Critical Resistance

In this section, the proposed methodology (Figure 5.9) is validated by comparing the delay faults in terms of timing critical resistance values of resistive open fault with SPICE in nominal operating conditions, Sec. 5.4.3.1, and under the influence of process variation, Sec. 5.4.3.2.

5.4.3.1 Resistive Opens in Nominal Operating Conditions

Simulation results reported in this section utilise the simulation flow (Figure 5.9) for 1500 fault-sites, where each fault-site is generated without using process variation permutation generator. For each fault-site, exhaustive transition test is applied for each of the two delay test classes (Class-I and Class-II, Figure 5.1). Table 5.6 shows simulation results for the fault-site shown in Figure 5.1, with timing critical resistance using both delay test classes. Results in Table 5.6 show that in nominal operating conditions, the maximum error in critical resistance calculation is 3.2% and the lowest runtime improvement is

Table 5.8: Timing critical resistance comparison of an open fault-site (Figure 5.1) under the influence of process variation: PM vs SPICE.

Class	Input of Gate D		R_{crit} under PV (Ω)				Max Error (%)	Speedup $\frac{SPICE}{PM}$
			SPICE		PM			
			Min	Max	Min	Max		
I	↑	0	431	5487	454	5728	5.3	48.4x
	↓	0	368	3487	352	3345	4.3	48.2x
	0	↑	478	6055	498	6340	4.7	47.8x
	0	↓	385	3754	368	3602	4.4	47.5x
II	0	0	1469	10257	1423	9932	3.2	56.4x
	0	0	1126	9076	1164	9344	3.4	56.7x

* PM → Proposed Delay Fault Simulation Methodology

Table 5.9: Effect of process variation: accuracy and speed comparison of the proposed methodology with SPICE when calculating timing critical resistance over 1500 fault-sites.

Class	PV (%)			Speedup ($\frac{SPICE}{PM}$)		
	Min	Max	Avg	Min	Max	Avg
I	1.5	6.9	4.2	39x	53x	48x
II	1.2	5.8	3.9	45x	57x	52x

47.4 times in comparison with SPICE. Table 5.7 shows comparative analysis (SPICE vs PM) of timing critical resistance calculation when considering 1500 fault-sites. Results in Table 5.7 show that on average over 1500 fault-sites, error in accuracy is $\leq$ 2.3%, with speed improvement of $\geq$ 50 times when compared with SPICE. When comparing two test classes, average deviation in accuracy (with SPICE) of Class-II is 0.2% lower and it leads to higher runtime improvement than Class-I test.

5.4.3.2 Resistive Opens under Process Variation

Simulation results reported in this section utilise the complete simulation flow shown in Figure 5.9. Table 5.8 shows the results of critical resistance for the fault-site shown in Figure 5.1 using two classes of delay test under the influence of process variation. "Min" and "Max" represent the smallest and the largest timing critical resistance per test obtained from 600 permutations of process variation. The second last column shows the maximum calculation error for each test when compared with SPICE. It can be seen that the error in critical resistance calculation is $\leq$ 5.3% and the runtime improvement is between 47.5 to 56.7 times when compared with SPICE. Table 5.9 shows simulation results of timing critical resistance calculation over 1500 fault-sites to compare accuracy error and runtime improvement of the proposed delay fault simulation methodology in

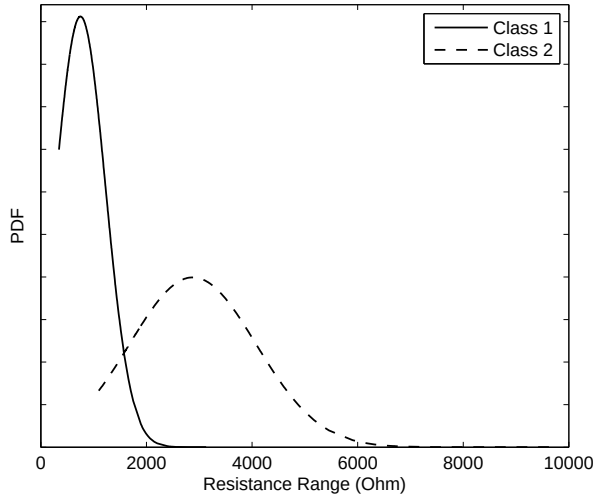


Figure 5.12: Class-I vs Class-II: resistance coverage comparison under the influence of process variation for a fault-site (Figure 5.1).

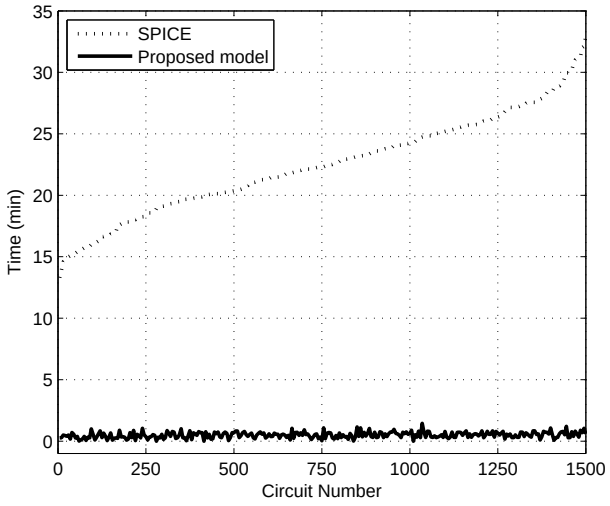


Figure 5.13: Computation time improvement: proposed methodology vs SPICE.

Table 5.10: Simulation time comparison: SPICE vs. proposed model using Intel Xeon Quad Core 2.7 GHz processor with 12 GB RAM.

CKT	# Gates	# Opens	Time (Days)	
			SPICE	PM
b10	100	113	1.77	0.04
b12	658	665	10.95	0.24
b14	1465	1485	24.06	0.50
b20	5016	5773	93.81 (Estimated)	1.96

comparison with SPICE. It can be seen that on average over 1500 fault sites, accuracy error is $\leq$ 4.2% and runtime improvement is up to 52 times. When comparing two test classes (Class-I and Class-II), simulation results show that class-II leads to higher runtime improvement with less deviation in accuracy (with SPICE) when compared with Class-I test. However, when comparing resistance coverage by two test classes, it can be observed from Table 5.8 that Class-I can detect lower critical resistance values (352 Ω) than Class-II (1164 Ω) test. This is elaborated in Figure 5.12, which shows resistance values of Class-I and Class-II test (fault-site shown in Figure 5.1) to indicate that Class-I has higher critical resistance coverage than Class-II. Results over 1500 fault-sites also show that, on average Class-I test has 25.1% higher likelihood of detecting lower timing critical resistance than Class-II delay test.

In terms of simulation runtime improvement, Figure 5.13 shows simulation time of Class-I test over 1500 fault-sites using the proposed delay fault simulation methodology (PM) and SPICE. It clearly shows that the proposed methodology significantly reduces simulation time, on average runtime improvement is up to 52 times when compared with SPICE. The runtime improvement is achieved by using a three-step strategy (Section 5.3.1) to accelerate the calculation of transient gate output voltage and an approximation algorithm (Section 5.3.2) to calculate the timing critical resistance of an open fault-site. Table 5.10 shows delay fault simulation time comparison, when using SPICE and the proposed methodology. For comparison, four ITC'99 benchmark designs are synthesised using 65-nm STMicroelectronics gate library with Synopsys Design Compiler. For each design, exhaustive list of fault-site is considered and exhaustive set of Class-I test are used, as Class-I shows higher timing critical resistance coverage (Table 5.8 and Figure 5.12). For each design, the number of gates and fault-sites are shown in second and third columns respectively. The last column shows simulation time using SPICE and the proposed methodology. It can be observed that the proposed methodology significantly reduces delay fault simulation time when compared with SPICE. In case of "b14" design with 1465 gates and 1485 open fault-sites, simulation time with SPICE is 24.06 days. In comparison delay fault simulation time of the proposed methodology is only 0.50 days. The last row of Table 5.10 shows estimated runtime for a comparatively larger design "b20" with 5016 gates and 5773 fault-sites. SPICE runtime is estimated to avoid prohibitively long computation time by using average runtime results shown in Table 5.9. It can be observed that using the proposed methodology, delay fault simulation time is only 1.96 days. The simulation time of the proposed methodology can be further reduced by using parallel computing through Cluster server.

5.4.4 Simulation using 45-nm Technology

This fault simulation flow shown in Figure 5.9 has been demonstrated on a 65-nm gate library, but it can be used with other technology nodes. This section shows simulation

Table 5.11: Varied process parameters in 45-nm gate library.

Parameter	Mean (μ)	Std. Deviation (σ)
L	50-nm	5-nm
V_{thn}	0.471 V	0.045 V
V_{thp}	-0.423 V	0.045 V
μ_{effn}	435.9 cm^2/V.s	87.2 cm^2/V.s
μ_{effp}	43.2 cm^2/V.s	8.6 cm^2/V.s
T_{oxn}	1.75-nm	0.15-nm
T_{oxp}	1.85-nm	0.15-nm

results by applying the flow shown in Figure 5.9 to a 45-nm gate library. The flow uses the FreePDK 45-nm gate library [151] with 45-nm PTM transistor model card[4] [21]. The gate library consists of a variety of gates including simple (NAND, NOR, INV) and compound gates (AOI22, OAI22 etc.). According to the gate library manual, 0.9 V is used as the nominal operating voltage in all simulations. The effect of process variation is modelled by varying four un-correlated transistor parameters: gate length (L), threshold voltage (V_{th}), mobility (μ_{eff}) and oxide thickness (T_{ox}), which follow Gaussian distribution with $\pm 3\sigma$ variation. The mean and standard deviation (σ) for both NMOS/PMOS transistors are shown in Table 5.11, which shows that the standard deviations are 5-nm for L, 45 mV for V_{th}, 87.2 cm^2/V.s for μ_{eff} in NMOS, 8.6 cm^2/V.s for μ_{eff} in PMOS and 0.15-nm for T_{ox}. Table 5.11 is based on data for relevant parameters reported in [2, 10, 17, 152, 153]. Note that for the thickness of gate oxide (T_{ox}), 0.15-nm standard deviation reflects the thickness of one atom layer [2]. The variation of these four transistor parameters is incorporated by the Process Variation Permutation Generator shown in Figure 5.9, which generates 600 permutations through Monte-Carlo simulation for each fault-site. The rest of the blocks shown in Figure 5.9 remains the same as described in Section 5.4.1. The following sub-sections validate the proposed methodology using 45-nm gate library by comparing the calculation of logic threshold voltage (Section 5.4.4.1), gate transition delay (Section 5.4.4.2) and timing critical resistance (Section 5.4.4.3) with SPICE in nominal operating conditions and under the influence of process variation.

5.4.4.1 Logic Threshold Voltage

The flow shown in Figure 5.9 uses the logic threshold voltage (L_{th}) calculation algorithm proposed in Section 3.3.1, Chapter 3 to calculate the logic threshold voltages of an open fault-site, which is needed for generated delay fault (Figure 5.2). Results based on a 65-nm gate library show that the algorithm is on average 257 times faster with 3.1% worst

[4]Appendix C shows SPICE description of the PTM transistor model card.

Table 5.12: Logic threshold voltage calculation in comparison with SPICE using 45-nm gate library

Gate	Node	Logic Threshold Voltage					Speedup
		Voltage (V) in Nom		Error (%) under PV			
		LG	SPICE	Min	Max	Avg	($\frac{SPICE}{LG}$)
INV	A	0.4377	0.4362	0.01	1.4	0.8	2603x
NAND2	A	0.4124	0.4218	0.03	5.3	2.8	380x
	B	0.4069	0.4141	0.42	5.9	2.5	234x
NAND3	A	0.4155	0.4321	0.35	6.8	3.9	8x
	B	0.4128	0.4295	0.78	7.2	**4.4**	10x
	C	0.4218	0.4365	0.53	6.6	4.1	7x
NOR2	A	0.4546	0.4457	0.11	5.8	3.0	353x
	B	0.4557	0.4466	0.32	6.0	2.7	331x
NOR3	A	0.4627	0.4468	0.56	6.3	3.6	9x
	B	0.4622	0.4475	0.6	6.8	3.7	**6x**
	C	0.4583	0.4431	0.42	6.8	4.0	11x
AND2	A	0.4439	0.4602	0.33	5.6	3.6	263x
	B	0.4424	0.4586	0.41	6.4	3.9	289x
OR2	A	0.4672	0.4511	0.26	6.2	3.5	213x
	B	0.4406	0.4515	0.5	5.8	3.1	223x
XOR2	A	0.4501	0.4612	0.33	7.3	4.0	56x
	B	0.4599	0.4452	0.68	5.9	3.4	63x
XNOR2	A	0.4678	0.4521	0.64	6.7	3.9	66x
	B	0.4621	0.4489	0.57	6.3	3.6	62x

* LG $\rightarrow$ Logic Threshold Voltage Calculation Algorithm

case accuracy deviation, when compared with SPICE, which is shown in Section 3.4.2, Chapter 3. The logic threshold voltage calculation algorithm can be used for different technology nodes, which requires the algorithms shown in Figure 3.7 and Figure 3.8 in Chapter 3 to read the gate library with respective transistor model card. This section demonstrates the logic threshold voltage calculation algorithm on a 45-nm technology by comparing the results with SPICE in nominal operating conditions and in the presence of process variation. The results of comparison (accuracy and speed) are shown in Table 5.12. The first column of Table 5.12 shows gates that are expected to cause higher approximation error in the 45-nm gate library. Higher error is expected in gates with transistors in series (Figure 3.8, Chapter 3). The second column shows different input nodes of each gate. The third column of Table 5.12 shows the voltage values in nominal operating conditions and the error in accuracy when considering process variation. The minimum (Min), maximum (Max) and average (Avg) errors per gate-input per gate are shown in Table 5.12. It can be seen that using the logic threshold calculation algorithm, the minimum error is 0.01% (INV "Inverter") and the maximum error is 7.2% (input-B of NAND3 "3-input NAND gate") when compared with SPICE. Error is highest in case of NAND3 gate as it consists of 3 transistors in series in the pull-down network, which loses accuracy when approximating transistor current I_n as shown in Figure 3.8, Chapter 3. The maximum average error under process variation is 4.4% as shown in Table 5.12, which is also observed from input-B of NAND3 gate. The last column of Table 5.12 shows the relative runtime improvement when comparing the logic threshold calculation algorithm and SPICE ($\frac{SPICE}{LG}$). The maximum improvement is in case of INV (2603 times), and least improvement is in case of NOR3 gate (3-input NOR), which is 6 times. This is because NOR3 gate has three transistors in series in the pull-up network and approximation of current (Figure 3.8, Chapter 3) for each transistor is needed to compute logic threshold voltage of each gate input. When considering all gates in the 45-nm gate library, it was found that average runtime improvement is 239 times with average error of 4.4% by using the logic threshold calculation algorithm.

5.4.4.2 Gate Transition Delay

This section validates the accelerated delay fault simulation methodology for calculating transient gate output voltage described in Section 5.3.1 by comparing the results of gate transition delay in 45-nm gate library using the proposed delay fault simulation methodology (PM) with SPICE. In this section, the transition delay of each gate is calculated by taking the time difference between 10% to 90% of Vdd (t_{delay}, Figure 4.5(a), Chapter 4), as described in 45-nm FreePDK gate library manual [151]. Table 5.13 shows gate delay results to compare speed improvement and accuracy deviation between the proposed methodology and SPICE. The first column of Table 5.13 lists the gates in the 45-nm gate library that exhibit highest calculation error when compared with SPICE. Column two marked with "Nominal (ps)" shows gate delay values in nominal

Table 5.13: Gate delay comparison when using 45-nm gate library: proposed methodology vs SPICE

Gate	Nominal (ps)		PV (ps)				Max Error (%)	Speedup ($\frac{SPICE}{PM}$)
	SPICE	PM	SPICE		PM			
			Min	Max	Min	Max		
Simple Gate								
INV	9.5	9.4	4.0	19.2	3.9	18.8	2.2	40x
NAND2	11.5	11.3	5.8	25.8	5.7	25.2	2.6	34x
NOR2	10.1	10.3	5.0	25.6	5.1	25.1	2.5	33x
AND2	8.5	8.3	5.9	22.6	5.8	23.3	3.2	32x
OR2	8.8	8.6	5.7	22.4	5.8	23.2	3.6	32x
Complex Gate								
NAND3	15.8	16.3	9.1	31.4	8.9	30.4	3.3	28x
NOR3	17.1	16.6	8.5	36.6	8.7	37.8	3.5	26x
XOR2	12.6	13.0	7.6	27.4	7.4	28.4	3.7	27x
XNOR2	16.8	16.3	8.5	36.9	8.6	35.5	3.8	26x
AOI22	15.9	16.4	9.2	33.6	9.0	35.0	4.2	**24x**
OAI22	16.5	16.1	8.8	35.8	9.0	34.3	**4.3**	25x

* PM → Proposed delay fault simulation methodology

operating conditions, when simulating through SPICE and the proposed methodology "PM". Third column marked with "PV (ps)" shows minimum (Min) and maximum (Max) delays under the influence of process variation (over 600 permutations). The fourth column shows the maximum error found for each gate over 600 permutations of process variation when comparing "PM" with SPICE. Table 5.13 shows that in nominal operating conditions, the highest error (0.5-ps, 3.2%) is observed in case of AOI22 (4-input AND-OR) gate, while in case of process variation, the maximum calculation error over 600 permutations is 4.3% as obseved in case of OAI22 (4-input OR-AND) gate. The highest accuracy errors occur in these gates (AOI22 and OAI22) because of higher number of transistors (on average about 4 times than inverter), which require additional output voltage calculation. The last column of Table 5.13 shows simulation runtime improvement ($\frac{SPICE}{PM}$) of the proposed methodology, when considering the influence of process variation. For the listed gates, maximum speedup (40x) is achieved in case of INV, while AOI22 shows minimum (24x) improvement. The difference in speedup is due to different number of transistors in INV and AOI22 gates, which require additional time for calculating current and voltage values. On average the proposed methodology is 30 times faster with 4.3% worst case deviation in accuracy, when considering all gates in the 45-nm gate library.

Table 5.14: Timing critical resistance comparison of an open fault-site (Figure 5.1) using 45-nm technology node: PM vs SPICE.

Class	Input of gate D		R_{crit} (Ω)						Max Error (%)	Speedup ($\frac{SPICE}{PM}$)
			Nominal		PV					
			SPICE	PM	SPICE		PM			
					Min	Max	Min	Max		
I	↑	0	2832	2759	539	12663	510	12154	**5.4**	46x
	↓	0	1974	1921	302	10212	292	9753	5.1	45x
	0	↑	2661	2723	552	12872	564	13527	5.1	46x
	0	↓	2115	2165	324	9879	338	10365	4.9	47x
II	0	0	3417	3347	884	16130	897	16594	3.1	54x
	0	0	2465	2509	789	15244	806	15768	3.5	56x

5.4.4.3 Timing Critical Resistance

Simulation results reported in this section utilise the simulation flow (Figure 5.9) for 1500 fault-sites, where each fault-site is generated using 600 permutations of Monte-Carlo simulation by varying four transistor parameters: L, V_{th}, μ_{eff} and T_{ox}. Table 5.14 shows simulation results for the fault-site shown in Figure 5.1, with timing critical resistance using both delay test classes (Class-I and Class-II). The column in Table 5.14 marked with "Nominal" shows timing critical resistance values in nominal operating conditions while the column marked with "PV" shows smallest (Min) and largest (Max) critical resistance values under the influence of process variation for both methods (proposed methodology (PM) and SPICE). The last two columns of Table 5.14 show that the maximum error over 600 permutations of process variation and the runtime improvement when comparing the proposed methodology with SPICE. Results in Table 5.14 show that the maximum error in critical resistance calculation is 5.4% and the lowest runtime improvement is 45 times in comparison with SPICE, which is both observed in case of Class-I test. When comparing two test classes over 1500 fault-sites, simulation results show that class-II leads to higher runtime improvement with less deviation in accuracy (with SPICE) when compared with Class-I test. The difference in accuracy and speed between these two test classes is due to additional number of gates output transition signal has to propagate through in Class-I, which leads to smaller calculation error and faster calculation time for Class-II delay test. When comparing resistance coverage of the two test classes, it is found that Class-I can detect lower timing critical resistance values (302 Ω) than Class-II (789 Ω) test, which is the same trend as shown in Table 5.8 and Figure 5.12.

When considering simulation results of timing critical resistance calculation over 1500 fault-sites, Table 5.15 shows the best (B) and the worst (W) case accuracy and runtime

Table 5.15: Performance comparison over 1500 open fault-sites using 45-nm technology node.

Driving Gate	Driven Gate	Rcrit (Ω)		Error (%)	Speedup ($\frac{SPICE}{PM}$)
		SPICE	PM		
INV	INV	2504	2478	**1.04** (B)	53x
AOI22	NAND3	5779	6216	**7.56** (W)	42x
INV	INV	1816	1790	1.43	**57x** (B)
NAND3	NAND3	15324	15946	4.06	**38x** (W)
Average results				4.43	50x

* B → Best case W → Worst case

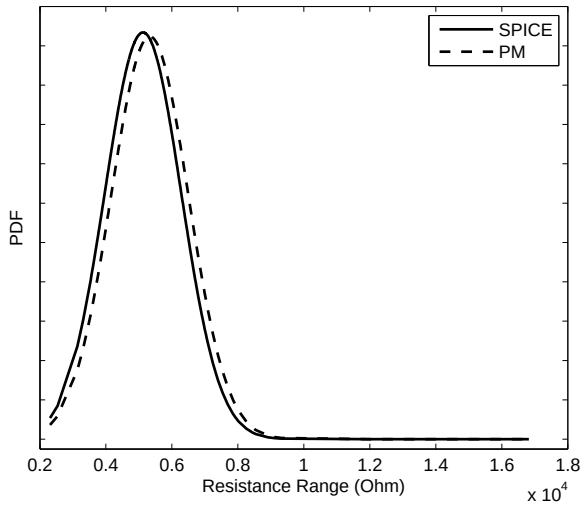


Figure 5.14: The effect of process variation on timing critical resistance of an open fault-site with driven gate AOI22 and driving gate NAND3.

improvement over 1500 fault-sites both in nominal operating conditions and under process variation. The first column of Table 5.15 shows driving gate of each fault-site and the second column shows driven gate. The third column shows selected timing critical resistance values from 600 permutations of process variation that show best case (B), or worst case (W) accuracy percentage error and relative speedup ($\frac{\text{SPICE}}{\text{PM}}$), when comparing the proposed methodology with SPICE. Results show that the best case accuracy (1.04%) and speedup (57x) is in case of a fault-site driven by an inverter with an inverter as a fanout gate. This is because inverter is the simplest gate, which requires minimum number of voltage and current calculation steps. The worst case error (7.56%) is found in case of a fault-site driven by AOI22 and NAND3 as a fanout gate, which is further shown in Figure 5.14 when considering the effect of process variation. The minimum speedup (38x) is in case of a fault-site driven by NAND3 with NAND3 as fanout gate. Results in Table 5.15 show that using the proposed methodology on a 45-nm gate library

is on average 50 times faster with 4.43% error in accuracy when comparing with SPICE in term of timing critical resistance calculation over 1500 fault-sites.

5.5 Concluding Remarks

This chapter has analysed the delay behaviour of resistive open defects, and it has shown how their detectability is affected due to process variation (Figure 5.2). It provides simulation results to demonstrate high computation cost of using SPICE in case of variation-aware fault modelling and simulation of resistive opens. Simulation results show that using ITC'99 benchmark designs synthesised with 65-nm gate library, b14 design with 1465 gates and 1485 open fault-sites requires 24.06 days to complete fault simulation by using SPICE. This issue of high computation cost is addressed in this chapter through an accelerated process variation-aware delay fault simulation flow (Figure 5.9), which employs two efficient algorithms to compute delay faults of an open fault-site (Figure 5.2). The first algorithm consists of transistor model elimination, transistor electrical parameters elimination and step size adjustment to reduce the computation time of transient gate output voltage calculation, which is needed for generating delay faults. The second algorithm uses a binary search method to determine timing critical resistance of an open fault-site. Simulations are conducted on a 65-nm gate library with a PTM transistor model card. Results are compared with SPICE, and it is shown that on average, the proposed methodology is up to 52 times faster with 4.2% error in accuracy, when computing timing critical resistance over 1500 open fault-sites (Table 5.9). This simulation flow has also been validated using a 45-nm gate library with a 45-nm PTM transistor model card and it is shown that the proposed flow is on average 50 times faster with 4.43% error in accuracy when compared with SPICE.

Chapter 6

Conclusions and Future Work

The continuous scaling of CMOS technology has enabled higher clock frequencies, lower power and higher density for digital ICs. However, advances in technology have also led to more manufacturing defects which affect logic function and delay performance leading to IC failures. Therefore manufacturing defects are aggressively targeted by industry during manufacturing test. The most prominent defect types in deep submicron technology are resistive bridges and resistive opens. The research in this thesis has investigated the manufacturing defects behaviour to show how their detectability is affected by process variation. Additional logic and delay faults are introduced as a result of process variation, and therefore test without considering process variation will lead to loss of fault coverage. Fault modelling and fault simulation serve as a backbone of manufacturing test. Recent research has shown that fault modelling and fault simulation using SPICE is accurate but it requires a long computation time to simulate the additional logic and delay faults introduced as a result of process variation. This thesis addresses the problem of long computation time by developing fast and accurate variation-aware fault modelling techniques and simulation methodologies for resistive bridge and resistive open defects, which are summarised in the next section followed by the proposed future work.

6.1 Thesis Contributions

Recent research has shown that process variation affects the quality of manufacturing test. Fabrication process variation is mainly due to random dopant distribution, sub-wavelength lithography, line edge roughness and stress engineering [10, 11, 12]. Process variation is considered as die-to-die variation and within-die variation [18, 19]. Die-to-die process parameters variation can be modelled as independent and random fluctuations while within-die variation is considered to be systematically spatially correlated [20, 25]. It was shown that the logic and delay behaviours of resistive open and resistive bridge

defects are affected by process variation leading to additional faulty behaviour and may lead to loss of fault coverage [2, 29, 30, 79, 92]. The first objective of this thesis[1] is met by analysing the logic and delay behaviour of resistive bridge and resistive open defects under the influence of process variation. It is found that the logic fault of resistive bridge defects is determined by the drive strength of the driving gates and the logic threshold voltages of the successor gates. Additional logic faults occur when drive strength and logic threshold voltages are changed due to process variation. The most influential parameters for computing delay faults are transient gate output voltage, logic threshold voltage and signal capture time, which are valid for both nominal operating conditions and under process variation.

Fault modelling and fault simulation through SPICE is the most accurate method of modelling and simulating fault behaviour of deep submicron defects. However, SPICE requires a long computation time to model and simulate fault behaviour when considering process variation [2, 3, 79, 80]. Therefore new and efficient fault modelling and simulation techniques targeting resistive bridges and resistive opens are needed for manufacturing test to increase fault coverage and reduce test costs. The second objective of this thesis is to address the problem of long computation time to model and simulate logic fault behaviour of resistive bridge defects when considering the influence of process variation. This objective is met by developing a fast and accurate technique to model the effect of process, voltage and temperature variation on resistive bridge defects using the BSIM4 (I-V_{ds}) transistor model, as discussed in Chapter 3. The proposed modelling technique employs two efficient algorithms to compute bridge logic faults. The first algorithm (Section 3.3.1) calculates the logic threshold voltage of gate inputs driven by the bridge fault-site. The second algorithm (Section 3.3.2) calculates the critical resistance by using a bridged net voltage approximation algorithm, which can use either a linear search method (Section 3.3.2.1) or a binary search method (Section 3.3.2.2). The effect of process variation is considered for both die-to-die variation and within-die variation. Die-to-die variation is modelled by using three un-correlated transistor parameters: L, V_{th} and μ_{eff}, with Gaussian distributions and standard deviations of 4% of mean for L, 5% of mean for V_{th} and 21% of mean for μ_{eff} (Table 3.2) and the within-die variation is modelled by using spatially correlated gate length (L) variation (Eq. (3.7)). Variation in supply voltage is modelled by varying the supply voltage and the effect of temperature variation is modelled by using BSIM4 temperature dependent transistor models. The effect of voltage and temperature variation are incorporated by varying their respective values within prescribed gate library operating ranges, where the supply voltage is varied from 0.8 V to 1.2 V with the step size of 0.1 V and temperature is varied between three values: -40°C, 25°C and 125°C, which are the minimum, nominal and maximum working temperatures for 65-nm ST gate library. The proposed modelling technique is extensively validated through comparison with HSPICE when operating at nominal (1.2

[1] All objectives are listed in Section 1.5, Chapter 1.

V and 25°C) condition. It is shown that the worst-case error for logic threshold generation algorithm is 3.1%. When calculating the critical resistance through the bridged net voltage approximation algorithm, it is shown that the binary search algorithm (worst-case error 0.73%) is 41 times faster than the linear search algorithm (worst-case error 0.8%). Therefore, using these two algorithms together (logic threshold voltage calculation algorithm and critical resistance calculation algorithm based on binary search) for resistive bridges under the influence of process, voltage and temperature variation over 350 fault-sites, the worst-case error is 2.64%, when compared with HSPICE. In terms of run-time improvement it is shown that, on average, over 350 fault-sites, the proposed modelling technique is 53 times faster than HSPICE (Figure 3.20). The effect of spatially correlated gate length (L) variation is found to be covered by the variation effects of the three un-correlated parameters: L, V_{th} and μ_{eff} (Figure 3.21). Therefore a test considering die-to-die variation is likely to cover all logic faults due to within-die variation.

The third objective of this thesis addressed the problem of long computation time to simulate the variation induced delay faults for resistive bridge defects, which is met by developing an accelerated delay fault simulation methodology for resistive bridges as discussed in Chapter 4. Chapter 4 presented an accelerated delay fault simulation methodology to reduce the calculation time of transient gate output voltage which is needed to accurately compute delay faults when considering the influence of process variation. The acceleration is achieved by employing a three-step strategy, which consists of transistor model elimination, transistor electrical parameters elimination and step size adjustment. The first step involves identifying and retaining transistor models that affect the calculation accuracy of transient gate output voltage. The second step involves eliminating unnecessary electrical parameters within the retained transistor models obtained from the first step. The final step involves appropriately selecting step size during different stages of transient analysis when calculating gate output voltage. The presented methodology has been incorporated in an open-source SPICE (NGSPICE) with a BSIM4 transistor model. The effect of process variation is modelled by using die-to-die variation as discussed in Chapter 3. Within-die variation is not considered because die-to-die variation is likely to cover all the faults due to within-die variation (Chapter 3). Results based on a 65-nm gate library show that the proposed methodology is on average 17.4 times faster with 5.2% error in accuracy, when compared to SPICE (Table 4.4). Using a 65-nm gate library, a design with 3734 gates and 1194 bridge fault-sites, it was found that SPICE requires more than 32 days while the proposed methodology requires less than 2 days to complete delay fault simulation. The proposed methodology is used to analyse the delay behaviour of resistive bridge defects using three delay test classes (Figure 4.1). It was found out that the resistance coverage achieved by using any class of delay test is significantly higher than when using logic test, and this trend continues even with logic test at lower voltage and delay test at nominal voltage. When comparing three different classes of delay test under the influence of process variation, it is found

that the maximum coverage is achieved by using Class-I delay test. This trend continues in nominal operating conditions as well.

Chapter 5 addressed the problem of the long computation time of simulating delay fault behaviour of resistive open defects when considering the impact of process variation. To address the issue of high computation cost, this chapter has presented an accelerated delay fault simulation methodology (Figure 5.9), which meets the last objective of the thesis. The developed methodology employs two efficient algorithms to compute delay faults of an open fault-site (Figure 5.2). The first algorithm employs the method proposed in Section 4.3.1, Chapter 4, which uses transistor model elimination, transistor electrical parameters elimination and step size adjustment to reduce the computation time of the transient gate output voltage calculation that is needed for generating delay faults. The delay faults of an open fault-site can be represented in terms of timing critical resistance. The second algorithm uses the binary search method to determine timing critical resistance of an open fault-site. Simulation results using a 65-nm gate library show that, on average, the proposed methodology is up to 52 times faster with 4.2% error in accuracy, when computing timing critical resistance over 1500 open fault-sites (Table 5.9). Simulations are conducted using ITC'99 benchmark designs synthesised with a 65-nm gate library; in the case of the b14 design with 1465 gates and 1485 open fault-sites, it was found that it requires 24.06 days to complete fault simulation by using SPICE, while using the proposed methodology, the delay fault simulation only required half a day. The simulation flow shown in Figure 5.9 can be used with any other technology nodes, which will require a gate library with respective transistor model card, and appropriate values of mean and standard deviation of transistor parameters. This is demonstrated by using a 45-nm FreePDK gate library with respective PTM transistor model card by varying four transistor parameters: L, V_{th}, μ_{eff} and T_{ox}, where each follows Gaussian distribution (Table 5.11, Chapter 5). Results show that on average the proposed methodology is 50 times faster with 4.43% error in accuracy in term of calculating timing critical resistance.

The contributions of this thesis provide novel, fast and accurate fault modelling technique and fault simulation methodologies for resistive bridge and resistive open defects when considering the influence of process variation. The developed technique and methodologies are supported by extensive and realistic simulation results and compared with state-of-the-art EDA tools. It is hoped that the developed fault modelling technique and simulation methodologies in this thesis will make useful contributions towards the development of next generation process variation-aware test methods and EDA tools to improve test quality and reduce test costs in deep submicron manufacturing design. This is because the availability of effective and low-cost test methods developed specifically to mitigate the impact of process variation are of paramount importance if the test cost of nano-scale integrated circuits is to remain acceptable for the highly competitive microelectronics industry.

6.2 Future Work: Variation-aware Test Generation

A worthy future work involves using the developed fault modelling technique and simulation methodologies to incorporate them as part of the test generation process. This will lead to the development of efficient logic and delay test pattern generation methods for deep submicron defects (resistive bridges and resistive opens) that take process variation into account to reduce test cost and improve test quality. Test without considering process variation can lead to test escapes [2]. Therefore new and high quality manufacturing test methods targeting resistive bridge and resistive open defects are needed to minimise test escapes and to reduce test application time of nanometre ICs. Research in variation-aware manufacturing test is still in its infancy and significant academic and novel research is still needed. Variation-aware test is currently receiving considerable attention worldwide.

There are two specific areas for further research in variation-aware test generation. The first area targets delay faults due to resistive bridge and resistive open defects, while the second addresses logic faults due to resistive bridges. Both tasks will utilise the fault models and simulation methodologies developed in this thesis. The first stage will use the accelerated delay fault simulation methodology developed in Chapter 4 and Chapter 5 to develop a variation-aware test generation method for delay fault due to resistive bridge and resistive open defects. The second stage will use the proposed modelling technique described in Chapter 3 to develop a variation-aware test generation method for logic fault due to resistive bridge defects. The outcome from the future research would be practical test solutions that are attractive to both industrial exploitation and further academic research. It is timely and responds to present and future industrial needs. The key objectives of the future research include the following:

1. Using the fault simulation methodologies developed in Chapter 4 and Chapter 5 to identify variation-induced delay faults of resistive bridges and resistive opens that need to be targeted during test pattern generation. Develop process variation-aware test pattern generation methods for delay test leveraging the identified variation-induced delay faults to improve test quality (less test escapes) and reduce test cost (less test application time);

2. Using the fault model developed in Chapter 3 to identify variation-induced logic faults of resistive bridges that need to be targeted during test pattern generation. Develop process variation-aware test pattern generation methods for logic test leveraging the identified variation-induced logic faults to improve test quality and reduce test cost;

3. Evaluate the developed test generation methods using comprehensive simulation with nanometre benchmark circuits.

Appendix A

SPICE Simulation

This appendix expands the discussion in Section 1.3.3 to show more details of SPICE simulation. Fault modelling and fault simulation through SPICE is accurate, because

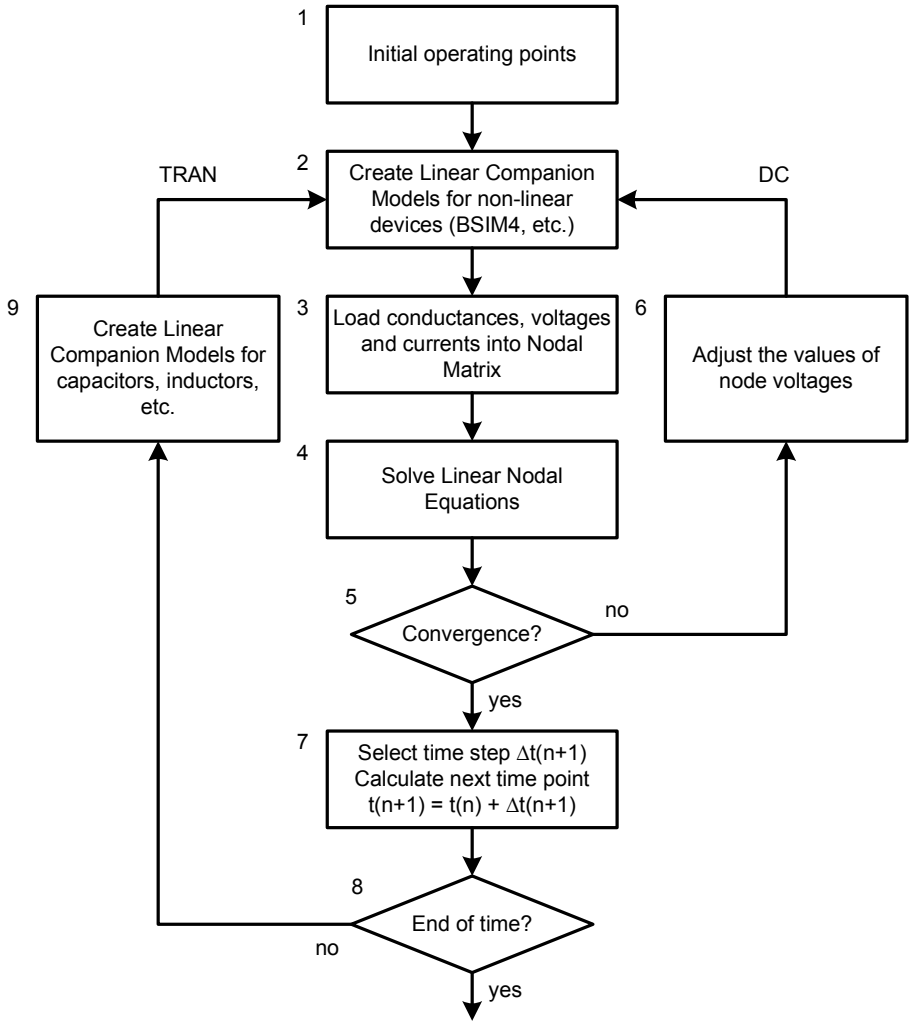


Figure A.1: SPICE simulation flow [85, 83, 146].

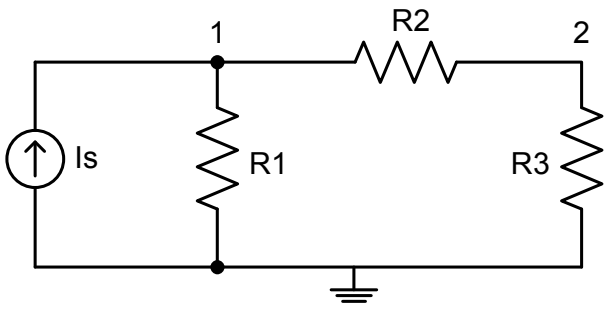


Figure A.2: Example circuit for nodal analysis.

SPICE employs advanced convergence algorithms to achieve results within the specified accuracy tolerance and sophisticated semiconductor device models such as BSIM3 and BSIM4 MOSFET models [86] to accurately simulate the device behaviour in a circuit. The common SPICE simulation includes DC (steady-state analysis), TRAN (transient analysis) and AC (frequency domains) simulation. This thesis uses DC and TRAN simulation to run logic and delay fault simulations respectively, therefore a brief introduction of how SPICE work in DC and TRAN simulation is presented. A simplified SPICE simulation flow is shown in Figure A.1, which includes 9 steps. Step-1 in Figure A.1 initializes the operating points on the circuit nodes by setting the voltage values on the circuit nodes. Step-2 creates linear companion models using device models such as BSIM4 transistor models for non-linear devices (capacitors, inductors, CMOS transistors). Step-3 and step-4 are the main steps in SPICE simulation, which create and solve the nodal matrix of conductances, voltages and currents for the simulated circuit. As an example, two equations for the circuit nodes 1 and 2 shown in Figure A.2 by applying Kirchhoff's Current Law (the sum of currents in and out of a node is zero) are shown in Eq. A.1. The aim of this example is to calculate the node voltages, therefore Eq. A.1 can be rearranged as Eq. A.2. Now the resistors can be rewritten in term of total conductances: $G_{11} = 1/R_1 + 1/R_2$, $G_{12} = -1/R_2$, $G_{21} = -1/R_1$, $G_{22} = 1/R_2 + 1/R_3$, $I_1 = I_s$ and $I_2 = 0$. Therefore Eq. A.2 is represented in term of Matrix form, as shown in Eq. A.3. Each matrix shown in Eq. A.3 can be treated as a single variable. Let $\mathbf{G}$ represents the conductances, $\mathbf{v}$ represents the voltages and $\mathbf{i}$ represents the currents. Eq. A.3 becomes $\mathbf{G} \cdot \mathbf{v} = \mathbf{i}$, which is what step-3 does in SPICE to create Nodal Matrix. Step-4 uses the matrix calculation to solve the Nodal Matrix for $\mathbf{v}$: $\mathbf{v} = \mathbf{G}^{-1} \cdot \mathbf{i}$. The solution of the Nodal Matrix is checked through convergence algorithm to achieve results within the specified accuracy tolerances in step-5. If the results does not converge, step-6 will adjust the values of node voltages and the flow goes back to step-2. It may take many iterations before the results converge to a solution. Otherwise step-7 selects time step $\Delta t(n+1)$ for the simulation and calculates next time point $t(n+1)$ by adding time step $\Delta t(n+1)$ to the current time point $t(n)$. Step-8 is used to check if $t(n)$ has reached the end of simulation time. If not, step-9 recreate linear companion models such as capacitors or inductors due to timing effect and the flow goes back to step-2. Otherwise simulation finishes. The inner loop (step-2 to step-6) is normally what SPICE

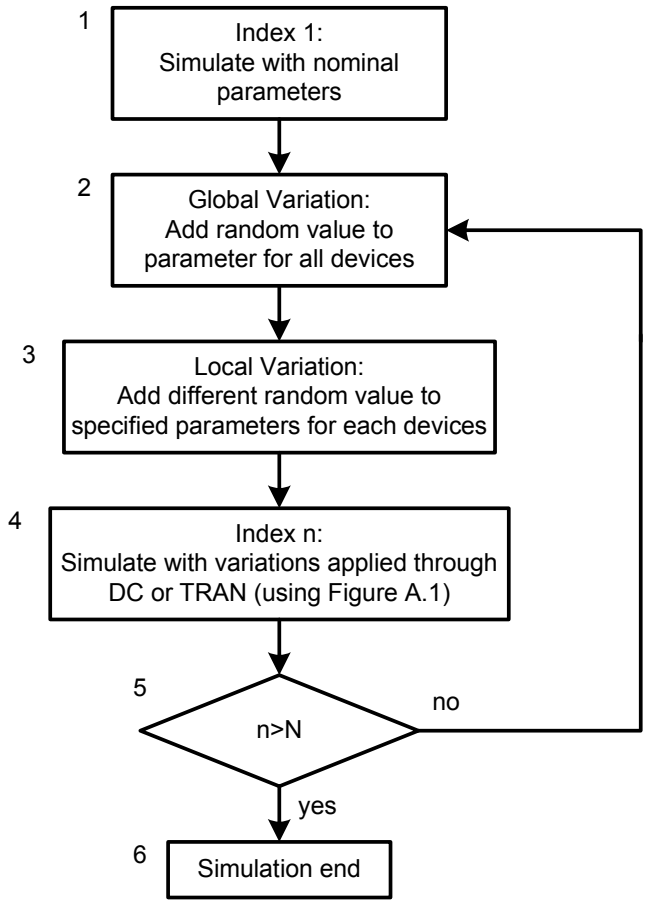


Figure A.3: Flow of Monte-Carlo analysis [83].

does in DC simulation and the outer loop (step-7 to step-9) together with the inner loop performs a TRAN simulation.

$$\begin{cases} -I_s + \frac{V_1}{R_1} + \frac{V_1 - V_2}{R_2} = 0 \\ \frac{V_2 - V_1}{R_2} + \frac{V_2}{R_3} = 0 \end{cases} \tag{A.1}$$

$$\begin{cases} \left(\frac{1}{R_1} + \frac{1}{R_2}\right) \cdot V_1 + \left(-\frac{1}{R_2}\right) \cdot V_2 = I_s \\ \left(-\frac{1}{R_2}\right) \cdot V_1 + \left(\frac{1}{R_2} + \frac{1}{R_3}\right) \cdot V_2 = 0 \end{cases} \tag{A.2}$$

$$\begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \times \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} I_1 \\ I_2 \end{bmatrix} \tag{A.3}$$

Process variation can be modelled through Monte-Carlo analysis in SPICE. The flow of Monte-Carlo analysis is shown in Figure A.3, which has 6 steps. The first sample (step-1) in Monte-Carlo analysis is executed with nominal values without variation. A sample is a set of variation parameter values. The subsequent samples (N-1 samples) are analysed from step-2 to step-5. Step-2 updates the global parameter such as supply

voltage or temperature with the same random value for all devices. Step-3 changes the local parameter with different random values for each device. The local parameter can be transistor gate length (L) or threshold voltage (V_{th}). The random values generated for parameter variation follow specific distributions (i.e. Gaussian distribution as shown in Eq. 1.1). Step-4 applies the generated random values to the circuit and runs simulation through SPICE using the flow shown in Figure A.1. Step-5 checks if the analysed samples (n) reached the total number of samples (N). If not, the flow goes back to step-2 and generates new sample. Otherwise the simulation stops.

Appendix B

ST Microelectronics 65-nm Gate Library

All the results reported in Chapter 3, Chapter 4 and Chapter 5 used ST Microelectronics 65-nm gate library. In the following, SPICE description of three gates (Inverter, NAND, NOR) are presented to provide detailed information.

```
* Inverter
.SUBCKT HS65_LS_IVX2 A Z gnd gnds vdd vdds
Xld_D0 gnds vdds DNWPS AREA=5.20365 PJ=9.53
XMM64 Z A gnd gnds NSVTLP AD=0.041 AS=0.041 L=0.06 PD=0.61 PS=0.61 W=0.2
XMM65 Z A vdd vdds PSVTLP AD=0.0574 AS=0.0574 L=0.06 PD=0.69 PS=0.69 W=0.28
C14 vdd A 3.93903e-17
C16 Z vdd 4.18569e-17
C17 Z gnd 1.73068e-16
C18 Z gnds 2.01431e-17
C19 Z A 1.39162e-16
C2 gnds A 2.78265e-17
C5 vdds A 7.89036e-18
C9 gnd A 9.15628e-17
Cg1 A 0 2.17049e-17
Cg10 gnd 0 1.19838e-16
Cg15 vdd 0 4.46402e-18
Cg20 Z 0 2.14667e-17
Cg3 gnds 0 5.41575e-17
Cg6 vdds 0 8.14917e-17
.ENDS HS65_LS_IVX2
```

```
* Two Input NAND Gate
.SUBCKT HS65_LS_NAND2X2 A B Z gnd gnds vdd vdds
Xld_D0 gnds vdds DNWPS AREA=5.54265 PJ=9.93
XMM64 net247 A gnd gnds NSVTLP AD=0.016 AS=0.039 L=0.06 PD=0.16 PS=0.59 W=0.2
XMM65 Z A vdd vdds PSVTLP AD=0.028 AS=0.049 L=0.06 PD=0.2 PS=0.63 W=0.28
XMM66 Z B vdd vdds PSVTLP L=0.06 W=0.28 ad=0.028 as=0.049 pd=0.2 ps=0.63
XMM67 Z B net247 gnds NSVTLP AD=0.039 AS=0.016 L=0.06 PD=0.59 PS=0.16 W=0.2
C11 gnd B 5.96568e-17
C12 gnd A 9.24749e-17
C14 Z gnd 1.48063e-16
C15 Z gnds 1.34414e-17
C16 Z B 1.564e-16
C17 Z A 1.12455e-16
C19 vdd Z 6.83136e-17
C2 B A 9.75336e-17
C23 vdd B 4.62538e-17
C24 vdd A 4.50046e-17
C26 net247 Z 2.64938e-18
C27 net247 gnd 1.86573e-18
C28 net247 B 1.29523e-19
C29 net247 A 1.29523e-19
C4 gnds B 1.5842e-17
C7 vdds A 6.31068e-18
Cg1 A 0 1.12452e-17
Cg13 gnd 0 1.19185e-16
Cg18 Z 0 7.15787e-18
Cg25 vdd 0 9.22087e-18
Cg3 B 0 2.00023e-17
Cg5 gnds 0 4.54605e-17
Cg8 vdds 0 7.84185e-17
.ENDS HS65_LS_NAND2X2
```

```
* Two Input NOR Gate
.SUBCKT HS65_LS_NOR2X2 A B Z gnd gnds vdd vdds
Xld_D0 gnds vdds DNWPS AREA=5.54265 PJ=9.93
XMM64 Z B gnd gnds NSVTLP L=0.06 W=0.2 ad=0.02 as=0.035 pd=0.2 ps=0.55
XMM65 net56 A vdd vdds PSVTLP AD=0.02975 AS=0.18305 L=0.06 PD=0.17 PS=2.31 W=0.35
XMM66 Z B net56 vdds PSVTLP AD=0.0665 AS=0.02975 L=0.06 PD=0.73 PS=0.17 W=0.35
XMM67 Z A gnd gnds NSVTLP AD=0.02 AS=0.035 L=0.06 PD=0.2 PS=0.55 W=0.2
C13 vdds A 9.93039e-18
C15 Z gnds 1.42954e-17
C16 Z vdd 1.594e-17
C17 Z B 1.59072e-16
C18 Z A 1.12618e-16
C2 B A 8.46331e-17
C20 gnd Z 1.59892e-16
C24 gnd B 1.20729e-16
C25 gnd A 9.8702e-17
C27 net56 gnd 8.10078e-19
C28 net56 Z 3.59377e-18
C29 net56 vdd 1.75908e-18
C4 vdd B 4.9915e-17
C5 vdd A 3.39757e-17
C8 gnds B 2.94526e-17
C9 gnds A 2.33945e-17
Cg1 A 0 1.75784e-17
Cg10 gnds 0 5.41732e-17
Cg14 vdds 0 7.76236e-17
Cg19 Z 0 9.14706e-18
Cg26 gnd 0 1.20968e-16
Cg3 B 0 2.12266e-17
Cg6 vdd 0 9.00591e-18
.ENDS HS65_LS_NOR2X2
```

Appendix C

PTM Transistor Model Card

All the results reported in Chapter 3, Chapter 4 and Chapter 5 used PTM Transistor Model. In the following, SPICE parameter description of NMOS and PMOS transistor model card in 65-nm and 45-nm technology is presented to provide detailed information. Further information can be found from [21].

```
* PTM 65nm NMOS

.model   nmos   nmos   level = 54

+version = 4.0         binunit = 1          paramchk= 1          mobmod   = 0
+capmod  = 2           igcmod  = 1          igbmod  = 1          geomod   = 1
+diomod  = 1           rdsmod  = 0          rbodymod= 1          rgatemod= 1
+permod  = 1           acnqsmod= 0          trnqsmod= 0

+tnom    = 27          toxe    = 1.85e-9    toxp    = 1.2e-9     toxm     = 1.85e-9
+dtox    = 0.65e-9     epsrox  = 3.9        wint    = 5e-009     lint     = 5.25e-009
+ll      = 0           wl      = 0          lln     = 1          wln      = 1
+lw      = 0           ww      = 0          lwn     = 1          wwn      = 1
+lwl     = 0           wwl     = 0          xpart   = 0          toxref   = 1.85e-9
+xl      = -30e-9
+vth0    = 0.423       k1      = 0.4        k2      = 0.01       k3       = 0
+k3b     = 0           w0      = 2.5e-006   dvt0    = 1          dvt1     = 2
+dvt2    = -0.032      dvt0w   = 0          dvt1w   = 0          dvt2w    = 0
+dsub    = 0.1         minv    = 0.05       voffl   = 0          dvtp0    = 1.0e-009
+dvtp1   = 0.1         lpe0    = 0          lpeb    = 0          xj       = 1.96e-008
+ngate   = 2e+020      ndep    = 2.54e+018  nsd     = 2e+020     phin     = 0
+cdsc    = 0.000       cdscb   = 0          cdscd   = 0          cit      = 0
+voff    = -0.13       nfactor = 1.9        eta0    = 0.0058     etab     = 0
+vfb     = -0.55       u0      = 0.0491     ua      = 6e-010     ub       = 1.2e-018
+uc      = 0           vsat    = 124340     a0      = 1.0        ags      = 1e-020
+a1      = 0           a2      = 1.0        b0      = 0          b1       = 0
+keta    = 0.04        dwg     = 0          dwb     = 0          pclm     = 0.04
+pdiblc1 = 0.001       pdiblc2 = 0.001      pdiblcb = -0.005     drout    = 0.5
+pvag    = 1e-020      delta   = 0.01       pscbe1  = 8.14e+008  pscbe2   = 1e-007
+fprout  = 0.2         pdits   = 0.08       pditsd  = 0.23       pditsl   = 2.3e+006
+rsh     = 5           rdsw    = 165        rsw     = 85         rdw      = 85
+rdswmin = 0           rdwmin  = 0          rswmin  = 0          prwg     = 0
+prwb    = 6.8e-011    wr      = 1          alpha0  = 0.074      alpha1   = 0.005
```

```
+beta0    = 30         agidl    = 0.0002    bgidl    = 2.1e+009  cgidl    = 0.0002
+egidl    = 0.8

+aigbacc  = 0.012      bigbacc  = 0.0028    cigbacc  = 0.002
+nigbacc  = 1          aigbinv  = 0.014     bigbinv  = 0.004     cigbinv  = 0.004
+eigbinv  = 1.1        nigbinv  = 3         aigc     = 0.012     bigc     = 0.0028
+cigc     = 0.002      aigsd    = 0.012     bigsd    = 0.0028    cigsd    = 0.002
+nigc     = 1          poxedge  = 1         pigcd    = 1         ntox     = 1

+xrcrg1   = 12         xrcrg2   = 5
+cgso     = 1.5e-010   cgdo     = 1.5e-010  cgbo     = 2.56e-011 cgdl     = 2.653e-10
+cgsl     = 2.653e-10  ckappas  = 0.03      ckappad  = 0.03      acde     = 1
+moin     = 15         noff     = 0.9       voffcv   = 0.02

+kt1      = -0.11      kt1l     = 0         kt2      = 0.022     ute      = -1.5
+ua1      = 4.31e-009  ub1      = 7.61e-018 uc1      = -5.6e-011 prt      = 0
+at       = 33000

+fnoimod  = 1          tnoimod  = 0

+jss      = 0.0001     jsws     = 1e-011    jswgs    = 1e-010    njs      = 1
+ijthsfwd= 0.01       ijthsrev= 0.001      bvs      = 10        xjbvs    = 1
+jsd      = 0.0001     jswd     = 1e-011    jswgd    = 1e-010    njd      = 1
+ijthdfwd= 0.01       ijthdrev= 0.001      bvd      = 10        xjbvd    = 1
+pbs      = 1          cjs      = 0.0005    mjs      = 0.5       pbsws    = 1
+cjsws    = 5e-010     mjsws    = 0.33      pbswgs   = 1         cjswgs   = 3e-010
+mjswgs   = 0.33       pbd      = 1         cjd      = 0.0005    mjd      = 0.5
+pbswd    = 1          cjswd    = 5e-010    mjswd    = 0.33      pbswgd   = 1
+cjswgd   = 5e-010     mjswgd   = 0.33      tpb      = 0.005     tcj      = 0.001
+tpbsw    = 0.005      tcjsw    = 0.001     tpbswg   = 0.005     tcjswg   = 0.001
+xtis     = 3          xtid     = 3

+dmcg     = 0e-006     dmci     = 0e-006    dmdg     = 0e-006    dmcgt    = 0e-007
+dwj      = 0.0e-008   xgw      = 0e-007    xgl      = 0e-008

+rshg     = 0.4        gbmin    = 1e-010    rbpb     = 5         rbpd     = 15
+rbps     = 15         rbdb     = 15        rbsb     = 15        ngcon    = 1



* PTM 65nm PMOS

.model   pmos   pmos   level = 54

+version  = 4.0        binunit  = 1         paramchk= 1         mobmod   = 0
+capmod   = 2          igcmod   = 1         igbmod   = 1         geomod   = 1
+diomod   = 1          rdsmod   = 0         rbodymod= 1         rgatemod= 1
+permod   = 1          acnqsmod= 0          trnqsmod= 0

+tnom     = 27         toxe     = 1.95e-009 toxp     = 1.2e-009  toxm     = 1.95e-009
+dtox     = 0.75e-9    epsrox   = 3.9       wint     = 5e-009    lint     = 5.25e-009
+ll       = 0          wl       = 0         lln      = 1         wln      = 1
+lw       = 0          ww       = 0         lwn      = 1         wwn      = 1
+lwl      = 0          wwl      = 0         xpart    = 0         toxref   = 1.95e-009
+xl       = -30e-9
+vth0     = -0.365     k1       = 0.4       k2       = -0.01     k3       = 0
+k3b      = 0          w0       = 2.5e-006  dvt0     = 1         dvt1     = 2
+dvt2     = -0.032     dvt0w    = 0         dvt1w    = 0         dvt2w    = 0
```

```
+dsub     = 0.1        minv     = 0.05       voffl    = 0          dvtp0    = 1e-009
+dvtp1    = 0.05       lpe0     = 0          lpeb     = 0          xj       = 1.96e-008
+ngate    = 2e+020     ndep     = 1.87e+018  nsd      = 2e+020     phin     = 0
+cdsc     = 0.000      cdscb    = 0          cdscd    = 0          cit      = 0
+voff     = -0.126     nfactor  = 1.9        eta0     = 0.0058     etab     = 0
+vfb      = 0.55       u0       = 0.00574    ua       = 2.0e-009   ub       = 0.5e-018
+uc       = 0          vsat     = 70000      a0       = 1.0        ags      = 1e-020
+a1       = 0          a2       = 1          b0       = -1e-020    b1       = 0
+keta     = -0.047     dwg      = 0          dwb      = 0          pclm     = 0.12
+pdiblc1  = 0.001      pdiblc2  = 0.001      pdiblcb  = 3.4e-008   drout    = 0.56
+pvag     = 1e-020     delta    = 0.01       pscbe1   = 8.14e+008  pscbe2   = 9.58e-007
+fprout   = 0.2        pdits    = 0.08       pditsd   = 0.23       pditsl   = 2.3e+006
+rsh      = 5          rdsw     = 165        rsw      = 85         rdw      = 85
+rdswmin  = 0          rdwmin   = 0          rswmin   = 0          prwg     = 3.22e-008
+prwb     = 6.8e-011   wr       = 1          alpha0   = 0.074      alpha1   = 0.005
+beta0    = 30         agidl    = 0.0002     bgidl    = 2.1e+009   cgidl    = 0.0002
+egidl    = 0.8

+aigbacc  = 0.012      bigbacc  = 0.0028     cigbacc  = 0.002
+nigbacc  = 1          aigbinv  = 0.014      bigbinv  = 0.004      cigbinv  = 0.004
+eigbinv  = 1.1        nigbinv  = 3          aigc     = 0.69       bigc     = 0.0012
+cigc     = 0.0008     aigsd    = 0.0087     bigsd    = 0.0012     cigsd    = 0.0008
+nigc     = 1          poxedge  = 1          pigcd    = 1          ntox     = 1

+xrcrg1   = 12         xrcrg2   = 5
+cgso     = 1.5e-010   cgdo     = 1.5e-010   cgbo     = 2.56e-011  cgdl     = 2.653e-10
+cgsl     = 2.653e-10  ckappas  = 0.03       ckappad  = 0.03       acde     = 1
+moin     = 15         noff     = 0.9        voffcv   = 0.02

+kt1      = -0.11      kt1l     = 0          kt2      = 0.022      ute      = -1.5
+ua1      = 4.31e-009  ub1      = 7.61e-018  uc1      = -5.6e-011  prt      = 0
+at       = 33000

+fnoimod  = 1          tnoimod  = 0

+jss      = 0.0001     jsws     = 1e-011     jswgs    = 1e-010     njs      = 1
+ijthsfwd= 0.01       ijthsrev= 0.001      bvs      = 10         xjbvs    = 1
+jsd      = 0.0001     jswd     = 1e-011     jswgd    = 1e-010     njd      = 1
+ijthdfwd= 0.01       ijthdrev= 0.001      bvd      = 10         xjbvd    = 1
+pbs      = 1          cjs      = 0.0005     mjs      = 0.5        pbsws    = 1
+cjsws    = 5e-010     mjsws    = 0.33       pbswgs   = 1          cjswgs   = 3e-010
+mjswgs   = 0.33       pbd      = 1          cjd      = 0.0005     mjd      = 0.5
+pbswd    = 1          cjswd    = 5e-010     mjswd    = 0.33       pbswgd   = 1
+cjswgd   = 5e-010     mjswgd   = 0.33       tpb      = 0.005      tcj      = 0.001
+tpbsw    = 0.005      tcjsw    = 0.001      tpbswg   = 0.005      tcjswg   = 0.001
+xtis     = 3          xtid     = 3

+dmcg     = 0e-006     dmci     = 0e-006     dmdg     = 0e-006     dmcgt    = 0e-007
+dwj      = 0.0e-008   xgw      = 0e-007     xgl      = 0e-008

+rshg     = 0.4        gbmin    = 1e-010     rbpb     = 5          rbpd     = 15
+rbps     = 15         rbdb     = 15         rbsb     = 15         ngcon    = 1
```

```
* 45nm NMOS

.model  NMOS_VTL  nmos  level = 54

+version = 4.0    binunit = 1    paramchk= 1    mobmod  = 0
+capmod  = 2      igcmod  = 1    igbmod  = 1    geomod  = 1
+diomod  = 1      rdsmod  = 0    rbodymod= 1    rgatemod= 1
+permod  = 1      acnqsmod= 0    trnqsmod= 0

* parameters related to the technology node
+tnom = 27        epsrox = 3.9
+eta0 = 0.0049    nfactor = 2.1     wint = 5e-09
+cgso = 1.1e-10   cgdo = 1.1e-10    xl = -2e-08

* parameters customized by the user
+toxe = 1.75e-09   toxp = 1.1e-09    toxm = 1.75e-09    toxref = 1.75e-09
+dtox = 6.5e-10    lint = 3.75e-09
+vth0 = 0.471      k1 = 0.53         u0 = 0.04359       vsat = 147390
+rdsw = 155        ndep = 3.3e+18    xj = 1.4e-08

* secondary parameters
+ll      = 0          wl      = 0          lln     = 1          wln     = 1
+lw      = 0          ww      = 0          lwn     = 1          wwn     = 1
+lwl     = 0          wwl     = 0          xpart   = 0
+k2      = 0.01       k3      = 0
+k3b     = 0          w0      = 2.5e-006   dvt0    = 1          dvt1    = 2
+dvt2    = -0.032     dvt0w   = 0          dvt1w   = 0          dvt2w   = 0
+dsub    = 0.1        minv    = 0.05       voffl   = 0          dvtp0   = 1.0e-009
+dvtp1   = 0.1        lpe0    = 0          lpeb    = 0
+ngate   = 2e+020     nsd     = 2e+020     phin    = 0
+cdsc    = 0.000      cdscb   = 0          cdscd   = 0          cit     = 0
+voff    = -0.13      etab    = 0
+vfb     = -0.55      ua      = 6e-010     ub      = 1.2e-018
+uc      = 0          a0      = 1.0        ags     = 1e-020
+a1      = 0          a2      = 1.0        b0      = 0          b1      = 0
+keta    = 0.04       dwg     = 0          dwb     = 0          pclm    = 0.04
+pdiblc1 = 0.001      pdiblc2 = 0.001      pdiblcb = -0.005     drout   = 0.5
+pvag    = 1e-020     delta   = 0.01       pscbe1  = 8.14e+008  pscbe2  = 1e-007
+fprout  = 0.2        pdits   = 0.08       pditsd  = 0.23       pditsl  = 2.3e+006
+rsh     = 5          rsw     = 85         rdw     = 85
+rdswmin = 0          rdwmin  = 0          rswmin  = 0          prwg    = 0
+prwb    = 6.8e-011   wr      = 1          alpha0  = 0.074      alpha1  = 0.005
+beta0   = 30         agidl   = 0.0002     bgidl   = 2.1e+009   cgidl   = 0.0002
+egidl   = 0.8

+aigbacc = 0.012      bigbacc = 0.0028     cigbacc = 0.002
+nigbacc = 1          aigbinv = 0.014      bigbinv = 0.004      cigbinv = 0.004
+eigbinv = 1.1        nigbinv = 3          aigc    = 0.012      bigc    = 0.0028
+cigc    = 0.002      aigsd   = 0.012      bigsd   = 0.0028     cigsd   = 0.002
+nigc    = 1          poxedge = 1          pigcd   = 1          ntox    = 1

+xrcrg1  = 12         xrcrg2  = 5
+cgbo    = 2.56e-011  cgdl    = 2.653e-10
+cgsl    = 2.653e-10  ckappas = 0.03       ckappad = 0.03       acde    = 1
+moin    = 15         noff    = 0.9        voffcv  = 0.02

+kt1     = -0.11      kt1l    = 0          kt2     = 0.022      ute     = -1.5
+ua1     = 4.31e-009  ub1     = 7.61e-018  uc1     = -5.6e-011  prt     = 0
+at      = 33000
```

```
+fnoimod = 1          tnoimod = 0

+jss     = 0.0001     jsws    = 1e-011    jswgs   = 1e-010    njs     = 1
+ijthsfwd= 0.01       ijthsrev= 0.001     bvs     = 10         xjbvs   = 1
+jsd     = 0.0001     jswd    = 1e-011    jswgd   = 1e-010    njd     = 1
+ijthdfwd= 0.01       ijthdrev= 0.001     bvd     = 10         xjbvd   = 1
+pbs     = 1          cjs     = 0.0005    mjs     = 0.5        pbsws   = 1
+cjsws   = 5e-010     mjsws   = 0.33      pbswgs  = 1          cjswgs  = 3e-010
+mjswgs  = 0.33       pbd     = 1         cjd     = 0.0005     mjd     = 0.5
+pbswd   = 1          cjswd   = 5e-010    mjswd   = 0.33       pbswgd  = 1
+cjswgd  = 5e-010     mjswgd  = 0.33      tpb     = 0.005      tcj     = 0.001
+tpbsw   = 0.005      tcjsw   = 0.001     tpbswg  = 0.005      tcjswg  = 0.001
+xtis    = 3          xtid    = 3

+dmcg    = 0e-006     dmci    = 0e-006    dmdg    = 0e-006     dmcgt   = 0e-007
+dwj     = 0.0e-008   xgw     = 0e-007    xgl     = 0e-008

+rshg    = 0.4        gbmin   = 1e-010    rbpb    = 5          rbpd    = 15
+rbps    = 15         rbdb    = 15        rbsb    = 15         ngcon   = 1


* 45nm PMOS

.model  PMOS_VTL  pmos  level = 54

+version = 4.0    binunit = 1    paramchk= 1    mobmod  = 0
+capmod  = 2      igcmod  = 1    igbmod  = 1    geomod  = 1
+diomod  = 1      rdsmod  = 0    rbodymod= 1    rgatemod= 1
+permod  = 1      acnqsmod= 0    trnqsmod= 0

* parameters related to the technology node
+tnom = 27         epsrox = 3.9
+eta0 = 0.0049     nfactor = 2.1      wint = 5e-09
+cgso = 1.1e-10    cgdo = 1.1e-10     xl = -2e-08

* parameters customized by the user
+toxe = 1.85e-09    toxp = 1.1e-09     toxm = 1.85e-09     toxref = 1.85e-09
+dtox = 7.5e-10     lint = 3.75e-09
+vth0 = -0.423      k1 = 0.491         u0 = 0.00432        vsat = 70000
+rdsw = 155         ndep = 2.54e+18    xj = 1.4e-08

*secondary parameters
+ll      = 0          wl      = 0          lln     = 1          wln     = 1
+lw      = 0          ww      = 0          lwn     = 1          wwn     = 1
+lwl     = 0          wwl     = 0          xpart   = 0
+k2      = -0.01      k3      = 0
+k3b     = 0          w0      = 2.5e-006   dvt0    = 1          dvt1    = 2
+dvt2    = -0.032     dvt0w   = 0          dvt1w   = 0          dvt2w   = 0
+dsub    = 0.1        minv    = 0.05       voffl   = 0          dvtp0   = 1e-009
+dvtp1   = 0.05       lpe0    = 0          lpeb    = 0
+ngate   = 2e+020     nsd     = 2e+020     phin    = 0
+cdsc    = 0.000      cdscb   = 0          cdscd   = 0          cit     = 0
+voff    = -0.126     etab    = 0
+vfb     = 0.55       ua      = 2.0e-009   ub      = 0.5e-018
+uc      = 0          a0      = 1.0        ags     = 1e-020
+a1      = 0          a2      = 1          b0      = -1e-020    b1      = 0
+keta    = -0.047     dwg     = 0          dwb     = 0          pclm    = 0.12
+pdiblc1 = 0.001      pdiblc2 = 0.001      pdiblcb = 3.4e-008   drout   = 0.56
```

```
+pvag    = 1e-020    delta   = 0.01      pscbe1  = 8.14e+008 pscbe2  = 9.58e-007
+fprout  = 0.2       pdits   = 0.08      pditsd  = 0.23      pditsl  = 2.3e+006
+rsh     = 5         rsw     = 85        rdw     = 85
+rdswmin = 0         rdwmin  = 0         rswmin  = 0         prwg    = 3.22e-008
+prwb    = 6.8e-011  wr      = 1         alpha0  = 0.074     alpha1  = 0.005
+beta0   = 30        agidl   = 0.0002    bgidl   = 2.1e+009  cgidl   = 0.0002
+egidl   = 0.8

+aigbacc = 0.012     bigbacc = 0.0028    cigbacc = 0.002
+nigbacc = 1         aigbinv = 0.014     bigbinv = 0.004     cigbinv = 0.004
+eigbinv = 1.1       nigbinv = 3         aigc    = 0.69      bigc    = 0.0012
+cigc    = 0.0008    aigsd   = 0.0087    bigsd   = 0.0012    cigsd   = 0.0008
+nigc    = 1         poxedge = 1         pigcd   = 1         ntox    = 1

+xrcrg1  = 12        xrcrg2  = 5
+cgbo    = 2.56e-011 cgdl    = 2.653e-10
+cgsl    = 2.653e-10 ckappas = 0.03      ckappad = 0.03      acde    = 1
+moin    = 15        noff    = 0.9       voffcv  = 0.02

+kt1     = -0.11     kt1l    = 0         kt2     = 0.022     ute     = -1.5
+ua1     = 4.31e-009 ub1     = 7.61e-018 uc1     = -5.6e-011 prt     = 0
+at      = 33000

+fnoimod = 1         tnoimod = 0

+jss     = 0.0001    jsws    = 1e-011    jswgs   = 1e-010    njs     = 1
+ijthsfwd= 0.01      ijthsrev= 0.001     bvs     = 10        xjbvs   = 1
+jsd     = 0.0001    jswd    = 1e-011    jswgd   = 1e-010    njd     = 1
+ijthdfwd= 0.01      ijthdrev= 0.001     bvd     = 10        xjbvd   = 1
+pbs     = 1         cjs     = 0.0005    mjs     = 0.5       pbsws   = 1
+cjsws   = 5e-010    mjsws   = 0.33      pbswgs  = 1         cjswgs  = 3e-010
+mjswgs  = 0.33      pbd     = 1         cjd     = 0.0005    mjd     = 0.5
+pbswd   = 1         cjswd   = 5e-010    mjswd   = 0.33      pbswgd  = 1
+cjswgd  = 5e-010    mjswgd  = 0.33      tpb     = 0.005     tcj     = 0.001
+tpbsw   = 0.005     tcjsw   = 0.001     tpbswg  = 0.005     tcjswg  = 0.001
+xtis    = 3         xtid    = 3

+dmcg    = 0e-006    dmci    = 0e-006    dmdg    = 0e-006    dmcgt   = 0e-007
+dwj     = 0.0e-008  xgw     = 0e-007    xgl     = 0e-008

+rshg    = 0.4       gbmin   = 1e-010    rbpb    = 5         rbpd    = 15
+rbps    = 15        rbdb    = 15        rbsb    = 15        ngcon   = 1
```

Appendix D

Eliminated Transistor Parameters

This appendix expands the details of transistor electrical parameter elimination that are shown in Section 4.3.1.2, Chapter 4 to show all the eliminated parameters from transistor sub-models shown in Table 4.1. The removed parameters are shown in Table D.1. Table D.1 shows that it is possible to remove 30 out of 380 (8%) parameters from Drain Current Model, 20 out of 80 (25%) from Channel Charge and Subthreshold Swing Model, 9 out of 90 (10%) from Gate Direct Tunneling Current Model and 79 out of 394 (20%) from Capacitance Model.

Table D.1: Removed parameters from transistor models.

Transistor Models	Removed Parameters
Drain Current	Voffcv, VgstNVt, Vges, Vged, Vgms, Vfbcv, icVGS, icVBS, icVDS, Qcheq, Qchqs, RDW, RSW, PRWB, dRs_dvg, dRd_dvg, dRs_dvb, dRd_dvb, gstot, gstotd, gstotg, gstots, gstotb, Voxdepinv gdtot, gdtotd, gdtotg, gdtots, gdtotb, GBMIN
Channel Charge & Subthreshold Swing	Voffcv, NOFF, CDSCD, CDSCB, Qcheq, Qchqs, dVgsteff_dVg, dVgsteff_dVd, dVgsteff_dVb, dnoff_dVb, dVdseffCV_dVg, dVdseffCV_dVd, dVdseffCV_dVb, dQac0_dVg, dQac0_dVb, dQsub0_dVg, dQsub0_dVd, dQsub0_dVb
Gate Direct Tunneling Current	Igc, Igcs, Igcd, PIGCD, DLCIG, DLCIGD, NTOX, VFBSDOFF, EIGBINV
Capacitance	Voffcv, NOFF, Vfbcv, CGSL, CGDL, VOFFCVL, MINVCV, Vges, Vged, Vgms, AbulkCV, abulkCVfactor, Voxdepinv, dVgsteff_dVg, dVgsteff_dVd, dVgsteff_dVb, dnoff_dVb, dVoxdepinv_dVg, dVoxdepinv_dVd, dVoxdepinv_dVb, dIgc_dVg, dIgc_dVd, dIgc_dVb, dIgcs_dVg, dIgcs_dVd, dIgcs_dVb, dIgcd_dVg, dIgcd_dVd, dIgcd_dVb, dPigcd_dVb, dVdseffCV_dVg, dVdseffCV_dVd, dVdseffCV_dVb dIgs_dVg, dIgs_dVs, dIgd_dVg, dIgd_dVd, dIgbacc_dVg, dIgbacc_dVd, dIgbacc_dVb, dIgbinv_dVg, dIgbinv_dVd, dIgbinv_dVb, dIgb_dVg, dIgb_dVd, dIgb_dVb gstot, gstotd, gstotg, gstots, gstotb, gdtot, gdtotd, gdtotg, gdtots, gdtotb, vses, vdes, vdedo, delvses, delvded, delvdes, ceqqb, ceqqd, ceqqg, ceqqjd, ceqqjs, ceq, geq, dVASCBE_dVg, dVASCBE_dVd, dVASCBE_dVb, dVADIBL_dVg, dVACLM_dVg, dVACLM_dVd, dVACLM_dVb, dVADITS_dVg, dVADIBL_dVd, dVADITS_dVd

References

[1] V. Iyengar, J. Xiong, S. Venkatesan, V. Zolotov, D. Lackey, P. Habitz, and C. Visweswariah. Variation-aware performance verification using at-speed structural test and statistical timing. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 405 –412, Nov. 2007.

[2] U. Ingelsson, B.M. Al-Hashimi, S. Khursheed, S.M. Reddy, and P. Harrod. Process variation-aware test for resistive bridges. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 28(8):1269–1274, Aug. 2009.

[3] U. Ingelsson. Investigation into voltage and process variation-aware manufacturing test. *Ph.D Thesis, University of Southampton*, July 2009.

[4] Gareth Roy, Andrew R. Brown, Fikru Adamu-Lema, Scott Roy, and Asen Asenov. Simulation study of individual and combined sources of intrinsic parameter fluctuations in conventional nano-mosfets. *IEEE Transactions on Electron Devices*, 53(12):3063–3070, Dec. 2006.

[5] T.M. Mak and S. Nassif. Guest editors' introduction: Process variation and stochastic design and test. *Design Test of Computers, IEEE*, 23(6):436 –437, June 2006.

[6] ITRS, The International Technology Roadmap for Semiconductor 2007.

[7] S.S. Majzoub, R.A. Saleh, S.J.E. Wilton, and R.K. Ward. Energy optimization for many-core platforms: Communication and pvt aware voltage-island formation and voltage selection algorithm. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 29(5):816 –829, May 2010.

[8] S.K. Mathew, S. Srinivasan, M.A. Anders, H. Kaul, S.K. Hsu, F. Sheikh, A. Agarwal, S. Satpathy, and R.K. Krishnamurthy. 2.4 gbps, 7 mw all-digital pvt-variation tolerant true random number generator for 45 nm cmos high-performance microprocessors. *Solid-State Circuits, IEEE Journal of*, 47(11):2807 –2821, Nov. 2012.

[9] I. A. K. M. Mahfuzul, A. Tsuchiya, K. Kobayashi, and H. Onodera. Variation-sensitive monitor circuits for estimation of global process parameter variation. *Semiconductor Manufacturing, IEEE Transactions on*, 25(4):571 –580, Nov. 2012.

[10] W. Zhao, F. Liu, K. Agarwal, D. Acharyya, S.R. Nassif, K.J. Nowka, and Y. Cao. Rigorous extraction of process variations for 65-nm CMOS design. *Semiconductor Manufacturing, IEEE Trans. on*, 22(1):196–203, Feb. 2009.

[11] K. Bernstein, D. J. Frank, A. E. Gattiker, W. Haensch, B. L. Ji, S. R. Nassif, E. J. Nowak, D. J. Pearson, and N. J. Rohrer. High-performance cmos variability in the 65-nm regime and beyond. *IBM Journal of Research and Development*, 50(4.5):433 –449, July 2006.

[12] D. Reid, C. Millar, G. Roy, S. Roy, and A. Asenov. Analysis of threshold voltage distribution due to random dopants: A 100,000-sample 3-d simulation study. *Electron Devices, IEEE Trans. on*, 56(10):2255 –2263, Oct. 2009.

[13] D. Reid, C. Millar, G. Roy, S. Roy, and A. Asenov. Understanding ler-induced statistical variability: A 35,000 sample 3d simulation study. In *Proceedings of the European Solid State Device Research Conference (ESSDERC '09)*, pages 423 –426, 14-18 Sept. 2009.

[14] S. Kundu and A. Sreedhar. Modeling manufacturing process variation for design and test. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, pages 1–6, March 2011.

[15] Kelin Kuhn et al. Managing process variation in intel's 45nm CMOS technology. *Intel Technology Journal*, 12, June 2008.

[16] Sandip Kundu, Aswin Sreedhar, and Alodeep Sanyal. Forbidden pitches in subwavelength lithography and their implications on design. *Journal of Computer-Aided Materials Design*, 14:79–89, 2007.

[17] P. Oldiges, Q. Lin, K. Petrillo, M. Sanchez, M. Ieong, and M. Hargrove. Modeling line edge roughness effects in sub 100 nanometer gate length devices. In *International Conference on Simulation of Semiconductor Processes and Devices (SISPAD 2000)*, pages 131 –134, 2000.

[18] S.R. Nassif. Modeling and analysis of manufacturing variations. In *IEEE Conference on Custom Integrated Circuits*, pages 223 –228, 2001.

[19] Swarup Bhunia, Saibal Mukhopadhyay, and Kaushik Roy. Process variations and process-tolerant design. In *International Conference on VLSI Design, Held jointly with 6th International Conference on Embedded Systems*, pages 699 –704, Jan. 2007.

[20] K.A. Bowman, A.R. Alameldeen, S.T. Srinivasan, and C.B. Wilkerson. Impact of die-to-die and within-die parameter variations on the clock frequency and throughput of multi-core processors. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 17(12):1679 –1690, Dec. 2009.

[21] PTM, Arizona State Univ., Tempe, Feb 2012. http://ptm.asu.edu.

[22] P. Friedberg, Y. Cao, J. Cain, R. Wang, J. Rabaey, and C. Spanos. Modeling within-die spatial correlation effects for process-design co-optimization. In *Quality of Electronic Design (ISQED), Sixth International Symposium on*, pages 516 – 521, March 2005.

[23] M. Orshansky, L. Milor, P. Chen, K. Keutzer, and C. Hu. Impact of systematic spatial intra-chip gate length variability on performance of high-speed digital circuits. In *IEEE/ACM International Conference on Computer Aided Design (ICCAD-2000)*, pages 62 –67, Nov. 2000.

[24] M. Choi and L. Milor. Impact on circuit performance of deterministic within-die variation in nanoscale semiconductor manufacturing. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 25(7):1350 –1367, July 2006.

[25] Yu Cao and L.T. Clark. Mapping statistical process variations toward circuit performance variability: An analytical modeling approach. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 26(10):1866 –1873, Oct. 2007.

[26] M. Abramovici, M.A. Breuer, and A.D. Friedman. *Digital Systems Testing and Testable Designs*. Jaico Publishing House, 2008.

[27] B. Kruseman, A. Majhi, C. Hora, S. Eichenberger, and J. Meirlevede. Systematic defects in deep sub-micron technologies. In *Proceedings of Test International Conference (ITC 2004)*, pages 290 – 299, Oct. 2004.

[28] R.R. Montanes, J.P. de Gyvez, and P. Volf. Resistance characterization for weak open defects. *Design Test of Computers, IEEE*, 19(5):18 – 26, Sep-Oct 2002.

[29] Xiang Lu, Zhuo Li, Wangqi Qiu, D.M.H. Walker, and Weiping Shi. Longest-path selection for delay test under process variation. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 24(12):1924 – 1929, Dec. 2005.

[30] V.R. Devanathan, C.P. Ravikumar, and V. Kamakoti. Variation-tolerant, power-safe pattern generation. *Design Test of Computers, IEEE*, 24(4):374 –384, July-Aug. 2007.

[31] Sandip Kundu, Sujit T. Zachariah, Sanjay Sengupta, and Rajesh Galivanche. Test challenges in nanometer technologies. *J. Electron. Test.*, 17(3-4):209–218, 2001.

[32] Hong Hao and E.J. McCluskey. "resistive shorts" within cmos gates. In *Proceedings of International Test Conference*, page 292, Oct 1991.

[33] S.K. Goel, N. Devta-Prasanna, and M. Ward. Comparing the effectiveness of deterministic bridge fault and multiple-detect stuck fault patterns for physical bridge defects: A simulation and silicon study. In *International Test Conference (ITC 2009)*, pages 1 –10, Nov. 2009.

[34] V. Krishnaswamy, A.B. Ma, and P. Vishakantaiah. A study of bridging defect probabilities on a pentium (tm) 4 CPU. In *Proceedings of International Test Conference*, pages 688 –695, 2001.

[35] Yukiya Miura and Shuichi Seno. Behavior analysis of internal feedback bridging faults in cmos circuits. *Journal of Electronic Testing*, 18:109–120, 2002. 10.1023/A:1014985307989.

[36] F.J. Ferguson and J.P. Shen. Extraction and simulation of realistic CMOS faults using inductive fault analysis. In *Test Conference, 1988. Proceedings. New Frontiers in Testing, International*, pages 475 –484, Sept. 1988.

[37] J.J.T. Sousa, F.M. Goncalves, and J.P. Teixeira. Ic defects-based testability analysis. In *Test Conference, 1991, Proceedings., International*, page 500, oct 1991.

[38] E. Isern and J. Figueras. Test generation with high coverages for quiescent current test of bridging faults in combinational circuits. In *Proceedings of International Test Conference*, pages 73 –82, Oct 1993.

[39] M. Roca and A. Rubio. Current testability analysis of feedback bridging faults in cmos circuits. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 14(10):1299 –1305, Oct 1995.

[40] R. Rodriguez-Montanes, E.M.J.G. Bruis, and J. Figueras. Bridging defects resistance measurements in a CMOS process. In *Proceedings of International Test Conference*, page 892, Sept. 1992.

[41] F. Hapke, W. Redemund, J. Schloeffel, R. Krenz-Baath, A. Glowatz, M. Wittke, H. Hashempour, and S. Eichenberger. Defect-oriented cell-internal testing. In *Test Conference (ITC), 2010 IEEE International*, pages 1 –10, Nov. 2010.

[42] S. Mandava, S. Chakravarty, and S. Kundu. On detecting bridges causing timing failures. In *Computer Design, International Conference on*, pages 400 –406, 1999.

[43] R. Rodriguez-Montanes and J. Figueras. Estimation of the defective IDDQ caused by shorts in deep-submicron CMOS ICs. In *Proceedings of Design, Automation and Test in Europe*, pages 490 –494, Feb 1998.

[44] T. Nakura, Y. Tatemura, G. Fey, M. Ikeda, S. Komatsu, and K. Asada. SAT-based ATPG testing of inter- and intra-gate bridging faults. In *European Conference on Circuit Theory and Design (ECCTD 2009)*, pages 643 –646, Aug. 2009.

[45] A. Fawaz, A. Jaber, A. Kassem, A. Chehab, and A. Kayssi. Assessing testing techniques for resistive-open defects in nanometer CMOS adders. In *Electronics, Circuits and Systems (ICECS), 2011 18th IEEE International Conference on*, pages 165 –168, Dec. 2011.

[46] Gary S. Greenstein and Janak H. Patel. E-PROOFS: a CMOS bridging fault simulator. In *ICCAD '92: Proceedings of the 1992 IEEE/ACM international conference on Computer-aided design*, pages 268–271, 1992.

[47] M. Renovell, P. Huc, and Y. Bertrand. The concept of resistance interval: a new parametric model for realistic resistive bridging fault. In *Proceedings of the VLSI Test Symposium (VTS)*, pages 184–189, April 1995.

[48] G. Chen, S. Reddy, I. Pomeranz, J. Rajski, P. Engelke, and B. Becker. A unified fault model and test generation procedure for interconnect opens and bridges. In *European Test Symposium 2005*, pages 22 – 27, 2005.

[49] P. Engelke, I. Polian, M. Renovell, and B. Becker. Simulating resistive bridging and stuck-at faults. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 25:2181–2192, October 2006.

[50] S. Khursheed, U. Ingelsson, P. Rosinger, B.M. Al-Hashimi, and P. Harrod. Bridging fault test method with adaptive power management awareness. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(6):1117 –1127, June 2008.

[51] Syed Saqib Khursheed and Bashir Al-Hashimi. Test strategies for multi-voltage designs. In Patrick Girard, Nicola Nicolici, and Xiaoqing Wen, editors, *Power-Aware Testing and Test Strategies for Low Power Devices*. Springer, November 2009. Chapter: 8.

[52] H. Konuk. Voltage- and current-based fault simulation for interconnect open defects. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 18(12):1768 –1779, Dec 1999.

[53] S. Rafiq, A. Ivanov, S. Tabatabaei, and M. Renovell. Testing for floating gates defects in CMOS circuits. In *Proceedings of Seventh Asian Test Symposium (ATS '98)*, pages 228 –236, 1998.

[54] M. Pronath, H. Graeb, and K. Antreich. A test design method for floating gate defects (FGD) in analog integrated circuits. In *Proceedings of Design, Automation and Test in Europe Conference and Exhibition*, pages 78 –83, 2002.

[55] James Chien-Mo Li and E.J. McCluskey. Diagnosis of resistive-open and stuck-open defects in digital CMOS ICs. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 24(11):1748 – 1759, Nov. 2005.

[56] B. Kruseman and M. Heiligers. On test conditions for the detection of open defects. In *Proceedings of Design, Automation and Test in Europe (DATE '06)*, volume 1, page 6 pp., March 2006.

[57] M. Arai, A. Suto, K. Iwasaki, K. Nakano, M. Shintani, K. Hatayama, and T. Aikyo. Small delay fault model for intra-gate resistive open defects. In *VLSI Test Symposium (VTS '09), 27th IEEE*, pages 27 –32, May 2009.

[58] N. Rajderkar, M. Ottavi, S. Pontarelli, Jie Han, and F. Lombardi. On the effects of intra-gate resistive open defects in gates at nanoscaled CMOS. In *Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2011 IEEE International Symposium on*, pages 309 –315, Oct. 2011.

[59] Syed Saqib Khursheed. Test and diagnosis of resistive bridges in multi-vdd designs. *Ph.D Thesis, University of Southampton*, February 2010.

[60] Chen Liu, Wei Zou, S.M. Reddy, Wu-Tung Cheng, M. Sharma, and Huaxing Tang. Interconnect open defect diagnosis with minimal physical information. In *IEEE International Test Conference (ITC 2007)*, pages 1 –10, Oct. 2007.

[61] D. Arumi, R. Rodriguez-Montanes, J. Figueras, S. Eichenberger, C. Hora, and B. Kruseman. Gate leakage impact on full open defects in interconnect lines. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 19(12):2209 –2220, Dec. 2011.

[62] A.C. Miller. IDDQ testing in deep submicron integrated circuits. In *Proceedings of International Test Conference*, pages 724 –729, 1999.

[63] M. Fawaz, N. Kobrosli, A. Chkeir, A. Chehab, and A. Kayssi. Transient current and delay analysis for resistive-open defects in future 16 nm CMOS circuits. In *Electronics, Circuits, and Systems (ICECS), 2010 17th IEEE International Conference on*, pages 438 –441, Dec. 2010.

[64] K. Baker, G. Gronthoud, M. Lousberg, I. Schanstra, and C. Hawkins. Defect-based delay testing of resistive vias-contacts a critical evaluation. In *Proceedings of International Test Conference*, pages 467 –476, 1999.

[65] W. Moore, G. Gronthoud, K. Baker, and M. Lousberg. Delay-fault testing and defects in deep sub-micron ICs-does critical resistance really mean anything? In *Proceedings of International Test Conference*, pages 95 –104, 2000.

[66] J.L. Garci anda Gervacio and V. Champac. Computing the detection of small delay defects caused by resistive opens of nanometer ICs. In *IEEE European Test Symposium (ETS)*, pages 126 –131, May 2010.

[67] A. Czutro, N. Houarche, P. Engelke, I. Polian, M. Comte, M. Renovell, and B. Becker. A simulator of small-delay faults caused by resistive-open defects. In *13th European Test Symposium*, pages 113 –118, May 2008.

[68] S.K. Goel, N. Devta-Prasanna, and R.P. Turakhia. Effective and efficient test pattern generation for small delay defect. In *27th IEEE VLSI Test Symposium (VTS '09)*, pages 111 –116, 2009.

[69] Seongmoon Wang. An efficient method to screen resistive opens under presence of process variation. In *VLSI Test Symposium (VTS), 2011 IEEE 29th*, pages 122 –127, May 2011.

[70] M.T. Mohammadat, N.B.Z. Ali, and F.A. Hussin. Multi-voltage aware resistive open fault modeling. In *Test Symposium (ETS), 2012 17th IEEE European*, pages 1 –6, May 2012.

[71] J.T.-Y. Chang and E.J. McCluskey. Detecting delay flaws by very-low-voltage testing. In *Proceedings of International Test Conference*, pages 367 –376, Oct. 1996.

[72] Noohul Basheer Zain Ali, M. Zwolinski, B.M. Al-Hashimi, and P. Harrod. Dynamic voltage scaling aware delay fault testing. In *Eleventh IEEE European Test Symposium (ETS '06)*, pages 15 –20, May 2006.

[73] J.T.-Y. Chang and E.J. McCluskey. Quantitative analysis of very-low-voltage testing. In *Proceedings of 14th VLSI Test Symposium*, pages 332 –337, Apr-May 1996.

[74] H. Hao and E.J. McCluskey. Very-low-voltage testing for weak cmos logic ICs. In *Proceedings of International Test Conference*, pages 275 –284, Oct. 1993.

[75] J.M. Soden, R.K. Treece, M.R. Taylor, and C.F. Hawkins. CMOS IC stuck-open-fault electrical effects and design considerations. In *Proceedings of International Test Conference*, pages 423 –430, Aug. 1989.

[76] M. Bushnell and V.D. Agrawal. *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Kluwer Academic, Boston, 2000.

[77] A.K. Pramanick and S.M. Reddy. On the fault coverage of gate delay fault detecting tests. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 16(1):78 –94, Jan. 1997.

[78] Chin Jen Lin and S.M. Reddy. On delay fault testing in logic circuits. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 6(5):694 – 703, September 1987.

[79] F. Hopsch, B. Becker, S. Hellebrand, I. Polian, B. Straube, W. Vermeiren, and H.-J. Wunderlich. Variation-aware fault modeling. In *Asian Test Symposium,*, pages 87 –93, Dec. 2010.

[80] F. Hopsch, B. Becker, S. Hellebrand, I. Polian, B. Straube, W. Vermeiren, and H.-J. Wunderlich. Variation-aware fault modeling. *Information Sciences, Science China Press, co-published with Springer*, 54:1813–1826, 2011.

[81] A.H. El-Maleh, S.S. Khursheed, and S.M. Sait. Efficient static compaction techniques for sequential circuits based on reverse-order restoration and test relaxation. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 25(11):2556 –2564, Nov. 2006.

[82] A. El-Maleh and S. Khursheed. Efficient test compaction for combinational circuits based on fault detection count-directed clustering. *Computers Digital Techniques, IET*, 1(4):364 –368, July 2007.

[83] HSPICE, Data sheet 2012. *http://www.synopsys.com/Tools/Verification/AMSVerification/CircuitSimulation/HSPICE/Documents/hspice_ds.pdf.*

[84] PSPICE, Data sheet 2012. *http://www.cadence.com/rl/Resources/datasheets/pcb_pspice_simulation_ds.pdf.*

[85] NGSPICE-24, Jan, 2012. http://ngspice.sourceforge.net/.

[86] BSIM4.7.0, Manual, University of California, Berkeley, Feb 2011. *http://www-device.eecs.berkeley.edu/bsim/?page=BSIM4_LR.*

[87] L.T. Wang, C.W. Wu, and X. Wen. *VLSI Test Principles and Architectures: Design for Testability.* Elsevier Morgan Kaufmann Publishers, San Francisco, 2006.

[88] J. Savir. Skewed-load transition test: Part i, calculus. In *Proceedings of International Test Conference*, page 705, Sept. 1992.

[89] J. Savir and S. Patil. On broad-side delay test. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 2(3):368 –372, Sept. 1994.

[90] ITRS, The International Technology Roadmap for Semiconductor 2005.

[91] ITRS, The International Technology Roadmap for Semiconductor 2011. *http://www.itrs.net/Links/2011ITRS/2011Chapters/2011Test.pdf.*

[92] J. Moreno, V. Champac, and M. Renovell. A new methodology for realistic open defect detection probability evaluation under process variations. In *VLSI Test Symposium (VTS), 2011 IEEE 29th*, pages 184 –189, May 2011.

[93] M. Renovell, F. Azais, and Y. Bertrand. Detection of defects using fault model oriented test sequences. *Journal of Electronic Testing: Theory and Applications*, 14:13–22, February 1999.

[94] I. Polian, P. Engelke, B. Becker, S. Kundu, J.-M. Galliere, and M. Renovell. Resistive bridge fault model evolution from conventional to ultra deep submicron. In *IEEE VLSI Test Symposium, VTS*, pages 343–348, May 2005.

[95] P. Engelke, I. Polian, M. Renovell, S. Kundu, B. Seshadri, and B. Becker. On detection of resistive bridging defects by low-temperature and low-voltage testing. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(2):327 –338, Feb. 2008.

[96] K.C.Y. Mei. Bridging and stuck-at faults. *Computers, IEEE Transactions on*, C-23(7):720 – 727, July 1974.

[97] M.J.Y. Williams and J.B. Angell. Enhancing testability of large-scale integrated circuits via test points and additional logic. *Computers, IEEE Transactions on*, C-22(1):46 – 60, Jan. 1973.

[98] B. Chess, C. Roth, and T. Larrabee. On evaluating competing bridge fault models for CMOS ICs. In *Proceedings of VLSI Test Symposium*, pages 446 –451, Apr. 1994.

[99] S. Chakravarty, YiShing Chang, H. Hoang, S. Jayaraman, S. Picano, C. Prunty, E.W. Savage, R. Sheikh, E.N. Tran, and K. Wee. Experimental evaluation of bridge patterns for a high performance microprocessor. In *Proceedings of 23rd IEEE VLSI Test Symposium*, pages 337 – 342, May 2005.

[100] Huaxing Tang, Gang Chen, S.M. Reddy, Chen Wang, J. Rajski, and I. Pomeranz. Defect aware test patterns. In *Design, Automation and Test in Europe, 2005. Proceedings*, pages 450 – 455 Vol. 1, March 2005.

[101] J.A. Acken and S.D. Millman. Accurate modeling and simulation of bridging faults. In *Custom Integrated Circuits Conference, Proceedings of the IEEE 1991*, pages 17.4/1 –17.4/4, May 1991.

[102] S.D. Millman and J.M. Acken. Special applications of the voting model for bridging faults. *Solid-State Circuits, IEEE Journal of*, 29(3):263 –270, Mar. 1994.

[103] P.C. Maxwell and R.C. Aitken. Biased voting: A method for simulating cmos bridging faults in the presence of variable gate logic thresholds. In *Proceedings of International Test Conference*, pages 63 –72, Oct. 1993.

[104] J. Segura, J.L. Rossello, J. Morra, and H. Sigg. A variable threshold voltage inverter for CMOS programmable logic circuits. *Solid-State Circuits, IEEE Journal of*, 33(8):1262–1265, Aug 1998.

[105] M. Renovell, F. Azais, and Y. Bertrand. Test escapes: Analysis of short defect. *Integrated Circuit Design and System Design, Symposium on*, 0:0160, 1999.

[106] V. R. Sar-Dessai and D. M. H. Walker. Resistive bridge fault modeling, simulation and test generation. In *Proceedings of the International Test Conference (ITC)*, pages 596–605, Atlantic City, NJ, USA, September 1999.

[107] T. Shinogi, T. Kanbayashi, T. Yoshikawa, S. Tsuruoka, and T. Hayashi. Faulty resistance sectioning technique for resistive bridging fault ATPG systems. In *Proceedings of 10th Asian Test Symposium*, pages 76 –81, 2001.

[108] P. Engelke, I. Polian, M. Renovell, and B. Becker. Automatic test pattern generation for resistive bridging faults. *Journal of Electronic Testing: Theory and Applications*, 22:61–69, February 2006.

[109] W. Shockley and G. L. Pearson. Modulation of Conductance of Thin Films of SemiConductors by Surface Charges. *Physical Review*, 74:232–233, 1948.

[110] A. Sreedhar, A. Sanyal, and S. Kundu. On modeling and testing of lithography related open faults in Nano-CMOS circuits. *Design, Automation and Test in Europe (DATE '08)*, pages 616–621, March 2008.

[111] D. Arumi, R. Rodriguez-Montaes, J. Figueras, S. Eichenberger, C. Hora, and B. Kruseman. Full open defects in nanometric CMOS. In *26th IEEE VLSI Test Symposium (VTS 2008)*, pages 119 –124, April 2008.

[112] C.L. Henderson, J.M. Soden, and C.F. Hawkins. The behavior and testing implications of CMOS IC logic gate open circuits. In *Proceedings of International Test Conference*, page 302, 1991.

[113] S. Johnson. Residual charge on the faulty floating gate cmos transistor. In *Proceedings of International Test Conference*, pages 555 –561, 1994.

[114] U. Choudhury and A. Sangiovanni-Vincentelli. Automatic generation of analytical models for interconnect capacitances. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 14(4):470–480, Apr. 1995.

[115] Wei Zou, Wu-Tung Cheng, and S.M. Reddy. Interconnect open defect diagnosis with physical information. *15th Asian Test Symposium (ATS '06)*, pages 203–209, November 2006.

[116] S.-H. Lo, D.A. Buchanan, Y. Taur, and W. Wang. Quantum-mechanical modeling of electron tunneling current from the inversion layer of ultra-thin-oxide nMOSFET's. *Electron Device Letters, IEEE*, 18(5):209 –211, May 1997.

[117] R.S. Guindi and F.N. Najm. Design techniques for gate-leakage reduction in CMOS circuits. In *Proceedings of Fourth International Symposium on Quality Electronic Design*, pages 61 – 65, 2003.

[118] U. Ingelsson and B.M. Al-Hashimi. Investigation into voltage and process variation-aware manufacturing test. In *Proceedings of International Test Conference (ITC)*, pages 1 –10, Sept. 2011.

[119] A.N. Hariharan, S. Pontarelli, M. Ottavi, and F. Lombardi. Modeling open defects in nanometric scale CMOS. In *Defect and Fault Tolerance in VLSI Systems (DFT), 2010 IEEE 25th International Symposium on*, pages 249 –257, Oct. 2010.

[120] V. Champac, J. Vazquez Hernandez, S. Barcelo, R. Gomez, C. Hawkins, and J. Segura. Testing of stuck-open faults in nanometer technologies. *Design Test of Computers, IEEE*, 29(4):80 –91, aug. 2012.

[121] S.M. Reddy, I. Pomeranz, Huaxing Tang, S. Kajihara, and K. Kinoshita. On testing of interconnect open defects in combinational logic circuits with stems of large fanout. In *Proceedings of International Test Conference*, pages 83 – 89, 2002.

[122] Zhuo Li, Xiang Lu, Wangqi Qiu, Weiping Shi, and D.M.H. Walker. A circuit level fault model for resistive opens and bridges. In *Proceedings of 21st VLSI Test Symposium*, pages 379 – 384, April-May 2003.

[123] D. Arumi, R. Rodriguez-Montanes, and J. Figueras. Experimental characterization of CMOS interconnect open defects. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(1):123 –136, Jan. 2008.

[124] B. Kruseman, A.K. Majhi, G. Gronthoud, and S. Eichenberger. On hazard-free patterns for fine-delay fault testing. In *Proceedings of International Test Conference (ITC 2004)*, pages 213 – 222, Oct. 2004.

[125] M. Favalli and C. Metra. Testing resistive opens and bridging faults through pulse propagation. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 28(6):915 –925, June 2009.

[126] M. Yilmaz, K. Chakrabarty, and M. Tehranipoor. Test-pattern grading and pattern selection for small-delay defects. In *26th IEEE VLSI Test Symposium (VTS 2008)*, pages 233 –239, May 2008.

[127] J.C.M. Li, Chao-Wen Tseng, and E.J. McCluskey. Testing for resistive opens and stuck opens. In *Proceedings of International Test Conference*, pages 1049 –1058, 2001.

[128] M. Favalli and M. Dalpasso. Bridging fault modeling and simulation for deep submicron CMOS ICs. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 21(8):941 – 953, Aug. 2002.

[129] M. Favalli and M. Dalpasso. High quality test vectors for bridging faults in the presence of IC's parameters variations. In *Defect and Fault-Tolerance in VLSI*

Systems, (DFT '07). 22nd IEEE International Symposium on, pages 448 –456, Sept. 2007.

[130] I. Pomeranz and S.M. Reddy. Forward-looking reverse order fault simulation for -detection test sets. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 28(9):1424 –1428, Sept. 2009.

[131] I. Pomeranz. Multi-pattern n-detection stuck-at test sets for delay defect coverage. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 20(6):1156 –1160, June 2012.

[132] Yinghua Min and Zhongcheng Li. IDDT testing versus IDDQ testing. *Journal of Electronic Testing*, 13:51–55, 1998.

[133] S. Khursheed, Shida Zhong, B.M. Al-Hashimi, R. Aitken, and S. Kundu. Modeling the impact of process variation on resistive bridge defects. In *Proceedings of International Test Conference (ITC)*, pages 1 –10, Nov. 2010.

[134] Neil H. E. Weste and Kamran Eshraghian. *Principles of CMOS VLSI design: a systems perspective*. Addison-Wesley Publishing Co., 1994.

[135] G. Angelov and M. Hristov. SPICE modeling of MOSFETs in deep submicron. In *Electronics Technology: Meeting the Challenges of Electronics Technology Progress, 27th International Spring Seminar on*, pages 257–262, May 2004.

[136] P. Engelke. Resistive bridging faults defect-oriented modeling and efficient testing. In *Ph.D Thesis, Freiburg University*, Jan 2009.

[137] Wei Zhao and Yu Cao. New generation of predictive technology model for sub-45 nm early design exploration. *Electron Devices, IEEE Transactions on*, 53(11):2816 –2823, Nov. 2006.

[138] A. Asenov, A.R. Brown, J.H. Davies, S. Kaya, and G. Slavcheva. Simulation of intrinsic parameter fluctuations in decananometer and nanometer-scale MOSFETs. *Electron Devices, IEEE Transactions on*, 50(9):1837–1852, Sept. 2003.

[139] Qunzeng Liu and S.S. Sapatnekar. A framework for scalable postsilicon statistical delay prediction under process variations. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 28(8):1201 –1212, Aug. 2009.

[140] Y. Cao, P. Gupta, A.B. Kahng, D. Sylvester, and J. Yang. Design sensitivities to variability: extrapolations and assessments in nanometer VLSI. In *ASIC/SOC Conference, 15th Annual IEEE International*, pages 411 – 415, 2002.

[141] Ja Chun Ku and Y. Ismail. On the scaling of temperature-dependent effects. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 26(10):1882 –1888, Oct. 2007.

[142] R.C. Aitken. Defect or variation? characterizing standard cell behavior at 90 nm and below. *Semiconductor Manufacturing, IEEE Trans. on*, 21(1):46 –54, Feb. 2008.

[143] S. Zhong, S. Khursheed, B.M. Al-Hashimi, S.M. Reddy, and K. Chakrabarty. Analysis of resistive bridge defect delay behavior in the presence of process variation. In *Asian Test Symposium*, pages 389–394, Nov. 2011.

[144] Shida Zhong, S. Khursheed, and B.M. Al-Hashimi. A fast and accurate process variation-aware modeling technique for resistive bridge defects. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 30(11):1719 –1730, Nov. 2011.

[145] S. Khursheed, B.M. Al-Hashimi, K. Chakrabarty, and P. Harrod. Gate-sizing-based single v_{dd} test for bridge defects in multivoltage designs. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 29(9):1409 – 1421, Sept. 2010.

[146] Andrei Vladimirescu. *The Spice Book*. John Wiley and Sons, Inc., 1994.

[147] K.G. Nichols, T.J. Kazmierski, M. Zwolinski, and A.D. Brown. Overview of SPICE-like circuit simulation algorithms. *Circuits, Devices and Systems, IEE Proceedings*, 141(4):242 –250, Aug. 1994.

[148] Vishal J. Mehta, Malgorzata Marek-Sadowska, Kun-Han Tsai, and Janusz Rajski. Timing defect diagnosis in presence of crosstalk for nanometer technology. In *Proceedings of International Test Conference (ITC '06)*, pages 1 –10, Oct. 2006.

[149] D. Arumi, R. Rodriguez-Montane, and J. Figueras. Defective behaviours of resistive opens in interconnect lines. In *European Test Symposium*, pages 28 – 33, May 2005.

[150] Shahdad Irajpour, Sandeep K. Gupta, and Melvin A. Breuer. Test generation for weak resistive bridges. In *Asian Test Symposium (ATS '06)*, pages 265 –272, Nov. 2006.

[151] FreePDK45, Apr, 2011. http://www.eda.ncsu.edu/wiki/FreePDK45:Contents.

[152] A. Asenov. Random dopant induced threshold voltage lowering and fluctuations in sub-0.1 μm MOSFET's: A 3-d "atomistic" simulation study. *Electron Devices, IEEE Transactions on*, 45(12):2505 –2513, Dec. 1998.

[153] K.J. Kuhn. Reducing variation in advanced logic technologies: Approaches to process and design for manufacturability of nanoscale CMOS. In *IEEE International Electron Devices Meeting (IEDM 2007)*, pages 471 –474, Dec. 2007.