Multi-Level Analysis of Non-Functional Properties

Von der Fakultät
Informatik, Elektrotechnik und Informationstechnik
der Universität Stuttgart zur Erlangung der Würde
eines Doktors der Naturwissenschaften (Dr. rer. nat.)
genehmigte Abhandlung

Vorgelegt von
Nadereh Hatami Mazinani
aus Teheran in Iran

Hauptberichter:
Prof. Dr. rer. nat. habil. Hans-Joachim Wunderlich
Mitberichter:
Prof. Dr. Paolo Prinetto

Tag der mündlichen Prüfung: 14.03.2014

Institut für Technische Informatik der Universität Stuttgart
2014

To my parents, for teaching me who the real god, what the real love, and where the real home is.

Acknowledgments

This thesis is the result of my research at the institute of computer architecture (ITI) at the University of Stuttgart, Germany.

First and most of all, I'd like to express my gratitude to my adviser, Professor Hans-Joachim Wunderlich, for all his support and trust. Working with him was a unique opportunity to learn, not only about technology challenges in semiconductor industry on which this work is based, but also on how to do a research and think as a researcher. His guidance and support was crucial in developing this dissertation. Professor Wunderlich has shown me the true meaning of mentoring, and I cannot thank him enough for the time and effort he has put on my education.

I also sincerely thank my co-adviser, Professor Paolo Prinetto. Beside all his technical support, he taught me that in order to be a great scientist and a good researcher, you have to first be a great human with a big heart. The opportunities he offered me during my stay in Politecnico di Torino and during my PhD were countless, and invaluable.

In addition, I'd like to thank my colleague, Rafal Baranowski, for his technical contributions to this research. His help and support was necessary in shaping this work in the way it is.

Being thousand kilometers away from home, I learned the value of the Family. This work would look much more differently without the infinite support of my loved ones. They were far, but never let me feel the distance. My parents with their unlimited love, and my brother with his endless care, were sometimes my only reason for going on. Without them, I would probably have given up early at the beginning. In addition, I have to thank my dear Kaveh, for his technical as well as psychological support. I was lucky to have him by my side during writing this dissertation, for all the moments of despair and mindlessness. Him, just being there for me, was a priceless gift I was lucky to have.

I'd also like to thank all the friends who supported me during my doctoral studies. In particular, I have to thank Mahin Shahlari and Gert Schley, for their support, understanding, their endless patience in listening and encouraging and for their precious friendship.

PhD is a unique opportunity to learn and grow. During this time, I had the chance to know several people, improve my skills and learn to look at life from different angels. Being a student abroad made this experience more challenging, and at the same time more enjoyable for me. My PhD was not limited to trying to contribute to the edge of knowledge, but also learning new cultures and integrating myself with the new world. I sincerely thank all the people who didn't let me feel alone in this adventure, stood by my side and supported me, in any possible way they could.

Nadereh Hatami

17.02.2014

Contents

List of Figures

List of Tables

Abbreviations

2D-DCT	2 Dimensional Discrete Cosine Transform
AC-SIM	Autonomous Cycles SIMulation
AES	Advanced Encryption Standard
AHB	Advanced High-performance Bus
AHI	Anode Hole Injection
AHR	Anode Hydrogen Release
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
AVF	Architectural Vulnerability Factor
BERT	BErkeley Reliability Tools
BIST	Built-In Self-Test
CMOS	Complementary Metal-Oxide-Semiconductor
CPU	Central Processing Unit
DES	Discrete Event-driven Simulation
Triple-DES	Triple Data Encryption Standard
DMI	Direct Memory Interface
DUA	Design Under Analysis
DUE	Detected Unrecoverable Error
ESL	Electronic System Level
FIT	Failure In Time
FPU	Floating Point Unit
GIDL	Increased Gate-Induced Drain Leakage
GP-GPU	General Purpose Graphics Processing Units
HBD	Hard BreakDown

HCI	Hot-Carrier Injection
IC	Integrated Circuits
IDCT	Inverse Discrete Cosine Transform
IS-SIM	Input Stimuli SIMulation
LCD	Liquid-Crystal Display
LP	Logical Process
LUT	Look-Up Table
McPAT	MultiCore Power, Area, and Timing
NBTI	Negative Bias Temperature Instability
MOS	Metal-Oxide-Semiconductor
MSE	Mean Square Error
NFP	Non-Functional Property
NMOS	N-type Metal-Oxide-Semiconductor
NoC	Network-on-Chip
PDES	Parallel Distributed Event-driven Simulation
PMOS	P-type Metal-Oxide-Semiconductor
ROM	Read-Only Memory
RTL	Register Transfer Level
SBD	Soft BreakDown
SDC	Silent Data Corruption
SFI	Statistical Fault Injection
SoC	System-On-Chip
STA	Static Timing Analysis
SRAM	Static Random-Access Memory
TDDB	Time Dependent Dielectric Breakdown
TLM	Transaction Level Modeling
TPS	Time-Parallel Simulation
TCA	Transition Counter Array
ULSI	Ultra Large Scale Integrated
VCD	Value Change Dump
VGA	Video Graphics Array

VLSI	Very Large Scale Integrated

Abstract

System properties are usually classified into functional (behavioral) and non-functional properties (NFPs). While functional properties refer to system behavior, NFPs are attributes, or constraints of a system. Power dissipation, temperature distribution on the chip, vulnerability to soft and intermittent errors, reliability and robustness are all examples of NFPs.

The exponential increase of system complexity and the transistor's smaller feature sizes pose new challenges to functional as well as non-functional properties of the system. Therefore, it becomes more important to understand and model their impact at early design phases.

This work targets dynamic, quantifiable NFPs and aims at providing a basis for the accurate analysis of this class of NFPs at early design phases. It proposes an accurate and efficient NFP characterization and analysis method for complex embedded systems. An efficient NFP prediction method helps designers understand how the devices behave over time, identify NFP bottlenecks within circuits and make design trade-offs between performance and different NFPs in the product design stage. It assists manufacturers build their circuits such that no performance degradation due to specific NFPs dominate over the life of an operating device.

The developed methodology is based on an efficient multi-level system-wide simulation that considers the target system application. High NFP evaluation speed is achieved using a novel piecewise evaluation technique which splits the simulation time into evaluation windows and efficiently evaluates NFP models once per window by partial linearization. The piecewise evaluation method is a fast, yet accurate replacement for a cycle-accurate NFP evaluation. To consider the mutual impact of different NFPs on each other, all NFP models are integrated into a common evaluation framework. The effect of some positive or negative feedback between different NFPs is dynamically considered during simulation. Evaluations are based on target applications instead of corner case analysis to provide a realistic prediction.

The contributions of this work can be summarized as follows:

- **Generality**: This work proposes a holistic, scalable NFP prediction methodology for multiple, interdependent NFPs. The NFP simulation and evaluation method is independent of a specific NFP, a particular model or a specific system or core. In addition, the method allows for multiple designs under analysis and multiple NFPs. As soon as the system is available at transaction level, it can be used for NFP estimation.

- **Speed up**: The NFP-aware simulation is performed on a multi-level platform while low level simulation is accelerated using parallelism. The complete system simulation is always kept at transaction level. All the NFPs under analysis can be estimated with a single simulation run. In addition, the evaluation speed can be increased by increasing the size of the evaluation window.

- **Accuracy**: The accuracy is a function of the accuracy of the selected model and the evaluation methodology. This work provides a method for integrating arbitrary low-level models into the system analysis. The right choice of low-level models may depend on the requirements for accuracy and efficiency. To preserve the low level evaluation accuracy, the required observables for piecewise evaluation are obtained at low level. The evaluation accuracy for the piecewise approach can be adjusted by calibrating the window size. Besides, rather than using statistical or worst-case analysis techniques (which may be too pessimistic in case of embedded systems with well defined applications), the complete system is simulated with the target applications and actual workloads to obtain higher accuracy for specific applications.

Zusammenfassung

Systemeigenschaften werden üblicherweise in funktionale und nicht-funktionale Eigenschaften (NFPs) unterschieden. Während funktionale Eigenschaften sich auf das Verhalten eines Systems beziehen, entsprechen NFPs Attributen oder Einschränkungen des Systems. Die Verlustleistung, Temperaturverteilung auf dem Chip, die Anfälligkeit für transiente und intermittierender Fehler sowie Zuverlässigkeit und Robustheit sind Beispiele für NFPs.

Die exponentielle Zunahme der Systemkomplexität und immer kleineren Strukturgrößen der Transistoren stellen neue Herausforderungen sowohl an die funktionalen als auch an die nicht-funktionalen Eigenschaften des Systems. Daher wird es immer wichtiger, bereits in frühen Entwurfsphasen die Auswirkung dieser Eigenschaften auf kritische Design-Parameter zu verstehen und diese in einem Modell abzubilden.

Das Ziel dieser Arbeit ist es, eine Basis für eine genaue Analyse dynamischer, quantifizierbarer NFPs für frühe Entwurfsphasen zur Verfügung zu stellen. Das Ergebnis dieser Arbeit ist ein Verfahren zur genauen und effizienten NFP Charakterisierung sowie eine dazugehörige Analyse-Methode für komplexe Eingebettete Systeme. Eine effiziente Methode zur Analyse von NFPs hilft dem Designer, das Verhalten eines Systems über die Zeit zu verstehen, NFP Engpässe zu identifizieren und Abwägungen zwischen Performanz und NFPs in der Produktentwicklungsphase zu treffen. Sie unterstützt Hersteller ihre Schaltungen so zu entwerfen, dass kein Leistungsverlust aufgrund spezifischer NFPs während der Lebensdauer eines Geräts im Betrieb auftritt.

Die vorgestellte Methode basiert auf einer leistungsfähigen mehrstufigen Simulation des Gesamtsystems, welche die Anwendung des Zielsystems berücksichtigt. Eine hohe Geschwindigkeit bei der Auswertung von NFPs wird durch die Verwendung einer neuartigen, abschnittsweisen Auswertetechnik erreicht, welche die Simulationszeit in Evaluierungsfenster (oder Auswertungsfenster) unterteilt und NFP Modelle einmal pro Fenster mittels partieller Linearisierung bewertet. Dieses abschnittsweise Bewertungsverfahren ist ein schneller und dennoch genauer Ersatz für die bisherige zyklengenaue NFP Auswertung. Um die gegenseitige Beeinflussung unterschiedlicher NFPs betrachten zu können, werden die NFP Modelle in einem gemeinsamen Evaluierungsframework integriert. Der Einfluss von positiven oder negativen Rückkopplungen zwischen verschiedenen NFPs wird dynamisch während der Simulationen berücksichtigt. Um eine realistische NFP Vorhersage zu treffen, basiert die Evaluierung statt auf der Analyse von Extremfällen auf dem Einsatz der Zielanwendung.

Die Beiträge dieser Arbeit können wie folgt zusammengefasst werden:

- **Allgemeingültigkeit**: Diese Arbeit schlägt eine ganzheitliche, skalierbare NFP Vorhersagemethode für mehrere, voneinander abhängiger NFPs vor. Simulation und Bewertung sind unabhängig von spezifischen NFPs und Modellen oder bestimmten Systemen. Darüber hinaus ermöglicht die vorgestellte Methode die Analyse mehrerer Designs und NFPs gleichzeitig. Sobald das System auf Transaktionsebene verfügbar ist, kann es für die NFP Abschätzung verwendet werden.
- **Performanzgewinn**: Die NFP berücksichtigende Simulation wird auf einer ebenenübergreifenden Plattform durchgeführt. Während Simulationen auf unterer Ebene zur Beschleunigung parallel ausgeführt werden, wird das Gesamtsystem stets auf der Transaktionsebene simuliert. Die komplette Systemsimulation wird immer auf Transaktionsebene dürchgeführt. Alle zu analysierenden NFPs können mit einem einzigen Simulationslauf abgeschätzt werden. Darüber hinaus kann die Auswertungsgeschwindigkeit durch das Vergrößern des Auswertungsfensters erhöht werden.
- **Genauigkeit**: Die Genauigkeit hängt von der Exaktheit des ausgewählten Modells und der Bewertungsmethode ab. Diese Arbeit stellt eine Methode für die Integration beliebiger Modelle auf niedriger Ebene in der Systemanalyse vor. Die geeignete Wahl eines

Modells hängt von den Anforderungen an die Genauigkeit und Effizienz ab. Um die Auswertungsgenauigkeit von niedrigen Ebenen zu gewährleisten, werden die erforderlichen Messgrößen für eine abschnittsweise Bewertung ermittelt und durch Kalibrierung der Fenstergröße angepasst. Statistische oder Worst-Case Analysetechniken können im Fall von Eingebetteten Systemen mit festgelegten Applikationen pessimistische Ergebnisse liefern. Aus diesem Grund wird das Gesamtsystem zusammen mit der Zielapplikation und der entsprechenden Arbeitslast simuliert, um eine genauere Analyse spezifischer Applikationen zu erhalten.

Chapter 1

Introduction

Hardware design and production in today's semiconductor industry is a time-consuming, error-prone, and highly competitive task [Rigo11]. Introducing new generation of hardware chips in a better quality and a larger quantity in a short time is a challenge every semiconductor company has to face. To compensate the high investment necessary to become a pioneer and produce more complex hardware components in newer technologies, a company must be able to develop and produce new products with the fastest rate, minimum cost, and the highest quality [Doer07]. However, product quality and yield starts at the design stage and is not simply a manufacturing responsibility. Any potential problem should be discovered and eliminated as close to the beginning of the product cycle as possible [Chia07].

The advancements in manufacturing technology has fulfilled Moor's Law [Moor69] for more than 40 years by constantly reducing the transistors feature sizes [ITR12]. With shrinking geometries, the performance of very large scale integrated (VLSI) systems increases everyday. However, the impact of environmental variations on performance has also been increasing with each semiconductor technology generation. New technologies increase the number of transistors per silicon chip, while reduce the strength of the transistor facing operational and environmental variations [Baha07, Shan06, Kawa06]: Circuits will encounter dynamic variations of supply voltage and temperature; new nano-scale transistors are more prone to faults and frequent, intermittent soft errors; transistors slowly age and degrade over time, deteriorating circuit performance [McPh06, Bork04, Sriv05].

System properties are usually divided to functional (behavioral) and *non-functional properties* (NFPs). While functional properties refer to behavior of a system, NFPs are attributes, or constraints on a system [Glin07]. Power dissipation, temperature distribution on the chip, vulnerability to soft and intermittent errors, reliability and robustness are all examples of NFPs.

Beside the exponential increase of complexity which poses problems to functional behavior of silicon chips, smaller feature sizes introduce new challenges to non-functional properties: Checking NFPs such as on-chip temperature as well as the impact of environment on the system, that were ignored or only considered in rare application areas in the past, are getting more focused on during system design because of nanoelectronic effects [Vieh09]; new nano-scale transistors are more prone to variability which manifest as a violation of NFPs such as components' reliability and performance [Bork05]. On the other hand, rapid moves to new technologies leave older technologies immature and prevent the development of novel approaches to deal with the emerging challenges [De99].

As the dimensions of the above mentioned NFPs increase, it becomes more important to understand and model their impact on the most sensitive design parameters at early design phases [Chia07]. Proper design strategies can reduce the negative impact of scaling on NFPs [Bork05]. However, more accurate analysis is required to assure the reliability and performance of the new products which is the final goal of the production.

Traditional design methods, in which the system is designed directly at low hardware levels, are unable to characterize the whole complexity of NFPs [Segu04]. In addition, they are almost infeasible for today's complex systems [Gajs09]. Electronic system level (ESL) design fills the productivity gap generated by the disparity between the design productivity and design complexity by introducing high abstraction design methodologies. However, efficient methods for analyzing NFP fluctuations at these levels are not yet developed.

Current approaches for estimating NFPs are either based on corner, model-based analysis or analytical NFP model evaluation techniques. Traditional corner, model-based analysis approaches are limited to verifying the functional correctness by simulating the design at the number of process corners. The simulation results are used to provide guardbands for NFPs based on the worst operational and environmental conditions. However, the worst case conditions in a circuit may not always occur with all parameters at their worst or best process conditions. Such pessimism can lead to increased design effort and longer time to market, which ultimately may result in lost revenue. On the other hand, analytical models are unable to study the impact of dynamic variations on NFPs which should be simultaneously captured in a single comprehensive analysis, allowing correlations and impact on yield to be properly understood.

An efficient NFP prediction method helps designers to understand how the devices behave over time, identify the NFP bottlenecks within the circuits and make design trade-offs between performance and different NFPs in the product design stage. It assists manufacturers build their circuits such that no performance degradation due to specific NFPs dominate over the life of an operating device. An accurate NFP prediction considering dynamic variations and model interdependencies assures adequate performance and reliability for the product. Early NFP prediction in the design flow enables NFP-aware, system level design decisions and leads to more reliable architectures.

Available models for various NFPs use low level parameters for accurate NFP analysis. Nevertheless, low level analysis is hardly feasible for new, complex system on chips (SoCs). Heterogeneous multiprocessor platforms are increasingly being used in system design to deal with the growing complexity and performance demands of modern applications. Choosing an optimal platform for a given application and an optimal mapping of the application to the platform is crucial [Hwan08]. Such system level decisions require early and accurate estimation of NFPs such as power, temperature and reliability for a given design choice. Cycle-accurate models do provide accuracy, but may not be available for the whole platform. Furthermore, cycle accurate instruction set simulation models for processors and register transfer level (RTL) models for custom hardware are too slow for efficient design space exploration. The cost of wrong design decisions increases as we move through design steps. New generations of ULSI (Ultra Large Scale Integrated) circuits require architecture exploration and validation at early design phases to prevent the need for redesign [Gajs09].

Some sparse works in literature address analysis of particular NFPs and try to propose methodologies to realize early estimation of specific NFPs. Most of these works fall in the area of performance estimation [Vieh09, Liao05], temperature [Shan06, Pedr06, Paci06], power

[Burc93, Najm94, Reth11] or aging analysis [Obor12, Han11]. However, there is not enough research on defining and evaluating an arbitrary NFP. Timing constraints for reaching the market demand an integrated design methodology for hardware simulation and NFP prediction at early design stages. Furthermore, the need to increase yield and performance requires this methodology to be as accurate as possible to guarantee the reliability of the fabricated hardware.

This research targets dynamic, quantifiable NFPs and aims at providing basis for accurate analysis of this class of NFPs at early design phases. We propose a holistic, scalable NFP prediction methodology for multiple, interdependent NFPs. Evaluations are based on target application instead of corner case analysis to provide a realistic prediction of NFPs. The main steps considered to reach this goal are:

- NFP modeling: The first step in predicting an NFP is to study its nature and characteristics. Several NFPs have a strong dependence on system structure. For other NFPs, the environmental and operational variables play an important role. The first group can be well characterized via NFP-aware simulation. The latter requires that the NFP characteristics are collected in a general analytical model. The operational parameters (observables) and inter-related NFPs are extracted from the model.
- NFP-aware simulation: This work proposes a fast, yet accurate *multi-level NFP-aware simulation* methodology. The presented methodology is flexible in terms of NFP modeling. If the NFP is modeled as an structure-dependent parameter (such as vulnerability), it provides a comprehensive NFP prediction for the complete system at early design phases. For more complex NFPs with an analytical model, it obtains the operational parameters required for accurate NFP evaluation.
- NFP evaluation: Finally, the proposed *piecewise evaluation* method allows an accurate, long-term NFP evaluation which captures and applies temporal environmental and operational changes on NFP parameters at discrete timesteps during the evaluation.

In the following, a brief introduction to each step is presented.

NFP Modeling

The parameters which affect an arbitrary NFP are divided to three groups of process (λ's), operational (α's) and environmental (γ's) parameters. A general model for an arbitrary NFP is then expressed as:

$$\begin{aligned} y^{(t)} = f(&\lambda_1, \lambda_2, \ldots, \lambda_p, \\ &\alpha_1(.), \alpha_2(.), \ldots, \alpha_q(.), \\ &\gamma_1(.), \gamma_2(.), \ldots, \gamma_r(.), t) \\ f, \alpha_i, \gamma_i &: \mathbb{R} \rightarrow \mathbb{R}, \quad p, q, r \in \mathbb{N}, \quad \lambda_i \in \mathbb{R} \end{aligned}$$

where $t \in \mathbb{R}^+$ is the time. λ's are process dependent constants. α's, called *observables* in this text, heavily depend on system workload and can be obtained either by simulation or by the probabilistic methods. For instance, existing models for power consumption and heat distribution rely on the proportional relation of the transistor switching activity [Ghos92, Huan04]. γ's are NFPs which affect the NFP under analysis. The general NFP model is the building block for the proposed piecewise evaluation approach.

NFP-Aware Simulation

Time-dependent, quantifiable NFPs can be studied either via simulation or analytical evaluation, depending on how the NFP is defined and modeled. If the NFP is very structure dependent, such as random pattern testability or vulnerability to soft errors, simulation-based approaches may be applied. Otherwise, if it is a function of several various parameters and other NFPs, analytical models are the best solution. In this case, simulation can still be used to acquire low level observables. The proposed multi-level NFP prediction approach provides a simulation-based methodology for NFP prediction and observable acquirement acquirement at early design phases.

Most available NFP models utilize low level observables which cannot be estimated accurately at high abstraction levels. Typically, observables (α's) are obtained by modeling system workload with input signal probabilities or by a set of "typical workload patterns" that reflect the average application [Lore09, Wang10, Wang07c]. As the technology scales, the amplitude of NFP fluctuations under various workloads (e.g., the worst vs. average case) is growing [Srin04a]. This necessitates either a pessimistic worst-case analysis, or an extensive simulation of the target application.

To estimate α's, the proposed multi-level, NFP-aware simulation analyzes the design under analysis (DUA) with high accuracy within the target system and for the target application. To overcome the computational complexity, the system is simulated at multiple abstraction levels, from gate- up to transaction level.

Use of transaction level models makes NFP-aware architecture exploration achievable while the low level, parallel simulation provides the accuracy of cycle-accurate analysis at high speed.

A block view of the proposed approach is shown in Fig. 1.1. The observables of the DUA are collected during gate level simulation to maintain accuracy, while the rest of the system is simulated behaviorally at high level. High level simulation speeds up the simulation by abstracting the unnecessary details. High level models are easy to extract from system specifications early in the design flow for evaluating the design alternatives with respect to NFPs. To this end, high level system models from design space exploration can be reused. The intermediate simulation provides the logic simulator with the system state. Therefore, logic simulation is performed in parallel to speed up the low level DUA simulation.

For NFPs which require an analytical evaluation, the window-based simulation methodology is proposed which is synchronized with the piecewise evaluation as explained in the next section.

NFP Evaluation

To manage complex dependencies between different NFPs, the piecewise evaluation first builds an equivalent piecewise model from the original NFP model. The piecewise model allows iterative evaluation at discrete timesteps. During the evaluation, the operation time of the DUA

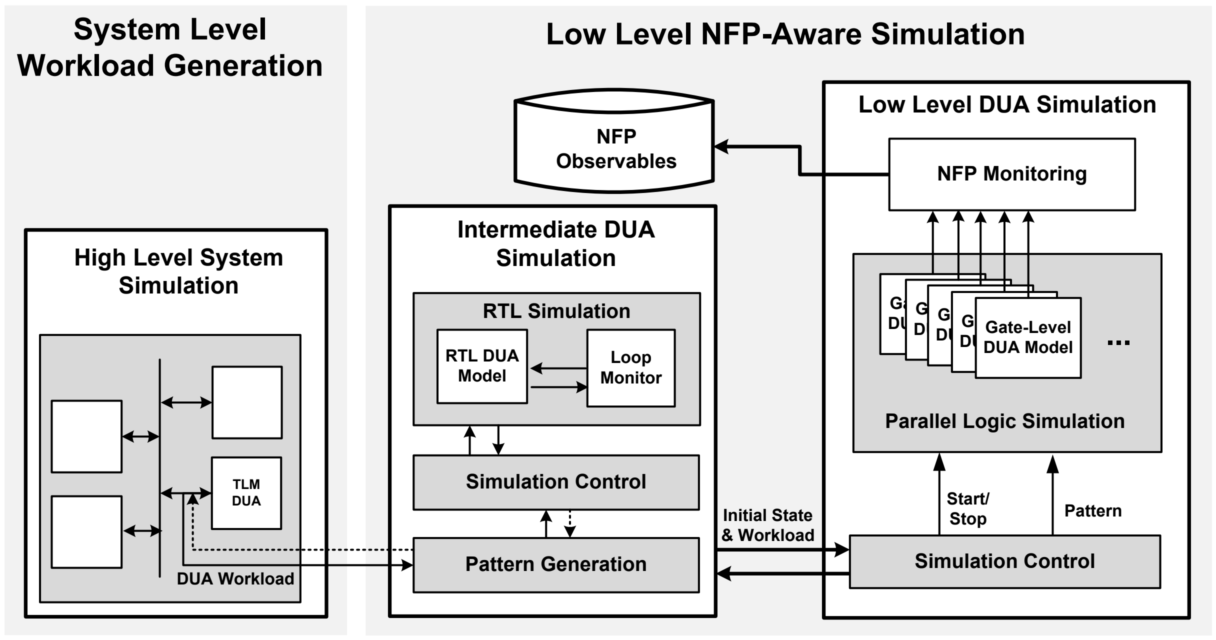


FIGURE 1.1: Multi-level parallel simulation approach

is split into *evaluation windows*. Within an evaluation window, the piecewise NFP model is partially linearized with respect to the observables and the interdependent NFPs.

Having the initial value of NFP y at time t_0, $y^{(t_0)}$, the piecewise evaluation approach converts the general NFP model to the form:

$$\begin{aligned} y^{(t_0)} &= g(t_0, .) \\ y^{(t_n)} &= f(\lambda_1, \lambda_2, \dots, \lambda_p, \\ &\quad \alpha_1^{(t_n)}(.), \dots, \alpha_q^{(t_n)}(.), \\ &\qquad \gamma_1^{(t_n)}(.), \gamma_2^{(t_n)}(.), \dots, \gamma_{r-1}^{(t_n)}(.), y^{(t_n - W)}, t_n) \\ f, \alpha_i, \gamma_i &: \mathbb{R} \to \mathbb{R}, \quad p, q, r \in \mathbb{N} \end{aligned} \tag{1.1}$$

At the end of each window at time t, the NFP under analysis as well as the interdependent NFPs ($\gamma_1, \dots, \gamma_{r-1}$) are fed with proper process parameters and observables evaluated for the current window. The window-based simulation methodology splits the simulation time to windows with the same length as in the evaluation approach. In this case, NFP acquisition is performed at the end of each window. The simulation output is the NFP progression over time.

The NFP under analysis is also fed with its previous value, i.e., the value at time $t - W$, where W is the size of the window.

The piecewise evaluation speeds up the predictions by avoiding cycle-by-cycle evaluation. An efficient NFP prediction methodology improves the final product and reduce the time to market by enabling NFP-aware design decisions at early design phases and preventing redesign due to wrong design choices.

Research Scope and Contributions

The proposed NFP prediction approach considers the accuracy and speed as the deciding factors for the final result. It enables early NFP-aware architecture exploration by providing accurate prediction at high design abstraction level. It contributes to solving the previously mentioned challenges in the following ways:

- **Generality**: The methodology is not limited to a specific system or core and allows for multiple DUAs and multiple NFPs. Similarly, the NFP simulation and evaluation are independent of a particular NFP or model. As soon as the system is available at transaction level, it can be used to estimate NFPs during system operation. To this end, the designer is free to select NFP models according to the design requirements. The accuracy of predictions is a function of the accuracy of the model.

- **Speed up**: The method performs the simulation on a multi-level platform and carries out the NFP-aware simulation on different abstraction levels. Multi-level simulation enables parallelism which speeds up the low level simulation. The complete system simulation is always kept at transaction level. All the under analysis NFPs can be estimated with a single simulation run to bring even more speed up. In addition, the evaluation speed can be increased by increasing the size of the evaluation window.

- **Accuracy**: The accuracy is a function of the accuracy of the selected model and the evaluation methodology. This work provides a method for integrating arbitrary low-level models into the system analysis. The right choice of low-level models may depend on the requirements for accuracy and efficiency. To consider the mutual impact of different NFPs on each other, all NFP models are integrated into a common evaluation framework. The effect of positive or negative feedback between different NFPs is dynamically considered during simulation. To preserve the accuracy of low level evaluation, the NFP properties of the DUA is predicted at low level. The required observables for piecewise evaluation are also obtained by low level DUA simulation. The evaluation accuracy for the piecewise approach can be adjusted by calibrating the window size. Besides, rather than using statistical or worst-case analysis techniques (which may be too pessimistic in case of embedded systems with well defined applications), the complete system is simulated with the target application and actual workloads to obtain higher accuracy for specific applications.

The remainder of this dissertation discusses the contributions of this work in detail and is organized as follows:

Part I, *Basics*, discusses the concepts and tools which are used in this research. Chapter 2, *Non-Functional Properties*, gives a clear definition of NFPs considering the state-of-the-art. It classifies system properties into two groups of functional and non-functional properties and discusses the effect of each class on system characteristics. Finally, *variations* and their effect on NFPs are discussed in detail.

NFP models and the mechanisms which affect them have a key role in accurate NFP prediction. Chapter 3, *NFP Models*, presents NFP modeling concepts used in the sequel of this work. Aging models as an important environmental variation which affects the new hardware technologies are also discussed in this chapter. Different mechanisms are introduced and the mutual effect of the mentioned mechanisms on NFPs are presented.

Chapter 4, *NFP Simulation*, gives a brief background on the materials and concepts which are used in this research. An introduction to different hardware abstraction levels is provided in this chapter. Later, we discuss multi-level and parallel simulation as efficient tools which are utilized in this work to increase the NFP-aware simulation performance.

Part II, *NFP Prediction Methodologies*, focuses on the proposed approach to predict NFPs early in the design paradigm. It considers two different aspects of accurate NFP prediction:

Multi-level, NFP-aware simulation and NFP Evaluation. These two perspectives are discussed in chapters 5 and 6, respectively.

Chapter 5, *Window-Based NFP-Aware Simulation*, presents the multi-level NFP-aware simulation methodology for efficient NFP prediction. Compared to previous multi-level simulation techniques, this approach enables simulation of the complete system at transaction level while the gate level model is required only for the DUA. The simulation methodology enables the NFP-aware architecture exploration and validation by predicting the NFP changes within system lifetime with a single simulation run. The system is simulated with real application to provide more accuracy.

Chapter 6, *Piecewise NFP Evaluation*, proposes the piecewise evaluation approach for analytical NFP prediction. The piecewise evaluation applies temporal changes of dynamic variations to NFP estimations by evaluating the NFP models at discrete timesteps. First, a general analytical modeling for dynamic, quantifiable NFPs is proposed and the available models in literature are classified. Thereupon, methods for piecewise evaluation of each category are discussed.

Part III, *Applications*, shows the sample application of the proposed techniques to predict two different NFPs. Chapter 7, *Architectural Vulnerability Analysis*, uses the multi-level simulation for vulnerability analysis while chapter 8, *Application on Aging Prediction*, uses the piecewise evaluation to analyze the effect of two dominant aging mechanisms, NBTI and HCI, on the system performance.

Part IV, *Final Notes*, is dedicated to discussions on the presented research and the future directions. Chapter 9, *Concluding Remarks*, summarizes the work and discusses possible research directions it may enable in the future.

Part I

Basics

Chapter 2

Non-Functional Properties

2.1 Introduction

As the technology scales and the complexity of system-on-chips (SoCs) increases, non-functional aspects become as important as functionality [Glin07, Vieh09, Tayl09]. *Non-functional properties* (NFPs) refer to *how* the system performs its prespecified function and have a broad domain: Testability, system performance, reliability, vulnerability, soft error rate, power budget, heat distribution on the chip, etc. are all examples of NFPs. They are affected by several different parameters and are sometimes interrelated: Reliability is affected by static and dynamic physical effects. It is influenced permanently by process variations and modeling inefficiencies while physical phenomena such as aging effects and soft and intermittent errors violate reliability dynamically and during system lifetime. Several aging effects are exacerbated at high temperatures. Temperature profile of the chip changes by dynamic and static power consumption of each transistor on the chip.

The market pressure forces corporations to produce hardware with certain functionality in a short time, but the success of the product in the market certainly depends on the quality of NFPs [Tayl09]. A successful product considers NFPs in the complete design paradigm to avoid designs which either fail much earlier than expected or deviate from prespecified characteristics [Vieh09]. For example, power consumption and heat dissipation—the most prominent NFPs of a chip—enforced power-aware methodologies for design and verification. Scaling makes the fabricated chips more vulnerable to static and dynamic variations, such as soft errors and aging effects. Therefore, reliability prediction methods are utilized at different design levels.

NFP consideration for hardware design demands prior knowledge of the property's essence and the physical phenomena behind it. NFPs must be clearly defined and modeled at each design abstraction level [Agt11, Vieh09]. This facilitates malfunction and/or failure prediction of the system, before the chip is actually fabricated. For instance, the soft error rate of a chip or its components can be measured via accelerated (and somewhat expensive) neutron beam tests in cyclotrons or alpha particle tests; but these techniques require a functioning chip. At such a stage, it is expensive and possibly too late to correct any reliability problem [Mukh05].

NFPs are different in nature: Properties such as system performance change during hardware lifetime. Testability and fault tolerance can be improved at design level and are considered and evaluated before manufacturing. Reliability is usually improved by fault tolerance techniques, however, it can be violated by dynamic variations, such as soft errors and aging effects, during system operation. Modeling and evaluation of NFPs such as vulnerability to soft errors at design phase can fix this problem, however, various nature of NFPs and inconsistency of definitions have prevented the researchers from proposing a general solution for predicting a desired NFP early during design phase.

This work concentrates on dynamic (time-dependent), quantifiable non-functional properties. Unless otherwise noted in the text, in the sequel of this work, the term NFP is used for time-dependent NFPs that can be formally presented, modeled, and quantified. Section 2.2 proposes a general, well-defined definition for this class of non-functional properties. Section 2.3 presents hardware variations as an important dimension in investigating NFPs. Section 2.4 discusses the effect of variations on these properties. Finally, section 2.5 gives a brief summary of the presented chapter.

2.2 Definition of Non-Functional Properties

Requirement engineering and system properties have been points of interest for few decades. System properties define what the system does and the circumstance under which it operates

[Koto98, Wieg09]. System's utility is determined by its functional as well as its non-functional characteristics.

Fig. 2.1 visualizes system properties' classification. Functional properties are defined as properties that specify the inputs (stimuli) to the system, the outputs (responses) from the system, and behavioral relationships between them [Davi93]. The type of inputs, outputs and the relationship between them varies from one hardware abstraction level to the other. For example, at electrical level, the inputs and outputs are in the form of electrical signals while at gate level, they are abstracted to the form of logic values.

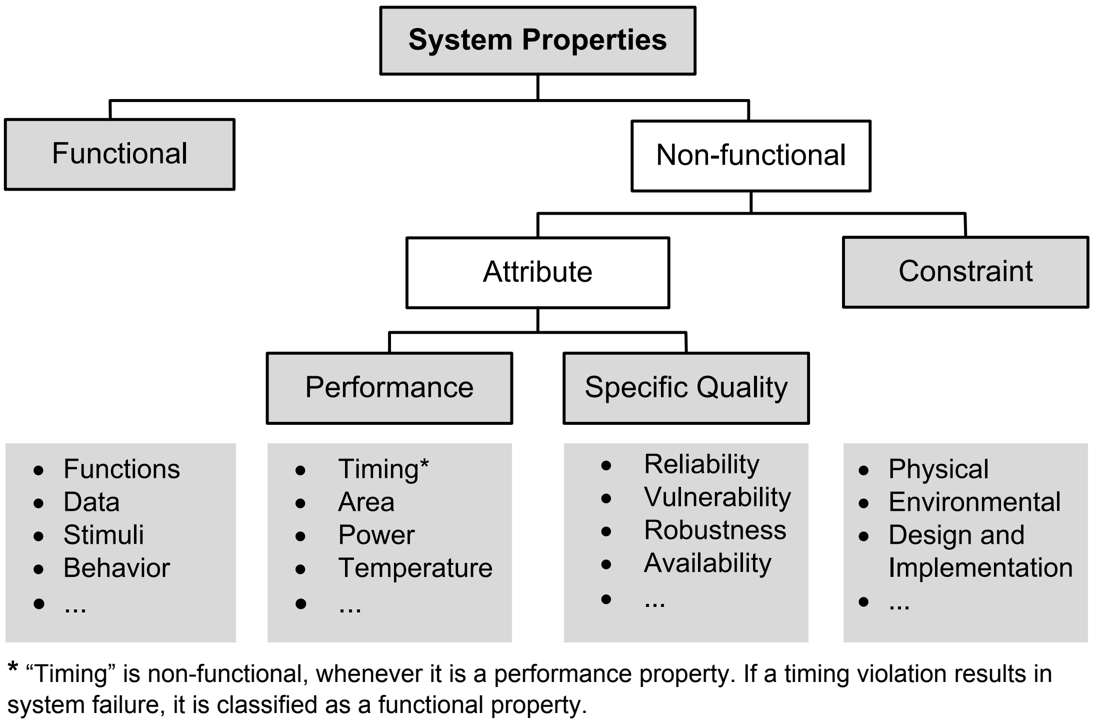

System Properties
Functional
Non-functional
Attribute
Constraint
Performance
Specific Quality
• Functions • Data • Stimuli • Behavior • ...
• Timing* • Area • Power • Temperature • ...
• Reliability • Vulnerability • Robustness • Availability • ...
• Physical • Environmental • Design and Implementation • ...
* "Timing" is non-functional, whenever it is a performance property. If a timing violation results in system failure, it is classified as a functional property.

FIGURE 2.1: A concern-based taxonomy of system properties [Glin07]

Contrary to functional properties, non-functional (extra-functional in some sources [Agt11, Saad12]) properties are defined as an attribute or a constraint on a system. An *attribute* is a performance specification or a specific quality of a system. Attributes are divided into two categories: Performance and quality properties [Glin07].

Similar to functional properties, NFPs may vary from one abstraction level to the other. The increase in transistor's threshold voltage at electrical level is translated to circuit delay at gate and correspondingly to performance degradation at register transfer or system level.

There are semantic differences that may arise for a certain NFP at different levels. For example, timing properties such as delay may be viewed as functional at electrical level while timing violation at this level results in system failure. However, at higher abstraction levels, such as gate or register transfer level, timing may also be considered as non-functional when it results in reduced frequency and therefore degraded performance. In this work, timing is considered non-functional whenever it is a performance property, else it is classified as a functional property.

The terms “functional” and “non-functional” properties are mentioned and defined in several works in literature. The existing definitions for functional properties follow two threads that coincide to a large extent. In the first thread, the emphasis is on functions. For example, the authors of [Robe06] believe that a functional property specifies what the product does. In [vL01], it is stated that functional concerns associate with the service to be provided. The second thread puts more emphasis on the behavior of the system and defines functional properties as system characteristics that describe the behavioral aspects of a system according to its inputs and outputs [Jaco99]. Davis in [Davi93] states that functional (or operational) properties are “those properties that specify the inputs (stimuli) to the system, the outputs (responses) from the system, and behavioral relationships between them. In [Wieg09], a functional property is defined as a statement of a piece of required functionality or a behavior that a system exhibits under specific conditions . Among the various definitions, the one from [Davi93] is widely accepted for its clarity and is also used in this work as a reference.

There are several debates [Koto98, Glin07] in literature relating to the definition of NFPs. Davis [Davi93] defines NFPs as the overall attributes of the system. In [Koto98], NFPs are defined as properties which are not specifically concerned with the functionality of the system. In [Robe06], NFPs are defined as properties, or qualities, that the product must have, such as an appearance, speed, or accuracy. Wiegers [Wieg09] defines NFPs as a description of a property or characteristic that a system must exhibit or a constraint that it must respect, other than an observable system behavior. [Sieg12] denotes that a non-functional property does not relate to the functionality of a variant, but to a quality or a behavioral attribute. The previously mentioned definition from Glinz [Glin07] summarizes other definitions and therefore is referenced in this text.

Most NFPs are usually too abstract to be stated formally [Zhan06]. Several NFPs are multi-dimensional properties which can be characterized in various dimensions [Vieh09]. For many NFPs, it is not obvious how we can interpret and handle the measured values [Sieg11]. However, effective NFPs such as power, heat distribution, reliability and vulnerability are modeled in literature and several metrics are proposed to predict or measure them. As an example, we discuss models for reliability and mean time to failure (MTTF) as one of the reliability metrics in more detail.

- **Reliability**: Reliability is one of the well known NFPs and is defined as the probability that a device will perform its intended function during a specified period of time under stated conditions [Baue10]. Several parameters including environmental conditions can affect reliability of a system during the operation time. Reliability is usually expressed with the exponential failure law:

  $$R(t) = \exp(-\lambda t)$$

  where t is time. λ is the constant failure rate and is usually expressed as the percentage of failures per 1000 hours or as failures per hour. More on reliability modeling and the proof of the above formula can be found in [Lala01].

- **Mean time to failure (MTTF)**: Mean time to failure is the average time before a component or system fails. Assume $R(t)$ is the reliability of the system at time t. MTTF is given by [Wang07a]:

  $$\text{MTTF} = \int_0^\infty R(t)d(t) \tag{2.1}$$

  In other words, MTTF is equal to the area under the reliability curve when plotted against time. This holds for any failure distribution. The units used for MTTF are typically hours.

As it can be seen, reliability and MTTF are well-defined and modeled. Several other NFPs, such as performance degradation due to several physical and environmental conditions [Stro06, Sriv05], vulnerability to soft errors [Mukh11], availability [Prad96] and robustness [Manz11, Bark12] can also be explained using formal models. This work, as will be discussed in chapter 3, targets this class of NFPs and provides solutions to predict them during the system lifetime.

2.3 Variability

The behavior of manufactured chips may differ from expectations or simulation models. This misbehavior is usually caused by different *variation*s which affect chip characteristics directly after production or during its lifetime [Orsh10]. The robustness and reliability of the final hardware system depend on the sensitivity of each single transistor to variations.

The concrete understanding of variability and it's impact on NFPs is one of the core activities in design for manufacturing[1]. Predicting the degradation due to variations and the effect on NFPs will result in higher yield and faster time-to-market which is crucial for any successful hardware production [Chia07].

Variability sources in IC design and manufacturing processes are as many as the required steps to design and manufacture. In addition, the operational environment of the finished IC product changes the transistor parameters and system behavior in time. Fig. 2.2 shows a possible classification for variation mechanisms. Variations are divided into two main categories based on their temporal occurrences [Alam11].

Static variations are the variations mostly caused by fabrication processes or modeling inaccuracies. In static variations, the distribution of measurements takes no account of time or sequence and the deviation from design specifications is permanent.

Runtime variations are variations which appear during the runtime of the system and are time-dependent. They can cause permanent (such as oxide breakdown or electromigration) or temporary malfunction (such as soft error effects or stress and recovery phases in some aging mechanisms) in the hardware.

[1]Design for manufacturing (DFM) stands for the methodology of ensuring that a product can be manufactured repeatedly, consistently, reliably, and cost effectively by taking all the measures needed for that goal. DFM starts at the concept stage of a design and implements these measures throughout the design, manufacturing, and assembly processes [Chia07].

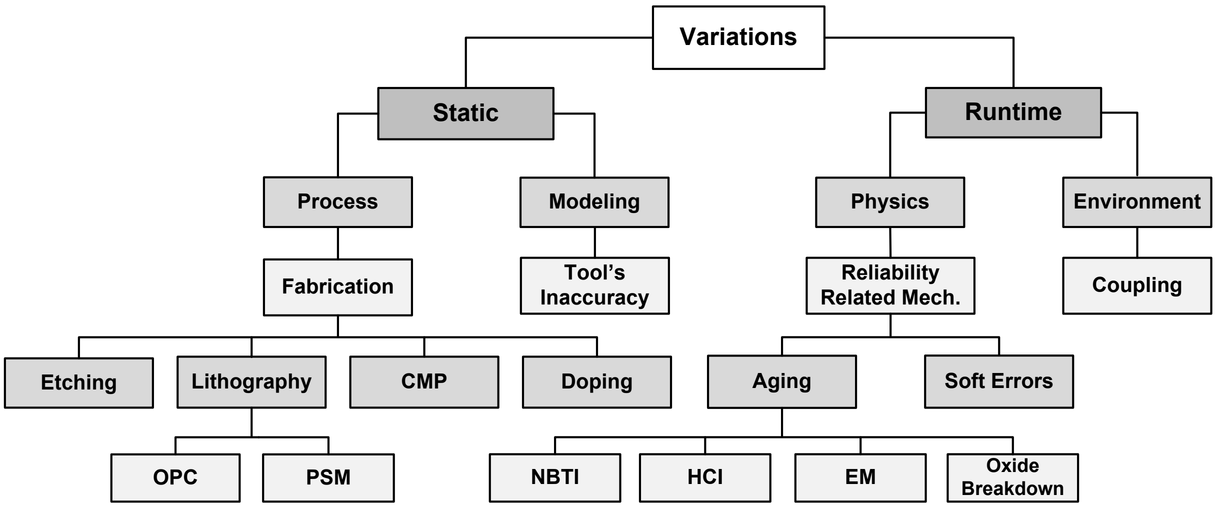


FIGURE 2.2: Mechanisms of variations

Accurate modeling and analysis of NFPs consider variation effects during system lifetime. It is important to model the sources of variability with the correct balance of computation effort and accuracy for NFP analysis [Nass04]. To this end, an understanding of variation mechanisms, as will be discussed in the following, is required. These mechanisms will later be used to perform NFP analysis at design level and before fabrication.

2.3.1 Static Variations

Static variations are a class of variations in which the distribution of measurements is independent of the time or sequence. The effect of these variations and the resulted deviation from design specifications is permanent. Static variations are usually divided into two groups of process and modeling variations.

2.3.1.1 Process Variations

Process variations are fluctuations in the value of process parameters observed after fabrication [Sriv05]. These variations result from a wide range of factors during the fabrication process. They affect the transistor's parameters and hence the hardware behavior in a permanent manner.

Large variations in process parameters lead to designs that deviate strongly from their specifications. These variations affect the performance characteristics of devices and the interconnects. The same variations in process parameters may influence two designs in very different manners.

Process variations are caused by different sources such as lithography (optical proximity correction (OPC), phase shifting masks (PSM)), etching, chemical mechanical polishing (CMP), doping process, etc. They can occur at different levels: Wafer level, inter-die level, and intra-die level. Some of process variations are systematic[2], i.e., those caused by the lithography process. Some have a purely random nature, such as edge roughness, etching, and CMP. Process parameters are mostly described by statistical analysis models such as Monte-carlo techniques or classical worst-case analysis [Sriv05, Shen12].

2.3.1.2 Modeling Variations

Design analysis and optimization models do not perfectly capture device characteristics and therefore apply some intrinsic inaccuracy to the evaluations. These models, if conservative, will make it harder to meet design specifications, whereas aggressive models will result in yield loss. The design variations posed by modeling inaccuracies are classified as modeling variations. The space of these variations is over design iterations, with different modeling errors at different design points. The alternative is to use smaller margins to capture modeling variations, but this may force post-fabrication tuning of particular paths or even a complete redesign [Sriv05].

Modeling variations affect all fabricated samples of the design and thus it is important to model them accurately and conservatively.

2.3.2 Runtime Variations

Runtime variations are time-dependent effects which appear during the operation time of the system. They include two main groups: Physical and environmental variations.

[2]A variation is *systematic* if a physical or environmental component of variability is known to be a function of specific design characteristics [Orsh10].

2.3.2.1 Physical Variations

Physical variations are physical effects that result in changes in process parameters with time. The main members of this group are reliability-related mechanisms which change system reliability over time. Among them, failure or malfunction due to soft errors as well as aging effects have a considerable consequence on nano-scale semiconductors [Wang07a]. Soft errors are caused by ionizing radiation and have emerged as a major concern for current generation of technologies [Mukh05]. Furthermore, aging mechanisms such as hot carrier injection (HCI), negative bias temperature instability (NBTI), electromigration (EM), and oxide breakdown degrade the circuit performance in time [Sriv05, Shen12].

HCI and NBTI effects induce gradual device degradation by causing the threshold voltage of the device to rise. Electromigration may cause increased propagation delay and wire resistance due to a reduction in the width of a wire. In the worst case, it will result in wire opens and shorts causing functional failure. Oxide breakdown is an irreversible local change of the dielectric isolation property caused by increasing voltage and electric field on the oxide. A breakdown happens after a certain amount of time during which the oxide is subjected to an electrical stress at product operation or elevated conditions [Stro06].

The effect of physical variations on the system is particularly difficult to analyze, as it depends strongly on process variations and environmental conditions and becomes visible after certain time of operation, from few days to few years depending on the system and the type of physical variation. Therefore, traditional accelerated test techniques, such as burn-in [McPh06], are used. These testing techniques stress the design to operate under worst-case conditions. However, they are expensive and have a large application time [Stro06]. If these effects are not properly accounted during the design process, they may result in violation of non-functional requirements such as timing errors that become visible during operation or burn-in [Stro06].

Physical variations have a direct influence on NFPs such as performance and reliability of systems. For example, long-term performance modeling must consider the threshold voltage shift due to aging effects. Chapter 3 provides a more comprehensive background on modeling and origins of reliability related mechanisms.

2.3.2.2 Environmental Variations

Like physical variations, environmental variations affect the system during its operation time. An IC is composed of many individual circuits manufactured simultaneously and working concurrently to perform the overall expected functions. This integration means that various components and circuits share a common operating environment. This environment includes [Orsh10]

1. the silicon substrate on which various circuits are integrated; it is typically electrically resistive and is an excellent thermal conductor;
2. the package in which the integrated circuit is sealed to be protected, and the connections between the packaged circuit and the external environment through which the circuit is supplied with power as well as input and output signals.

Sharing of the IC operating environment creates various types of coupling between the individual components and circuits, including [Orsh10]

1. coupling through the power supply network, where the distributed nature of the integrated circuit leads to temporal variations in the power supply voltage;
2. coupling through the common thermal environment composed of the chip substrate and package, where differences in power density lead to temporal variations in local temperature;
3. coupling through electro-static (capacitive) or electro-magnetic (inductive) mechanisms between neighboring wires or between the wires and the semiconducting (resistive) substrate. This coupling may result in an electrical activity in one and in noise or interference in another component.

Environmental variability is largely systematic since it depends predominantly on the details of the circuit operation. Thus, the study of environmental variations focuses on the efficient prediction and bounding of such variations [Orsh10]. A common method to deal with environmental variations is to set worst-case margins for the design. However, worst-case margins reduce the performance. Furthermore, identifying specific worst-case conditions for this type of variations is extremely difficult [Sriv05, Shen12].

Beside the above mentioned environmental variables, structural level variables (e.g., the switching activity defined by input vectors) rely on workload of the processor and are hence time-dependent. Therefore, some sources [Sriv05] address them as environmental variables as well. However, they are not considered as an environmental variation in this work as they are not influenced by the environmental conditions.

2.4 Variations and Non-Functional Properties

variations change system parameters either in time or during fabrication. Changes in system parameters can affect both functional and non-functional properties of the design. Process variations can cause permanent faults in the structure of the design resulting in violated functionality. Soft errors, on the other hand, can cause temporary or permanent system failures. Additionally, the interaction between the design and process fabrication steps can affect the manufactured component's parameters. This introduces a new set of systematic variability that will affect not only the process control, but also modeling, simulation, timing and chip integration [Berg09].

Fig. 2.3 shows the mutual effect of variations on NFPs. As stated in section 2.2, NFPs are defined as characteristics that affect hardware quality or performance rather than the functionality. Time is an inseparable element of dynamic NFP models. Section 2.3 explains that runtime variations are considered as changes to hardware performance during the lifetime of a system. In other words, runtime variations can be defined as temporal changes to hardware's NFPs during the lifetime of the chip. Performance degradation due to aging mechanisms, power consumption, temperature variation, error rate, etc., are all properties which should be estimated and analyzed in non-functional domain to predict hardware behavior during its operational lifetime.

Realistic system modeling demands a comprehensive study of variation effects on NFPs. The NFP degradation due to variations should be accurately modeled during design stages [Orsh02, Merr11]. An accurate model for NFPs should consider the effect of variations on design parameters. For example, aging effects increase the threshold value of the transistor and

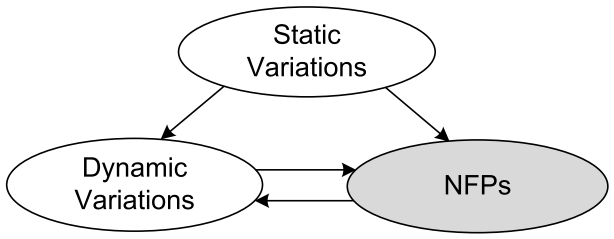


FIGURE 2.3: Variations and NFPs

therefore increase the transistor delay over time [Stro06]. The gradual increase in delay causes performance degradation (non-functional misbehavior) and finally leads to system failure (functional misbehavior). To study the effect of an aging mechanism, the process parameters which are sensitive to that mechanism should be identified and modeled. Then, the performance degradation based on the process parameter model is extracted. Using the performance model, we can predict the time in which the NFP violation, i.e., performance degradation, will lead to the functional misbehavior of the hardware.

As an example, Fig. 2.4, known as the bath-tub curve, illustrates how variations affect the reliability of fabricated chips during the predicted lifetime. This curve is usually suggested as a generic model of the system failure rate. Failures in early stages of the utilization period are usually due to variation in the manufacturing process. Due to process variations, some units are weaker than others and fail quite early. As time passes, these weaker units will disappear or be restored to a much better condition and the failure rate curve levels out. In the middle period, it is essentially only environmental high stresses that might sometimes be so severe that make a unit fail. In addition, soft errors can happen any time during the lifetime of the system. Thus, failures occur independently of the age of the unit, which explains the constant behavior of the failure rate curve. At the end of the life of a surviving unit, accumulated environmental stresses result in aged transistors which make the unit weaker and more prone to fail [Berg09].

2.5 Summary

This chapter defined and discussed non-functional properties in detail. A non-functional property is an attribute or a constraint on a system. Variations, introduced in section 2.3, change

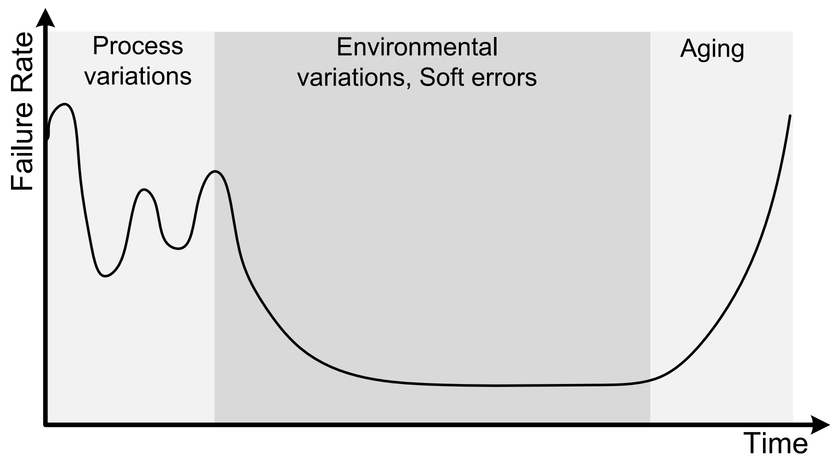


FIGURE 2.4: Bath-tub curve [Berg09]

non-functional properties either at production or during operation time of the system. Variations are either static or dynamic. Static variations are mostly caused by fabrication processes or model inaccuracies and have a permanent effect on chips. Runtime variations appear during the runtime of the system and cause permanent or temporary changes in its behavior. Accurate NFP prediction must consider the effect of variations in modeling the NFP under analysis.

Chapter 3

NFP Models

3.1 Introduction

The first step in predicting the effect of non-functional properties on a system's behavior is to clearly model them based on their underlying physical phenomena. As stated in chapter 2, this work focuses on dynamic, quantifiable NFPs. Among available NFP models of this category, this chapter discusses the architectural vulnerability to soft errors and performance degradation due to aging effects. The mentioned NFPs pose emergent challenges to today's nano-scale transistors and hence, it is important to find solutions to predict them as soon as possible in the design paradigm. In addition, they are well-defined and formally modeled in literature. Architectural vulnerability can be predicted either analytically or by simulation while performance degradation due to aging can well be predicted using analytical models. These models are later used as a reference in the sequel of this work.

3.2 Architectural Vulnerability

Vulnerability factor is the fraction of faults in the hardware that shows-up as visible errors [Mukh11]. Architectural vulnerability characterizes a system's vulnerability to soft errors[1].

[1]Energetic particles, such as neutrons from cosmic rays and alpha particles from packaging materials, generate electron-hole pairs as they pass through a semiconductor device. Transistor source and diffusion nodes can collect

The *architectural vulnerability factor* (AVF) is the probability of a fault in an SoC structure to result in a visible error in the final output of the system [Mukh03].

Soft errors are classified [Mukh11] as shown in Fig. 3.1. *Benign errors* do not lead to a system failure. *Recoverable errors* do not cause system failure, but may impact the performance. *Silent data corruption* (SDC) is the most severe failure mode, where a fault induces the system to generate erroneous outputs. To avoid SDC, designers often employ basic error detection mechanisms, such as parity. With the ability to detect a fault but not correct it, we avoid generating incorrect outputs, but cannot recover when an error occurs. Errors in this category are referred to as *detected unrecoverable errors* (DUE). Detecting an unrecoverable error allows the system to reach a safe state upon failure.

A conservative system that signals all detected faults as failures will unnecessarily raise the DUE rate by failing on false DUE events. Alternatively, if the system can identify false DUE events (e.g., a fault that corrupts only the result of a wrong-path instruction in a processor), then it can suppress the error signal.

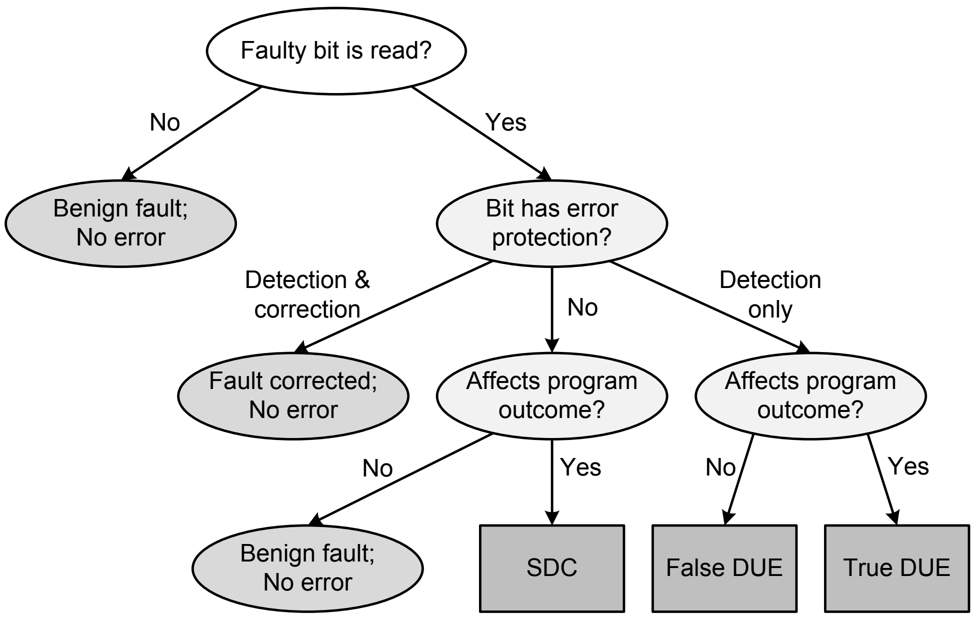


FIGURE 3.1: Classification of the possible outcomes of a faulty bit [Mukh11]

There are three known ways to compute the AVF of different hardware structures [Mukh11]: Statistical fault injection, analytical models, and performance models (simulators).

these charges. A sufficient amount of accumulated charge may invert the state of a logic device, such as an SRAM cell, a latch, or a gate, and introduce a logical fault into the circuit's operation. Because this type of fault does not reflect a permanent failure, it is termed *soft* or *transient*. [Mukh11]

Statistical fault injection (SFI) is based on executing a program repeatedly for each injected fault and detecting possible errors. An exhaustive search of this space is almost impossible as it requires a massive number of experiments spanning the total state of a silicon chip (e.g., upward of 200 million bits in a current microprocessor), potential space of benchmarks running on the chip, and the cycles in which faults can be injected. Therefore, statistical sampling is used.

SFI introduces bit flips—randomized in both time and space—into a model of the structure being studied, such as register transfer level (RTL) or a performance model. It then runs forward and compares the architectural state of the model with the state of an error-free model. After some number of simulation cycles, if the comparison does not result in a mismatch, the error is either latent or has been masked. The AVF of the structure being studied is estimated as the fraction of mismatches observed divided by the total number of bit flips introduced.

SFI is a very powerful technique and has the advantage of not requiring a prior understanding of the hardware architecture being studied. Unfortunately, it makes sense only in a very detailed model, such as RTL, which models all system state bits. RTL models are usually much slower than performance models and can realistically be run for only tens of thousands of simulated cycles per injected error. Hence, computing the AVF of an entire system would require an enormous amount of compute power to cover a sufficiently large number of injected errors.

In selected cases, when bits flow unmodified and without duplication through a structure, we can use Little's law[2] to compute the AVF. A bit is called an *architecturally correct execution* (ACE) bit when it contains information that, if changed, will affect the final output. It is called an un-ACE (unnecessary for ACE) bit otherwise. A mismatch between the models with and without errors may not necessarily mean that there is an error, because the architectural state may actually contain un-ACE bits, such as dynamically dead register values. Using Little's law, the AVF of an structure can be expressed as:

$$\text{AVF} = \frac{\text{Average number of ACE bits in a structure in a cycle}}{\text{Total number of bits in a structure}}$$

[2]Little's law states that for systems reaching steady state, the average number of customers in a system (L) is equal to the product of the average arrival rate (λ) and the average time spent in the system (W): $L = \lambda \cdot W$ [LG08].

Letting B_{ace} be the average bandwidth of ACE bits and L_{ace} be the average residence cycles of an ACE bit in a hardware structure, the above mentioned formula can be rewritten as [Mukh03]

$$\text{AVF} = \frac{B_{ace} \times L_{ace}}{bit_{total}} \tag{3.1}$$

where bit_{total} is the total number of bits in the hardware structure.

Both the SDC and DUE AVFs of various systems can be computed using a performance model. The basic idea is to distinguish the ACE and un-ACE bits. The fraction of time a bit contains ACE state is, by definition, the AVF of the bit. We refer to this process as lifetime analysis. The key challenge of lifetime analysis with a performance model is to identify the un-ACE fraction of a bit's lifetime. To provide a conservative upper bound on the AVF, we assume a bit is ACE unless it can be shown to be un-ACE. In a processor, sample instructions that give rise to an un-ACE state are dynamically dead, wrong path, and falsely predicated instructions.

Lifetime analysis requires an in-depth understanding of the architecture and microarchitecture. Otherwise, we may end up with an AVF number that is artificially too high. Chapter 7 shows how the multi-level NFP-aware simulation can overcome the shortcomings of SFI and provide an accurate AVF prediction with high simulation speed.

3.3 Wear-Out Mechanisms

All materials tend to degrade and will eventually wear out with time. Semiconductors are not an exception. Scaling has a direct effect on hardware chips' wear-out by speeding up aging processes. The rate of degradation and eventual time to failure depends on electrical, thermal, mechanical, and chemical environment to which the device is exposed. However, most systems have a reasonably well-defined lifetime. Estimating the degradation due to wear-out mechanisms during operational life of the system is crucial, to ensure that the catastrophic failure will occur well past the expected lifetime.

Wear-out mechanisms depend on several NFPs, the most significant of which are power and temperature. Many wear-out related defects are caused by a design or a production problem which accelerates physical processes. Estimating the degradation due to aging at early design phases enables aging-aware design and ensures the reliability of the system during the predicted lifetime. Studying the physics behind these mechanisms and the NFPs which affect or enforce aging effects helps to find effective ways to predict the system behavior at early design phases.

This section gives a brief overview of the dominant wear-out mechanisms as important reliability challenges of today's semiconductors.

3.3.1 Negative Bias Temperature Instability

The instability of transistor parameters, e.g., V_{th}, under negative bias temperature instability (NBTI) effect has been known since long ago, but the recent aggressive scaling of CMOS technology has made NBTI one of the foremost reliability concerns in today's nanoscale design. The NBTI effect manifests itself as an increase in the PMOS threshold voltage and hence degrades the circuit performance [Stro06].

There is a significant discrepancy on the nature of NBTI damage [Cao11, Stro06], but it is arguably believed that NBTI mechanism is caused by broken *Si–H* bonds. These bonds are induced by positive holes from the channel and hydrogen, in a neutral molecular form (H_2), diffuse away from the interface. This process leaves positive interface traps (N_{it}) (i.e., from $Si+$), which cause the increase of threshold voltage, V_{th}, in PMOS transistor (V_{tp}). The gradual increase in the magnitude of V_{tp} results in circuit performance degradation over time [Cao11].

The negative threshold voltage shift and the degradation in device current, due to NBTI, increase with exponential dependency on oxide thickness [Alam05, Chak04, Kris05]. At higher frequencies (higher than 100 Hz), NBTI degradation is frequency independent [Alam03, Vatt06], but increases with supply voltage (V_{dd}) and temperature. The affected PMOS transistor has higher V_{tp} than a transistor in normal inversion. NBTI effect is accelerated by relatively low negative voltages (E_{ox}—electric field in the PMOS gate dielectric—$<$ 6 to 8 MV/cm) and higher temperatures [Mass04, Giel08]. If the NBTI stress voltage is removed ($V_{in} = V_{dd}$), a fraction of NBTI degradation is recovered by annealing at high temperatures [Schr03, Stro06, Abad03].

NBTI stress and recovery phases are best expressed by *reaction-diffusion (R–D)* models [Turi52]. In the R–D formulation of NBTI degradation, it is assumed that NBTI arises due to hole-assisted breaking of *Si–H* bonds at the Si/SiO_2 interface. Based on this model, two critical steps happen [Alam07, Cao11]:

- *Stress*: Some *Si–H* bonds at the substrate/gate dielectric interface are broken under the electrical stress. Positive holes trigger this reaction. Consequently, interface charges are induced, which cause the increase of V_{tp}. Given the initial number of the *Si–H* bonds at the Si/SiO_2 interface (N_o), and the concentration of the inversion carriers (P), the generation rate of the interface traps (N_{it}) at time t due to NBTI stress is given by

$$\frac{dN_{it}}{dt} = k_F(N_o - N_{it})P - k_R N_H N_{it} \tag{3.2}$$

where k_F and k_R are the reaction rates of the forward and reverse reactions, respectively. N_H is the hydrogen concentration at the interface. P is expressed as

$$P = C_{ox}(V_{gs} - V_{th})$$

k_F is proportional to the number of inversion layer holes that are captured by *Si–H* bonds. The *Si–H* bond is weakened once a hole is captured and assisted by the electric field, it is easily broken at relatively moderate temperatures. The broken *Si* bonds act as a donor trap and contribute to the shift in the threshold voltage and reduction in transconductance. The generation rate is an exponential function of the electrical field and temperature. It is also proportional to the density of reaction species, namely holes or hot electrons.

Trap generation during the initial period of the stress phase is slow ($\frac{dN_{it}}{dt} \approx 0$ and $N_{it} \ll N_0$), therefore, equation 3.2 can be reduced to [Wang07b]:

$$N_H N_{it} \approx \frac{k_F}{k_R} P \cdot N_0 \tag{3.3}$$

With continued forward reaction, H is produced and two H atoms combine to generate a hydrogen molecule. The concentration of H_2, i.e., N_{H_2}, is related to the concentration of H, i.e., N_H, using

$$N_{H_2} = k_H N_H^2 \tag{3.4}$$

The total hydrogen can be divided into two parts: (1) Hydrogen in the oxide and (2) hydrogen in the poly-*Si*. Therefore, N_{it} can be calculated as [Wang07b]

$$N_{it} = 2\int_0^{t_{ox}} N_{H_2}(x)dx + 2\int_{t_{ox}}^{\sqrt{D_{H_2}t}+t_{ox}} N_{H_2}(x)dx \tag{3.5}$$

where t_{ox} is the oxide thickness. D_{H_2} is the H_2 diffusion constant and $N_{H_2}(x)$ is the concentration of H_2 at the Si/SiO_2 interface. Solving the above equation and substituting equation 3.5 in $\Delta V_{th} = qN_{it}/C_{ox}$, we obtain the general form of V_{th} degradation as

$$\Delta V_{th}(t) = (K_v(t-t_0)^{1/2} + \sqrt[2n]{\Delta V_{th}(t_0)})^{2n} \tag{3.6}$$

where

$$K_v = (\frac{qt_{ox}}{\varepsilon_{ox}})^3 K_1^2 C_{ox}(V_{gs} - V_{th})\sqrt{C}\exp(\frac{2E_{ox}}{E_0}) \tag{3.7}$$

C has a temperature dependence as $C = T_0^{-1}\exp(E_a/kT)$, k is the Boltzmann's constant and T_0 is another constant. Although the aforementioned result was derived assuming H_2 as the diffusing species, a similar dependence can be obtained if the diffusing species is assumed to be H. For a H_2 diffusion-based model, n is $1/6$, and for a H-based model, $n = 1/4$ [Wang07b].

- *Recovery*: This is where reaction generated species diffuse away from the interface toward the gate, driven by the gradient of the density. This process influences the balance of the reaction and is governed by

$$\frac{dN_H}{dt} = D_H \frac{d^2 N_H}{dx^2} \tag{3.8}$$

where D_H is the diffusion constant. The solution of equation 3.8 exhibits a *power-law* dependence[3] on the stress time [Alam07]. The exact value of the power-law index indicates the type of diffusion species. The closed-form solutions to equation 3.8 shows this dependence:

$$N_{it} = \sqrt{K^2 \cdot t^{2n} + N_{it0}^2} \tag{3.9}$$

where N_{it0} is N_{it} at the starting point and n is the H_2 diffusion constant. Considering the reaction of breaking *Si–H* or *Si–O* bonds, the generation rate, K, is linearly proportional to the hole or electron density and exponentially dependent on the temperature (T) and the electric field (E_{ox}). Therefore,

$$K \propto \sqrt{C_{ox}(V_{gs} - V_{th})} \cdot \exp(E_{ox}/E_0) \cdot \exp(-E_a/kT) \tag{3.10}$$

where $E_{ox} = (V_{gs} - V_{th})/t_{ox}$. E_0 and E_a are technology-independent characteristics of the reaction.

In the recovery phase, due to the absence of holes, there is no net generation of interface traps. The hydrogen species that were generated during the stress phase continue to diffuse away from the interface toward the poly-*Si*. At the same time, some of the hydrogen species that are closer to the interface diffuse back and repassivate the broken $Si+$ bonds. This results in the reduction of the H_2 density, $N_{H_2}^A$. This is because the H/H_2 diffusion is faster in the oxide, and very quickly anneals the broken *Si–H* bonds.

The *H*/H_2 density is much higher in the poly-*Si* than in the oxide. Let t_1 be the time for the recovery to be applied after stress and $N_{it}(t_1)$ be the number of interface charges at the end of the stress cycle. Let $N_{it}^A(t)$ be the number of charges annealed at time t. Due to the widely different diffusivity of H_2 in the oxide and poly-*Si*, the recovery becomes a two-step process, with fast recovery driven by H_2 in the oxide, followed by slow recovery of H_2 by back-diffusion from the poly-*Si*. The number of annealed traps can be due to two parts: (1) Recombination of H_2 in the oxide and (2) back-diffusion of H_2 in the poly-*Si*. Thus [Wang07b]

[3]A *power-law* is a type of probability distribution. If the frequency (with which an event occurs) varies as a power of some attribute of that event (e.g., its size), the frequency is said to have a power-law dependence [Clau09].

$$N_{it}^{A}(t) = 2N_{H_2}(0)(\xi_1 t_e + \frac{1}{2}\sqrt{\xi_2 C(t-t_1)}) \tag{3.11}$$

where ξ_1 and ξ_2 are the back-diffusion constants. Depending on the duration $t-t_1$ of the recovery, the effective oxide thickness t_e is either equal to t_{ox} or to the diffusion distance of hydrogen in the initial stage of recovery. Here, we define the time when all the hydrogen species in the oxide are recombined with the interface traps as t'. This corresponds to the time taken by the diffusing species to diffuse to a distance of t_{ox}. If $t-t_1 \geq t'$, t_e is equal to t_{ox}; otherwise, t_e equals the diffusion distance of hydrogen in the oxide. From equation 3.11, we obtain

$$N_{it}^{A}(t) = N_{it}(t)(\frac{2\xi_1 t_e + \sqrt{\xi_2 C(t-t_1)}}{(1+\delta)t_{ox} + \sqrt{Ct}}) \tag{3.12}$$

where δ is the correction factor. Using $\Delta V_{th} = qN_{it}/Cox$, we obtain the recovery equation as

$$\Delta V_{th}(t) = \Delta V_{th}(t_1)(1 - \frac{2\xi_1 t_e + \sqrt{\xi_2 C(t-t_1)}}{(1+\delta)t_{ox} + \sqrt{Ct}}) \tag{3.13}$$

In practice, the change in the threshold voltage (ΔV_{th}) due to NBTI is predicted by simulating the stress and recovery cycles for $m = t/T_{clk}$ cycles to obtain the long term degradation, where T_{clk} is the clock period and t is the duration of the simulation, i.e., the life time of the system. However, for high performance circuits, m can be very large, even for t = 1 month. Thus, it becomes impractical to perform cycle-to-cycle simulation in order to predict ΔV_{th}. A closed form for the upper bound on the long term ΔV_{th} as a function of the duty cycle α, clock period T_{clk} and time t can be expressed as [Cao11]

$$\Delta V_{th} = \frac{\sqrt{K_v^2 \alpha T_{clk}}}{1 - \beta_t^{\frac{1}{2n}}} \tag{3.14}$$

where n is a constant equal to $\frac{1}{6}$ for H_2 diffusion. K_v is the one from equation 3.7. β_t is expressed as

$$\beta_t = 1 - \frac{2K_1 + \sqrt{K_2 C(1-\alpha)T_{clk}}}{(1+\delta)t_{ox} + \sqrt{Ct}} \tag{3.15}$$

where K_1 and K_2 are constants on the effective oxide thickness and oxide capacitance, respectively.

3.3.2 Hot Carrier Injection

Hot carrier injection (HCI) is a major oxide failure mechanism. It occurs when the transistor's electric field at the drain-to-channel depletion region is too high. HCI degradation is mainly due to interface-state generation caused by "hot carriers". Hot carriers are particles accelerated by high electric fields and obtained very high kinetic energies through scattering and/or impact ionization. These particles are injected into the gate oxide causing interface-state generation. As the design of MOSFET transistors allows large electric fields at operating conditions, HCI degradation is a critical reliability concern. HCI leads to threshold voltage shifts and transconductance degradation of MOS devices. It alters circuit timing and high-frequency performance [Stro06, Fang98].

NMOS HCI degradation is mainly due to interface state generation, when the transistor is stressed at the peak substrate current bias condition. PMOS degradation is mainly due to electron trapping in the gate oxide when stressed at peak gate current conditions.

Fig. 3.2 sketches an NMOS transistor cross section showing the drain depletion field. The horizontal electric field in the channel, ξ_{ch}, gives kinetic energy to the free electrons moving from the inverted portion of the channel to the drain. When the kinetic energy is high enough, electrons strike *Si* atoms around the drain-substrate interface causing impact ionization. Electron-hole pairs are produced in the drain region and scattered. Some carriers go into the substrate, causing an increase in substrate current, I_{sub}, and a small fraction have enough energy to cross the oxide barrier and cause damage.

The energy follows Boltzmann distribution in which the particle's thermal energy is

$$E_t = kT/q$$

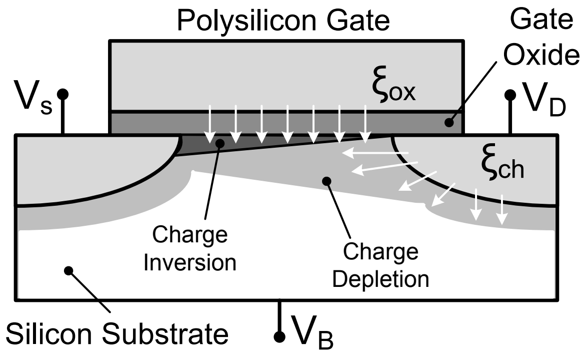


FIGURE 3.2: Saturated state NMOS transistor and its internal electric fields ξ_{ch} and ξ_{ox} [Segu04]

where k is the Boltzmann constant, T is the degree kelvin, and q is the electron charge. Carrier mobility increases as the temperature decreases. Carriers with higher mobility create hot holes and electrons more efficiently, so that HCI increases as temperature is lowered. Once the hot carrier enters the oxide, the vertical oxide field ξ_{ox} determines how deeply the charge will go. If the drain voltage is positive with respect to the gate voltage, then holes entering the oxide near the drain are accelerated deeper into the oxide, and electrons in the same region will retard from leaving the oxide interface. ξ_{ch} restricts the damage to oxide over the drain-substrate depletion region, with only a small amount of damage just outside the depletion region.

HCI is a dynamic effect and depends strongly on circuit design, fanout, and input waveforms [Fang98]. The typical parameters affected by HCI are I_{Dsat}, transistor transconductance (g_m), threshold voltage (V_{th}), weak inversion subthreshold slope (S), and increased gate-induced drain leakage ($GIDL$). I_{Dsat} is a function of V_{th} and most closely approximates the impact on circuit speed, since it influences the charge and discharge of load capacitors.

Similar to NBTI, HCI can also be expressed by the reaction-diffusion model. In the reaction phase, hot electrons broke some $Si-H$ or $Si-O$ bonds at the transistor's substrate/gate. Consequently, interface charges are induced, which cause an increase in V_{th}. Akin to other reactions, the generation rate is an exponential function of the electrical field and temperature. In the diffusion phase, the reaction-generated species diffuse away from the interface toward the gate, driven by gradient of the density. HCI impacts primarily the drain. This process influences the balance of the reaction as is governed by equation 3.8 which exhibits a power-law dependence on time. However, the recovery phase in HCI is negligible [Wang07b]. Based on the reaction-diffusion model, the threshold voltage degradation is usually expressed as [Wang07b]

$$\Delta V_{th} = \frac{q}{C_{ox}} K_2 \sqrt{Q_i} \exp(\frac{E_{ox}}{E_{o2}}) \exp(-\frac{\phi_{it}}{q\lambda E_m}) t^{n'} \tag{3.16}$$

where q is the electrical charge. C_{ox} is the oxide capacitance and λ, K_2, ϕ_{it} and E_{o2} are technology dependent constants. n' is the time exponential constant for HCI. E_{ox} is defined as in section 3.3.1. Q_i is the inversion charge.

3.3.3 Dielectric Breakdown of Gate Oxides

Scaling has a profound effect on gate oxide reliability. With ultra thin oxides, gate tunneling current is also increasing. This gate leakage current increases power consumption and imposes a practical bound on oxide thickness which accelerates wear-out due to time-dependent dielectric breakdown (TDDB).

TDDB (also known as oxide breakdown) is an irreversible local change of the dielectric isolation property. Each time a transistor has a voltage put across its gate oxide, a small amount of charge is injected into the oxide. The oxide wear-out time decreases as the oxide stress increases. The voltage and electric field of thin oxide will cause premature oxide wear-out. Oxide field strength accelerates electrons across the oxide. Significant tunneling of electrons through the gate oxide can occur when the oxide thickness becomes less than about 40 Å. A breakdown happens after a certain amount of time during which the oxide is subjected to an electrical stress at product operation or elevated conditions [Segu04, Stro06].

Due to the explained physical process, the complete oxide breakdown procedure is divided into three levels [Segu04]:

- a slow trap generation within the oxide (aging) until a defect path links the gate terminal to the substrate;
- a soft breakdown (SBD) at low voltages that permanently increases gate current and gate noise;
- the appearance of a hard breakdown (HBD), showing continuous exponential increase in gate current.

Trap generation is the key factor that determines the oxide degradation and breakdown. Three general models are discussed in the literature for trap generation: The "anode hole injection" (AHI), the "thermo-chemical", and the "anode hydrogen release" (AHR) models [Bern06].

In the AHI model, electrons injected from the gate metal cathode into the oxide undergo impact ionization events, which generate holes in the process. Some of these holes tunnel back into the cathode and create electron traps in the oxide [Bern06].

The thermo-chemical model states that defect generation is a field-driven process and the current flowing through the oxide plays a secondary role at most. The interaction of the applied electric field with the dipole moments associated with oxygen vacancies leads to a conduction sub-band formation and to severe Joule heating[4] at the stage of oxide breakdown [Bern06].

In the AHR model, the energy release of the incoming electrons at the anode may activate hydrogen release at the anode, beside creating holes. The released hydrogen diffuses through the oxide and can generate electron traps.

There have been contradicting opinions on the exact field acceleration law of time-to-breakdown, t_{BD}. According to the AHI model, the field dependence of the t_{BD} takes the form:

$$t_{BD}(t) \sim \tau_0 \exp(\frac{G}{E_{ox}}) \tag{3.17}$$

where E_{ox} is the electric field across the dielectric and τ_0 and G are constants.

According to the thermo-chemical model, the field dependence of the t_{BD} is of the form:

$$t_{BD}(t) \sim t_0 \exp(-\gamma E_{ox}) \tag{3.18}$$

where t_0 and γ are constants.

[4]Joule heating is the process by which the passage of an electric current through a conductor releases heat [vM06].

For ultra-thin oxides, gate voltage (V_g) is the primary driver of the breakdown process. Additionally, temperature dependence of ultra-thin oxides is non-Arrhenius[5], but the temperature acceleration factor is rather larger at higher temperatures. Considering these dependencies, the t_{BD} for ultra-thin oxides ($T_{ox} < 32\mathring{A}$) can be presented as [Mons01]

$$t_{BD} = t_0 \exp(\gamma(\alpha T_{ox} + \frac{E_a}{kT_j} - V_g)) \tag{3.19}$$

where γ is the acceleration factor, E_a is the activation energy, α is the oxide thickness acceleration factor, T_0 is a technology related constant, k is the Boltzmann constant, and T_j is the average junction temperature.

3.3.4 Electromigration

Under the influence of electron flow and temperature, metal atoms become thermally active and force out of their lattice sites. The atoms move under diffusion in the same direction as the electrons. When a metal atom is knocked from its lattice site while downstream, a small tensile stress is created. The displaced metal atom creates compressive forces and possible extrusions. A metal line will fail if sufficient current density and high temperature are applied. This effect is called electromigration (EM) [Stro06, Segu04].

Several variables affect EM. The flux J is the number of particles (atoms in this case) crossing a unit area per unit time. Atomic flux, J_{em} is expressed as

$$J_{em} = vN \tag{3.20}$$

where v is the mean velocity of metal atoms and N is the concentration of moving metal atoms. v is a function of its mobility, μ and electric field, ξ, and is calculated as $v = \mu\xi$ where μ is

[5]Arrhenius equation is a simple, yet accurate formula that shows the influence of temperature on the rate of chemical reactions. According to Arrhenius equation, a rate constant k is the product of a frequency factor A and an exponential term: $k = A\exp(\frac{-E_a}{k_B T})$ where E_a is the activation energy, k_B is the Boltzmann constant and T is the temperature [Laid84].

$$\mu = \frac{Q\,D}{k\,T} \tag{3.21}$$

Q is the effective charge and is calculated as $Q = Zq$ where Z is the electron-to-atom ratio and q is the electron charge. D is the self-diffusion coefficient:

$$D = D_0 \exp(\frac{-E_a}{kT}) \tag{3.22}$$

where k is the Boltzmann constant, T is the absolute temperature and E_a is the activation energy. Using equations 3.20, 3.21 and 3.22, the final flux expression can be expressed as

$$J_{em} = \frac{D}{kT}(Zq)\rho\, j_e N \tag{3.23}$$

In reality, J_{em} will not be the same throughout the metallization because of structural inhomogeneity. That constitutes a divergence of flux in the metallization which is more obvious under high density conditions. As this divergence becomes more significant, the original isentropic self-diffusion[6] is perturbed and the ions moving along the current flow will have the greatest probability of exchanging positions with the vacancies. The original random process changes to a directional process in which the metallic ions move downstream opposite to the electron wind direction while the vacancies move in the opposite direction. The metallic ions condense to form whiskers, whereas the vacancies condense to form voids. This process results in change in the density of the metal with respect to time. The formation of voids causes some of the metallization lines to fail. The failed lines force the current to go through the rest of the lines resulting in an increase in the current density and the Joule heat. The produced Joule heat increases the local temperature and causes more lines to fail. The increase of current density will also cause more lines to fail. Besides, as the whiskers and hill-locks form at the other end, a concentration gradient is produced which may create a stress-related force, thus enhancing the mass transport process and causing more lines to fail [Goel89]. The time to failure due to EM

[6]An isentropic process is one in which the process takes place from initiation to completion without an increase or decrease in the entropy of the system, i.e., the entropy of the system remains constant [VW86].

is usually calculated by the empirical equation known as “Black law” [Blac67] and is expressed as [Segu04, Goel89]

$$t_F = \frac{A_0}{J_{em}^2} \frac{\exp(E_a)}{kT} \tag{3.24}$$

where t_F denotes the time to failure. A_0 is a technology constant based on the cross-sectional area of the interconnect, while k is the Boltzmann constant. T is the temperature, j_e is the electron current density and E_a is the activation energy (eV) for EM failure. It is clear that J_{em} and the temperature are deciding factors in the design process that affect EM.

3.4 Summary

Accurate estimation of non-functional properties demands a comprehensive knowledge of the physics behind them. This chapter presented the modeling of two sample NFPs, architectural vulnerability and performance degradation due to wear-out mechanisms. Architectural vulnerability is becoming more important as the transistors shrink. Vulnerability prediction at early design phases can provide useful hits for fault-tolerant design before manufacturing. Wear-out mechanisms are exacerbated by technology scaling. They deteriorate system performance and have a destructive effect on several NFPs, including reliability. Aging effects and their impact on NFPs can partially be avoided by proper design and production strategy.

Chapter 4

NFP Simulation

4.1 Introduction

Simulation is the most widely used technique for predicting the functional as well as non-functional behavior of hardware at different abstraction levels [Cham95]. NFPs with high dependency to circuit structure and workload, such as power consumption [Bona04, Zhan09, Boli97], testability [Koch10, Bara11, Hsia95] and heat distribution on the chip [Brya10], can be best studied by simulation.

System simulation starts early in the design's life-cycle. It is usually customized to capture not only the system functionality, but also all relevant system observables (as will be described later in chapter 6). However, over long simulation periods, a system-wide analysis with fine grain NFP models is computationally expensive (if feasible at all).

Hardware design representation has been raised in different dimensions through electrical, switch, gate, register transfer, and transaction levels. Design methodology selects the proper level of abstraction at different design phases to assure the design quality: If the level of abstraction is not sufficiently high, it may be difficult to vary the ultimate design implementation to evaluate different alternatives. In addition, the analysis of alternative design solutions with respect to different NFPs at a given abstraction level reduces the probability of not fulfilling certain requirements after refinement through lower levels, leading to redesign [Gajs09, Rajs00].

Non-functional properties are usually modeled at low design abstraction levels. The properties such as leakage power, temperature, interconnect delays, yield, and cost can best be analyzed at electrical or switch level [Nico12]. Widely used fault models to analyze the design testability are mostly defined at electrical, switch or gate level [Wang06]. Simulation at these levels provides accurate results for NFP prediction, but it is not applicable for complex SoCs and long simulation runs. This implies that the gap with low level effects inherent to very deep sub-micron technologies is widening: High level NFP models are usually either not available, due to the nature of the NFP, or do not provide the required accuracy. Conversely, low level simulation requires the complete description of the system at that level which is usually not available at early design phases. A trade-off between the abstraction level and the NFP-aware simulation accuracy is required to preserve the accuracy of obtained observables in a reasonable simulation time.

This chapter discusses multi-level simulation as an efficient tool for NFP analysis. Multi-level simulation can switch between abstraction levels to benefit from accuracy of low and simulation speed of high level models. It can be used effectively for early design exploration where the complete system is not available at a certain lower level. We first address the design paradigm by discussing the main abstraction levels for hardware design in section 4.2. In section 4.3, multi-level simulation is discussed in detail and the state-of-the-art for available multi-level simulation approaches for NFP prediction are presented. Section 4.4 discusses parallel simulation which speeds up traditional simulation. Section 4.5 concludes the chapter with a brief summary.

4.2 Design Abstraction Levels

The primary driving factor behind the exponential increase of IC functionality over the past 40 years has been the ability to continually scale MOS devices to smaller dimensions [Doer07]. To manage the emerged complexity, the key method is to describe the system resorting to several levels of abstraction [Gajs09, Chu06]. An *abstraction* is a simplified model of the system, showing only the selected features and ignoring the associated details [Chu06]. The purpose is to reduce the amount of data to a manageable level, so that only the critical information is presented. The history of electronic hardware design has followed a path of increasing levels of abstraction much like many other technologies. As the lower levels of the technology are understood, they become the building blocks of higher levels, allowing designers to abstract details at that level and devote the design effort to more global and presumably important issues [Sang12, Chu06].

A high level abstraction contains only the most vital data. A low level abstraction is more detailed and takes account of previously ignored information. Despite complexity, the low level abstraction model is more accurate and is closer to the real circuit [Chu06].

In the following, the hardware abstraction levels which are widely used in this work are briefly discussed. More information on hardware abstraction levels and design dimensions can be found in [Gers09].

4.2.1 Electrical Level

Electrical level refers to modeling hardware at its lowest conceptual level [Russ89]. The circuit is modeled with several physical details describing transistors, wires, capacitors and resistors and their respective interconnectivity. The fundamental step of electrical level modeling is the sizing of each transistor to achieve the required specifications. A good electrical model is based on physical behavior covering significant physical effects such as non-uniform doping effects, mobility effects, velocity saturation, short/narrow channel effects, substrate current, thermal/flicker noise, and temperature effects. Beside the correct $I-V$ characteristics, the electrical-level model should also consider the correct current derivatives, i.e., transconductances. In addition, the correct best/worst case modeling must be guaranteed for design robustness [Stef08].

Physical phenomena that change NFPs by affecting transistor parameters during the lifetime of the system, e.g., aging effects, can be best modeled at this level. Process variation effects on device characteristics can also be studied at this level [Sriv05]. Transistor fault models (stuck-open and stuck-short) can be applied and evaluated accurately, while transmission gate and tri-state buffer faults can also be tested at this level. A defect-based test methodology is more effective with an electrical level model of the circuit as it contains more detailed structural information than higher abstraction levels and will yield a more accurate defect coverage analysis [Wang06].

Electrical level models can predict NFP observables with high accuracy, however, the slow simulation speed and complexity of these models make them inefficient for complex hardware systems. On the other hand, electrical level modeling is possible only when all the technology parameters are decided. Therefore, early NFP prediction at this level is unfeasible.

4.2.2 Switch Level

Switch level modeling provides a level of abstraction between gate and electrical level. The primitives here include MOS, resistive, bidirectional and resistive bidirectional switches, pull-ups, pull-downs and power and ground nets. A switch level model describes the interconnection of transmission gates which are abstractions of individual MOS and CMOS transistors. While electrical models deal with analog input and output signal values, the switch level transistors are modeled as being either on or off, conducting or not conducting. Further, the values carried by the interconnections are abstracted from the whole range of analog voltages or currents to a small number of discrete values. These values are referred to as signal strengths [Thom08].

Although switch level models are an abstraction of electrical level models, they still include enough information to provide an accurate NFP analysis for several NFPs. However, as for electrical level models, unavailability of the models or technology parameters at early design phases and their slow simulation speed prevents early NFP prediction at this level.

4.2.3 Gate Level

At gate level, the circuit is defined as a netlist with logic gates and their interconnections. Typical building blocks include simple logic gates, such as *and*, *or*, *xor*, 1-bit 2-to-1 multiplexer, and basic memory elements. The timing information is also simplified at this level. A single discrete number, known as the *propagation delay*, which is defined as the time interval for a system to obtain a stable output response, is used to specify the timing of a gate [Chu06].

Structure-dependent NFPs such as timing behavior of the circuit, power and random-pattern testability are well modeled and analyzed using available gate level tools [Kell08]. In the area of testability, the stuck-at fault model is usually employed to evaluate the effectiveness of the input stimuli used for simulation-based design verification. In addition, delay fault models and delay testing have been traditionally based on the gate level description [Wang06].

However, lifetime NFP prediction at gate level is not practical for several NFPs in deep submicron technologies. In spite of high accuracy, increasing complexity makes design exploration and analysis of the complete system at gate level error prone, time consuming, and inefficient [Wang06].

4.2.4 Register Transfer Level

At *register transfer level* (RTL), the basic building blocks are modules constructed from simple gates. They include functional units such as adders and comparators, storage components such as registers, and data routing components such as multiplexers [Chu06]. RTL models a synchronous digital circuit in terms of the flow of digital signals (data) between hardware registers and the logical operations performed on those signals [Gers09].

An RTL model uses a common clock signal in the storage components. The clock signal functions as a sampling and synchronizing pulse, putting data into the storage component at a particular time, normally the rising or falling edge of the clock signal. In a properly designed system, the clock period is long enough so that all data signals are stabilized within the clock period [Chu06].

The data representation at RTL becomes more abstract than gate or switch level. Signals are frequently grouped together and interpreted as a special kind of data type, such as an unsigned integer or system state. The RTL description in behavioral domain uses an extended finite state machine (FSM) and general expressions to specify the functional operation and data routing [Chu06].

Some works in literature tried to lift various NFP models to RT or higher abstraction levels. Traditional NFP evaluation methods cannot effectively (if not at all) handle designs employing blocks for which implementation details are either unknown or subject to change. However, approaches for dominant NFPs such as power [Maci04, Ravi03, Jian98, Wang03], temperature [Sadr12, Engl08, Velu05] and testability [Sant03, Makr98, Yin01] at RT or higher levels have been proposed. These methods provide faster NFP evaluation using high level simulation speed with the cost of accuracy.

4.2.5 Transaction Level

Electronic System Level (ESL) is a set of design methodologies that enable embedded system design, verification, and debugging through the hardware and software implementation of SoCs [Mart07]. ESL design methodologies elevate design and verification to higher abstraction levels to overcome complexity. At high levels, many engineering tasks and design optimizations are successfully accomplished quicker and cheaper than at lower levels such as RTL.

Transaction Level Modeling (TLM) is an ESL design methodology to facilitate simulation-driven design space exploration and design verification [Ghen05, Gajs09]. It also allows early software development to reduce the time-to-market. After identifying a suitable (optimized) system architecture, the initial TLM can be refined by adding details about computation components and communications, leading to more precise models in terms of behavior and timing, and ultimately to a cycle-accurate description of the system [Donl04].

In TLM, the communication details among the computation components are separated from the components' implementation details, while the implementation details of both are abstracted. Communication is modeled as channels. Transaction requests take place by calling interface functions of channel models. Unnecessary details of communication and computation are hidden and may be worked out later [Ghen05].

TLM allows flexibility in temporal modeling [Ghen05], while the accuracy can be dynamically adapted during simulation [Rade08]. Fig. 4.1 shows the range of temporal flexibility that TLM offers [Blac10, Cai03]. The x-axis represents the abstraction level of the communication while the y-axis is the abstraction level of model's functionality. In loosely-timed transaction-level models, the temporal behavior is managed at the level of transactions. In the more accurate approximately-timed TLM, the time is managed at sub-transaction level, i.e., it is updated for each transaction phase, such as request, transfer, and acknowledgment. This implies that the functionality can be refined independent of the model or logical block interface or communication [Blac10, Cai03].

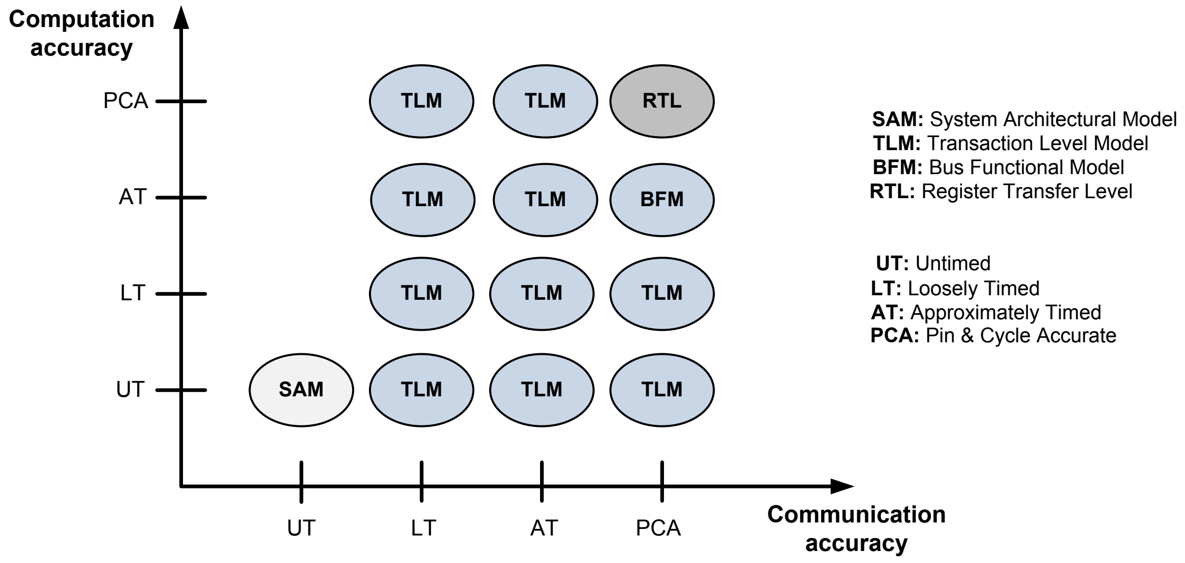


FIGURE 4.1: TLM model mapping [Blac10]

The modularity and separation of communication and functionality in TLM allow quick exploration of different architecture alternatives early before the implementation details of the design are explored [Gajs09, Ghen05]. It enables the user to change system components and communication models, to modify their interconnection patterns, and to change the mapping of application tasks to system components without much effort [Cai03].

TLM abstracts signal level communications and models complex communication operations as atomic transactions. Therefore, the number of events to be processed by event-driven simulators and the number of context switches between simulation processes are reduced. This event reduction enables system level design and simulation of large hardware/software systems and provides significant improvement in simulation performance by orders of magnitude [Cai03].

TLM can be exploited not only for architecture exploration and validation, but also for test exploration [Koch09], and reliability analysis [Hata12] for which traditional simulation is too slow. Such NFPs demand a significant amount of infrastructure to be integrated together with the system implementation. TLM provides enough details to help with important design decisions considering NFPs such as performance, die area and power [Hwan08, Chee06], although the accuracy should be carefully measured to prevent wrong design choices [Gers09]. Low level observables which are mandatory for accurate prediction of several NFPs can not be obtained at this level of abstraction.

4.3 Multi-Level Simulation

The simulation process is carried out at several abstraction levels. Each level entails a different trade-off between simulation accuracy and computing requirements [Wang09, Russ89]. High level simulation is used to check the behavioral (functional) aspects of the design, whereas the lower level simulation relates to design aspects associated with the physical implementation. As one descends through the abstraction levels, signal representation changes from character representation to discrete logic-level and then to quasi-continuous voltage levels. In a similar way, *event*s, i.e., an instantaneous occurrence that may change the state of the system, are described as number of clock cycles at the high and as sub-multiples of seconds at the lower levels [Russ89].

Complex digital systems require sophisticated simulation environment for performance prediction and correctness validation [Baga08]. As we move through abstraction levels, the simulation speed increases while several unnecessary details are abstracted. Consequently, the simulation accuracy decreases. Multi-level simulation is a method of integrating two or more independent simulations at different abstraction levels. It is used when a trade-off between the simulation accuracy and the speed up is required. Multi-level simulation permits simulation at more than one level in the same environment [Ghos86]. It accelerates the simulation by moving some parts of the system to higher abstraction levels while keeping the necessary cores at lower levels.

Fig. 4.2 shows the multi-level simulation concept. The system is modeled at high level while the design under analysis (DUA) is modeled at low level. A protocol adapter connects the two abstraction levels and facilitates the communication. The simulation procedure begins with the high level model. When an interesting region of the simulation is reached, the high level model is switched to the detailed, low level model. The detailed model can be switched back to a less detailed, high level model to quickly simulate unimportant portions of the workload. Switching back and forth may take place repeatedly and can involve multiple abstraction levels with different speed-detail characteristics. The simulator outputs the state of the simulated object at one or more predefined time points. Interaction between different abstraction levels results in an integrated simulation output without affecting the individual simulation states [Dewe90, Baga08].

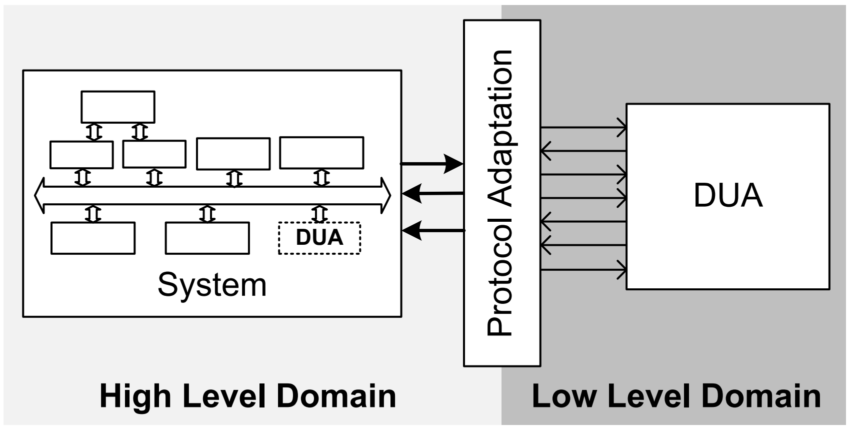


FIGURE 4.2: Multi-level simulation concept

The concept of multi-level modeling has found widespread use for design validation [Paul02, Zurc68], fault simulation [Bara11, Belt09, Leve03, Meye93], as well as NFP prediction (e.g., reliability [Suta12], power [Broo00, Shim99] and performance degradation due to aging effects [Sidd10]).

In [Paul02], the authors simulate both the hardware and the software cores and the network-on-chip (NoC) based communication using multi-level simulation to perform system-level design exploration for network processors. Wrappers translate high level signals to low level and the other way round wherever necessary. The range of abstraction levels the method supports is restricted to the ability of the adopted hardware description language (HDL).

In [Ghos87] approach, hardware simulation and verification is initiated at the behavioral RTL. When detailed results are required for one or more high level components or for tracing the source of an error to any of the low level devices, the appropriate components are expanded into their low level implementation and the simulation or verification initiates at that level. Upon completion of the execution, the control returns to the high level model. Consequently, only sub-parts that are dynamically warranted by the high level simulation results are selected for expansion and detailed simulation.

A multi-level simulation methodology at switch/RT level for system verification is proposed in [Tham84]. The full chip is modeled at pure RTL. A mixed RTL-schematics model is represented as transistors. RTL modules correspond to the schematics sheet being verified. The verification is done by comparing the output of the circuit under analysis at two abstraction levels. If a discrepancy is found, the models at both switch and RT level would be debugged and re-simulated.

In the area of fault simulation, numerous approaches for the different abstraction levels have been proposed. In [Leve03], a multi-level fault injection method is introduced. The method uses two functionally equivalent models at architecture and RT level. The fault is injected at both levels and its effect on signals (at RTL) and system states (at architectural level) is studied. Later in [Leve04], the authors perform serial fault injection at RTL with error propagation at system level. Injection of structural faults into mixed gate/high level SystemC models is presented in [Mise08]. Beltrame et al. [Belt09] perform the mutator-based fault injection at RT and transaction level.

Structural fault simulation at gate/architecture level is discussed in [Hsia95, Sina01]. Moreover, Meyer et al. [Meye95] represent a multi-level, hierarchical fault simulation methodology which switches between switch, gate, and RT level while the goal is to achieve highest fault coverage at switch level. The fault simulation starts at RTL. As soon as a faulty component is recognized, the simulation continues at gate level. The faulty gates at gate level are simulated at switch level, likewise. The presented tool switches between the three abstraction levels to force simulation at higher levels without losing the switch level accuracy. Similarly, the authors in [Saab90] present a hierarchical multi-level fault simulation methodology. Fault injection is done at switch level while the simulation is carried out at RTL.

Serial simulation of structural faults in gate/RT level models with event-based simulators is discussed in [Sant99, Nava04]. Mixed-level fault simulation of gate/RT level using concurrent simulation is presented in [Gai88, Lent00]. In [Lent99], the authors propose a multi-level concurrent fault simulation approach allowing multi-level simulation of gate, RT and behavioral level models.

Some works have employed multi-level simulation approach to predict NFPs. In [Shim99], the authors propose a multi-level power modeling at architecture and switch level. “Wattch”

[Broo00] is an architectural level power estimator which uses RTL to obtain cycle count required for power analysis. In [Shan02], a two-level modeling approach is proposed to predict component's surface temperature. The system level model is meant to incorporate all significant flow obstruction mechanisms and heat source distribution on the boards. Since the exact component temperature itself is not the quantity of interest at this level, component and heat sink details may be simplified. The behavioral switch level model incorporates the details of the heat sinks and models each of the components separately.

The authors in [Suta12] use multi-level simulation to estimate system reliability. The simulation method starts from a compact reliability model at switch level. The degradation of device parameters such as transistors' threshold voltage is propagated to gate and system level models for reliability analysis. In the area of aging mechanisms, the authors in [Sidd10] propose a microarchitecture/gate level approach to reduce the impact of NBTI on functional units of processors.

The goal of this work is not to compete with the aforementioned approaches, but to take advantage of multi-level simulation as a practical and beneficial tool for NFP prediction. Chapter 5 explains how multi-level simulation is used to predict the NFP observables.

4.4 Parallel Logic Simulation

Multi-level simulation speeds up the simulation by moving most parts of the system to high level. However, the low level simulation of the DUA can still be a bottleneck. Parallelism, i.e., discrete event-driven simulation (DES), is an efficient way to accelerate the traditional logic simulation.

DES concerns the modeling of a system as it evolves over time by a representation in which the state variables change instantaneously at separate (countable) points in time. These points are the ones at which an event occurs [Law07].

DES models share three common components [Fuji99a]:

- *System state*: The collection of state variables necessary to describe the system at a particular time;
- *Simulation clock*: A variable giving the current value of simulated time;
- *Event list*: A list containing the next time in which the next event will occur.

This implies that DES is sequential in nature. Each event contains a timestamp, and usually denotes changes in the state of the system being simulated. If an event e is the next to occur, the simulation clock is advanced to time t_e in which e will occur. After simulation of the event, the system state is updated to the new state and the occurrence time of the future events are generated. This information is added to the event list. The simulator repeatedly removes the event with the smallest timestamp from the event list, and processes it to obtain the new system state. Choosing the smallest time-stamped event is crucial, because if an event with larger timestamp is selected, it would be possible that it modifies state variables used by the event with smaller timestamp. This is a challenge parallel DES algorithms must overcome. [Fuji99a]

Parallel distributed event-driven simulation (PDES) and time-parallel simulation (TPS) are the most well known techniques to parallelize DES [Fuji99b].

PDES [Nico94, Fuji99a, Cham95, Bagr95] refers to the execution of a single DES program on a parallel computer. These mechanisms usually assume that the simulation consists of a collection of logical processes (LPs) communicating by exchanging the time-stamped messages or events. Each LP can be viewed as a sequential DES with some local state and a list of time-stamped events that are scheduled for it. The LP's event list must also include events received from other LPs. The events are simulated sequentially at each LP while LPs run in parallel. Each LP maintains a simulation time clock that indicates the timestamp of the most recent event processed by the LP.

Much of the work concerning PDES on multiprocessor computers deals with synchronization of LPs. The goal of the synchronization mechanism is to ensure that each LP processes the events in the timestamp order. For this purpose, conservative [Chan81, Misr86, Baue98, Nico96] and optimistic [Jeff85, Gafn88, Zhu05] synchronization mechanisms are proposed in literature [Xueh09, Fuji99b, Cham95]. A complete discussion on PDES and synchronization algorithms can be found in [Fuji99a].

TPS methods partition the entire simulation into a set of shorter simulation runs (slices) in the temporal domain [Fuji99b]. The challenge here is the state matching at the boundaries of the time intervals. Specifically, it is clear that the state computed at the end of the interval $[T_{i-1}, T_i]$ must match the state at the beginning of the interval $[T_i, T_{i+1}]$. This approach relies on the ability to perform the simulation corresponding to the i^{th} interval without first completing the simulations of the preceding intervals.

One approach to solve the state matching problem is to have each processor guess the initial state of its simulation and simulate the system based on the guessed initial state [Lin91, Heid90]. When interval simulations are completed, a “fix-up” computation is performed to account for the fact that the wrong initial state was used. This might be performed, for instance, by simply repeating the simulation, using the final state computed in the previous interval as the new initial state. The fix-up process is repeated until the initial state of each interval matches the final state of the previous interval. In the worst case, N such iterations are required when there are N simulators. However, if the final state of each interval simulator is seldom dependent on the initial state, far fewer iterations will be needed.

The proposed method in [Kies04] allows state deviation at the interval boundary according to the predefined rules. Therefore, the cost of achieving state consistency between time intervals is reduced. The obvious drawback of this approach is the error introduced by the approximate state matching. It must be considered that this error might even invalidate simulation results. However, for many simulation models (e.g., queuing networks), small errors in state changes occurring infrequently do not significantly influence the simulation results.

To overcome the problem of inaccuracy, Kim et al. [Kim11] propose an accurate state prediction approach. The technique consists of two major steps:

- Fast reference simulation, performed at high level model (higher than gate level) of the design to store essential state information at selected checkpoints. This simulation is done on single processor.
- Detailed, fine-grain target simulation, performed at low level (gate level) model. It is applied in parallel to each simulation slice, distributed among individual simulators.

In this approach, as shown in Fig. 4.3, the initial design state for each slice of the target simulation must be first captured and saved during the fast reference run. This is accomplished at predetermined checkpoints, decided by the number of processors available for parallel simulation. The design state consists of the state of all integral registers and memory print of the design. By restoring the design states, each slice can be made independent of the other. As a result, target simulation runs concurrently and independently for each slice.

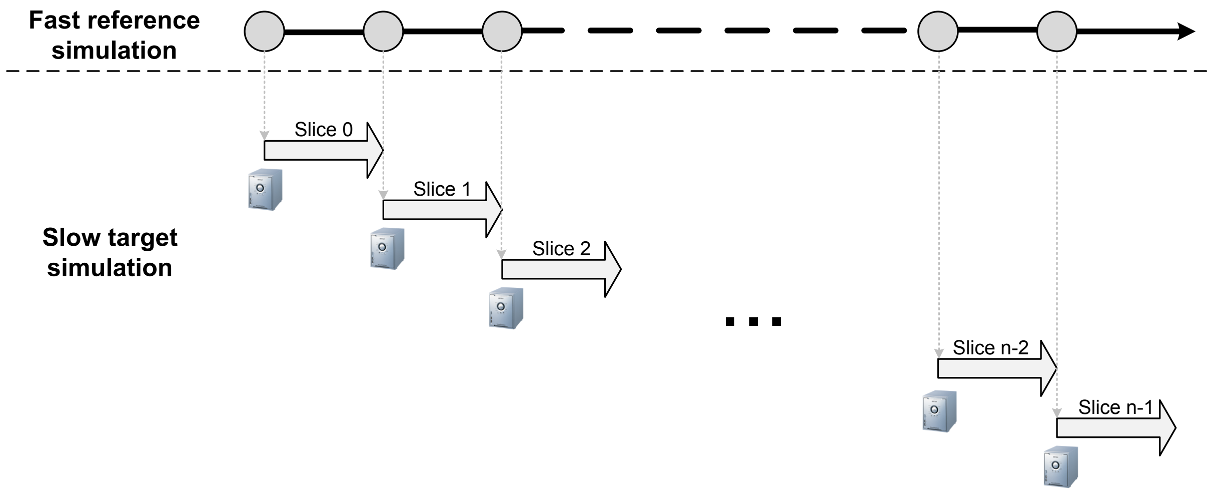


FIGURE 4.3: Concept of TPS approach proposed in [Kim11]

The performance of this method measured in total simulation time is estimated as [Kim11]:

$$T = \Sigma_{i=1}^{n} T_{Ss}(i) + T_{Rsim} + \max[T_{Tsim}(i) + T_{Sr}(i)], \quad 1 \leq i \leq n \tag{4.1}$$

where $T_{Ss}(i)$ is the state saving time for slice i. T_{Rsim} is the conventional simulation time for the reference model. T_{Tsim} is the conventional simulation time and T_{Sr} is the state restoring time for one simulation slice.

Since the overhead of the state saving, $T_{Ss}(i)$, and restoring, $T_{Sr}(i)$, is considerably small, the efficiency of the method depends on T_{Rsim}. For gate level simulation, an RTL model reduces T_{Rsim} to a great extent and is the natural candidate for the reference simulation. RTL simulation is more than 100 times faster than corresponding gate level timing simulation, yet it is still cycle-accurate and the state correspondence between the two abstraction levels is applicable [Kim11].

Chapter 5 shows how the presented TPS method is used as the basis for our parallel, NFP-aware simulation approach.

4.5 Summary

This chapter built the required background on hardware simulation and optimization. Hardware design starts from higher abstraction levels. During the design process, the hardware is refined through different levels. High abstraction levels are faster to simulate, but include less detail. Low level models are accurate, as they include several details about the design, but slower. Section 4.3 introduces multi-level simulation as a powerful method to verify hardware's functional and non-functional properties. Multi-level simulation methodologies move through more than one abstraction level to provide a trade-off between simulation accuracy and speed. The low level simulation of the design under analysis can be accelerated using parallelism, as explained in section 4.4.

Part II

NFP Prediction Methodologies

Chapter 5

Window-Based NFP-Aware Simulation

5.1 Introduction

Several non-functional parameters have a strong dependency to system structure and workload. Operational parameters (called *observables* in this text) are workload dependent parameters that depend on functional properties of the design. The workload dependency can be defined at different abstraction levels concerning the NFP under analysis. However, most observables can be best evaluated at lower levels where system structure is available. As the technology scales, the amplitude of NFP fluctuations under various workloads (e.g., worst vs. average case) grows [Srin04a], which necessitates either a pessimistic worst-case analysis, or an extensive simulation of the target application. For instance, existing models for power consumption and heat distribution rely on the proportional relation of transistor switching activity [Ghos92, Huan04], models for aging mechanisms such as NBTI and HCI are governed by switch level workloads [Wang07e, Wang07c], and for vulnerability models, the gate level input patterns are required [Mukh05].

Observables are time-dependent functions of the system workload. In general, they can be formally defined as:

$$\alpha_i^{(t_i)} = g(workload)$$

where α is an observable under analysis and t_i is the time[1].

[1] $f^{(t_i)}(...) := f(...,t_i)$ where t_i denotes the time.

Observables play an important role in NFP prediction. Different approaches in literature propose methods to predict observables. These approaches are mainly based on average applications [Wang10, Wang07c], probabilistic methods [Lore12], specification of worst-case values or logic simulation [Lore09].

An average application is reflected by modeling the system workload as a set of typical workload patterns. Probabilistic approaches consider signal probability of the inputs and propagate them through the circuit to obtain observables. Worst-case based analysis considers the most pessimistic conditions for observable prediction. Non of these methods provide the required accuracy for NFP analysis. Furthermore, they are not applicable for all the NFPs, specially the ones with high input pattern dependence, e.g., power [Najm94]. Inaccuracy in observable prediction may cause performance degradation and unpredicted failure during the lifetime of the system.

Simulation-based approaches are often used to predict various NFPs, most common of which are power consumption [Kang86, Yaco89, Huiz90] and temperature distribution on the chip [Liao03, Szek97, Dige97]. The advantages of these approaches are mainly accuracy and generality. They can be used to estimate a particular NFP of any circuit, regardless of technology, design style, functionality, architecture, etc. However, complete and specific information about the system structure and input signals is required. Therefore, it can not be applied at early design phases. Moreover, operational parameters change not only during active simulation time, when the core is busy processing transaction requests, but also when the core is autonomous, or idle. During this time, the DUA may perform internal operations, but it does not receive any incoming transaction. A transistor may age also when the system is in standby mode or the leakage power may increase the chip temperature even when the system is not receiving any particular workload. To preserve the accuracy, the DUA should be simulated not only during the active mode, but also during the autonomous periods. Hence, an extensive system-wide simulation at low level is required, which is computationally expensive, if feasible at all.

This chapter proposes a simulation-based methodology to facilitates observable acquisitions early in the deign phase. The presented approach is based on multi-level simulation at transaction/gate level to deal with the intensive simulation of complex SoCs. It is able to estimate different observables required by various NFP models with a single simulation run. The gate level simulation speed is accelerated using parallelism. Simulation during autonomous periods is optimized by the proposed fast-forwarding method (cf. section 5.4.1.3). The method can also be used to evaluate design alternatives regarding NFPs. To this end, high level system models from design space exploration can be reused.

The next section gives an overview of the proposed approach while sections 5.3 and 5.4 discuss the details. Section 5.5 introduces *window-based simulation* for long-term NFP predictions. This technique splits the simulation time into several timesteps, called *window*s, and acquires the observables periodically at the end of each window. Therefore, the temporal changes of observables during the simulation are captured. This feature is desirable for analytical NFP prediction, as it will be discussed in chapter 6. Section 5.6 concludes this chapter with a short summary.

5.2 NFP-Aware Simulation Overview

The NFP-aware simulation approach consists of two main parts, as shown in Fig. 5.1. The *system level workload generation* comprises the complete system modeled at a high abstraction level. The *low level NFP-aware simulation* contains two main components: An *intermediate DUA simulation* with RTL model of the DUA and a *low level DUA simulation* with the gate level description of the DUA. The simulation procedure simulates the gate level DUA with an actual input patterns received from the system level workload generation (section 5.3).

The complete system simulation is performed at transaction level (cf. section 5.3) while the DUA is also available at gate level. The NFP-aware simulation methodology simulates the gate level DUA with the workload received from the transaction level model. Observable acquisition is performed at gate level based on the gate model presented in section 5.4.2. During the low level DUA simulation, the internal nodes of the gate level DUA are observed and changes in observables are monitored. The output of the simulation is the predicted observables.

The simulation at gate level is performed in parallel using state prediction. This is enabled by functional decoupling of gate level simulations by the intermediate level state prediction. The intermediate level should preserve the correspondence of sequential elements between the

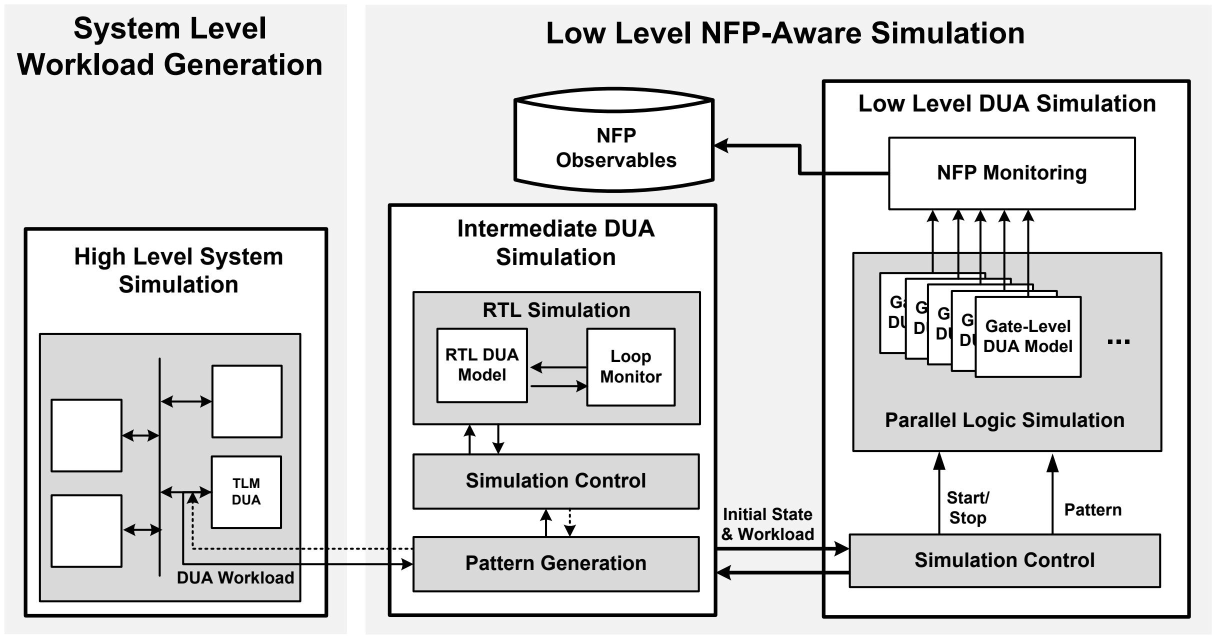


FIGURE 5.1: Multi-level, parallel simulation approach

intermediate and low level models while providing an as fast as possible simulation (else, it would be a bottleneck for the gate level simulation and compensates the speed up gained by parallel gate level simulation). RTL fulfills the aforementioned requirements and is used as an intermediate level. Section 5.4.1 describes the intermediate simulation process in detail.

In addition to state prediction, RTL DUA simulation predicts the length of the stimuli, i.e., the number of cycles required by the simulator to acknowledge the stimuli. This helps the simulation control at gate level to efficiently schedule the received packets. Furthermore, optimizing the simulation of autonomous cycles can significantly speed up the low level simulation as they are a big fraction of the system lifetime. Therefore, autonomous simulation optimization is performed at RTL: If RTL detects a phase of periodic operations, just one period is simulated at gate level and the effect on observables is weighted by the number of skipped iterations.

During RTL simulation, low level simulation jobs start intermittently. Gate level simulations begin with initial states provided by the RTL simulation and cover a limited simulation time span, as in [Kim11]. Fig. 5.2 shows an example of parallel simulation using an intermediate level state prediction. When a new stimuli from intermediate level is received, the initial state in low level simulation is updated and scheduled in a simulation slot. The system state and input stimuli provided by the RTL simulation enables parallel NFP evaluations for consecutive transactions.

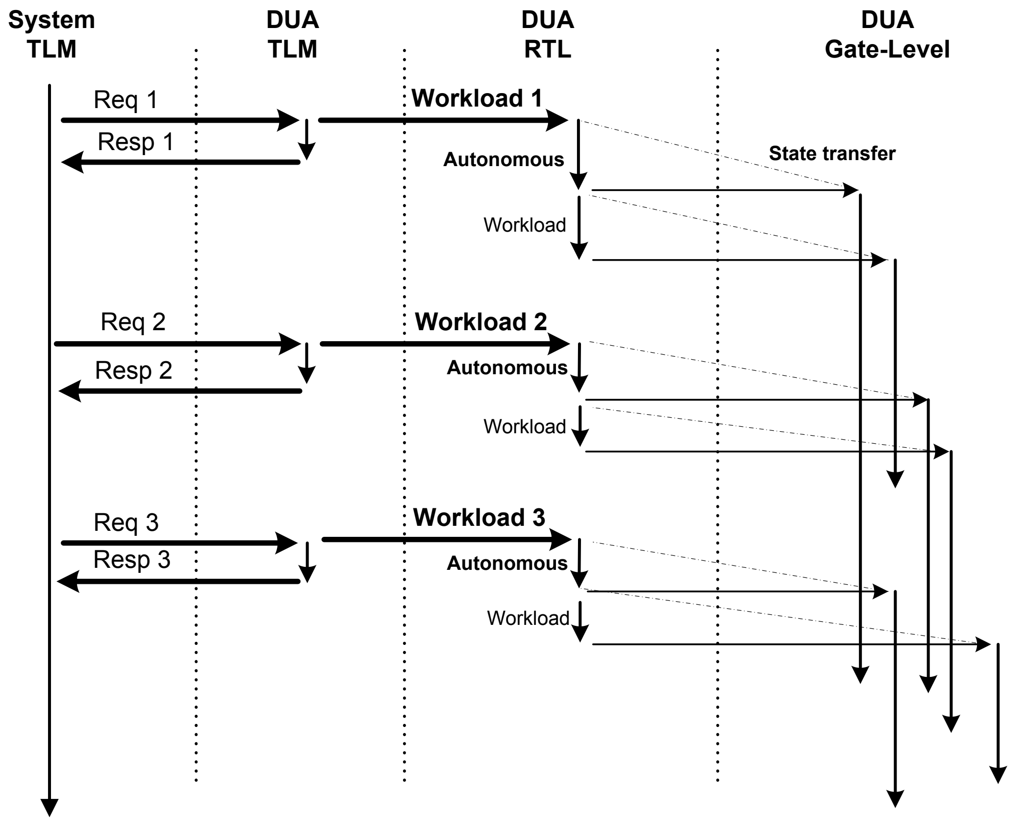


FIGURE 5.2: An example of multi-level parallel simulation

The rest of this chapter discusses different parts of the NFP-aware simulation approach in detail.

5.3 System Level Workload Generation

High level system simulation is used to generate the actual workload for the DUA based on the real application running on the system. To enable high-speed simulation, the temporal and functional behavior of the system's hardware and software modules are modeled at transaction level.

The communication-centric view of TLM provides an abstraction that is well-suited to NFP evaluation. Furthermore, performance modeling in TLM tries to accurately capture the concurrency in a system [Koch09], which can easily be adapted to fit other NFPs as well. For accurate NFP analysis, approximately-timed transaction level modeling is preferred as it provides a fine grain estimation of the system timing [Ghen05].

During high level system simulation, the workload of the transaction level DUA is continuously monitored. Whenever the DUA receives a transaction request, the payload of the transaction and its timestamp are captured as a high level DUA workload. The high level DUA workload is sent to the low level NFP-aware simulation to trigger the low level simulation.

If the subject of the simulation is to study the effect of one or more NFPs on the complete system—such as the effect of soft or intermittent errors in the DUA on the complete system, the transaction model sends the workload to the low level NFP-aware simulation and waits for the feedback. Just after receiving the response from the low level DUA, the system resumes the simulation. If the simulation targets NFP prediction, such as performance degradation due to aging effects, the system level simulation is carried out independent of the low level NFP-aware simulation. Thereby, the simulation speed of the low level model would not block the high level simulation procedure.

5.4 Low Level NFP-Aware Simulation

To start the parallel distributed event-driven simulation at gate level, the initial DUA state upon receiving the transaction as well as the waveforms at primary inputs are required [Fuji90]. The former is obtained by the intermediate level simulation while the latter is generated by the system level workload generation. The low level NFP-aware simulation includes the *intermediate DUA simulation* to predict hardware state and the *gate level DUA simulation* for observable acquisition. The Hardware *state* at time t is defined as the content of hardware's constituent registers at this time.

5.4.1 Intermediate DUA Simulation

The intermediate DUA simulation consists of three main parts:

- The *pattern generation* maps the high level workload to the low level stimuli.
- The *simulation control* provides the required control signals to the RTL simulation.
- The *RTL simulation* performs the RTL DUA simulation and collects the required data for the low level simulation.

In the following, the intermediate DUA simulation components are explained in detail.

5.4.1.1 Pattern Generation

The pattern generation translates high level workloads to pin- and cycle-accurate input patterns. Assuming the received workload as a two-tuple consisting in the transaction observed on the communication interface of the high level DUA and the time t at which the transaction is issued, i.e., the timestamp, the pattern generation performs the following conversion:

$$\mathrm{pg}(t) : (T_t, t) \rightarrow (e_t, t)$$

where T_t is the issued transaction at time t and e_t is the generated event list for the intermediate/low level simulation. *pg* is the conversion function.

The pattern generation has an extensive knowledge of TLM packet structure. We take advantage of TLM 2.0[2] standard [Open08] and use TLM_generic_payload [Ayns08] as a standard TLM transaction. A typical transaction in TLM 2.0 is usually expressed resorting to the following parameters [Ayns08]:

[2]TLM-2.0 consists of a set of core interfaces, initiator and target sockets for sending and receiving transactions, i.e., the generic payload instances. The TLM-2.0 core interfaces consist of the blocking and non-blocking transport interfaces, the direct memory interface (DMI), and the debug transport interface. The TLM_generic_payload is the standard packet format for TLM 2.0 which supports the abstract modeling of memory-mapped buses, together with an extension mechanism to support the modeling of specific bus protocols whilst maximizing interoperability [Ayns08].

- **m_command**: Denotes the type of the transaction. Three values are supported:
  - TLM_WRITE_COMMAND
  - TLM_READ_COMMAND
  - TLM_IGNORE_COMMAND
- **m_address**: Identifies the transaction's base address (byte-addressing).
- **m_data**: When *m_command* = *TLM_WRITE_COMMAND*, it contains a pointer to the data to be written in the target, also when *m_command* = *TLM_READ_COMMAND*, it contains a pointer where to copy the data read from the target.
- **m_length**: Shows the total number of bytes of the transaction.
- **m_response_status**: This attribute indicates whether an error has occurred during the transaction. The supported values are:
  - TLM_OK_RESP
  - TLM_INCOMPLETE_RESP
  - TLM_GENERIC_ERROR_RESP
  - TLM_ADDRESS_ERROR_RESP
  - TLM_COMMAND_ERROR_RESP
  - TLM_BURST_ERROR_RESP
  - TLM_BYTE_ENABLE_ERROR_RESP
- **m_byte_enable**: It can be used to create burst transfers where the address increment between each beat is greater than the word length of each beat. It can also be used to place words in selected byte lanes of a bus.
- **m_byte_enable_length**: For a read or a write command, the byte enable length attribute interprets the number of elements in the bytes enable array.
- **m_streaming_width**: This attribute is used in the streaming mode and specifies the width of the streamed data.

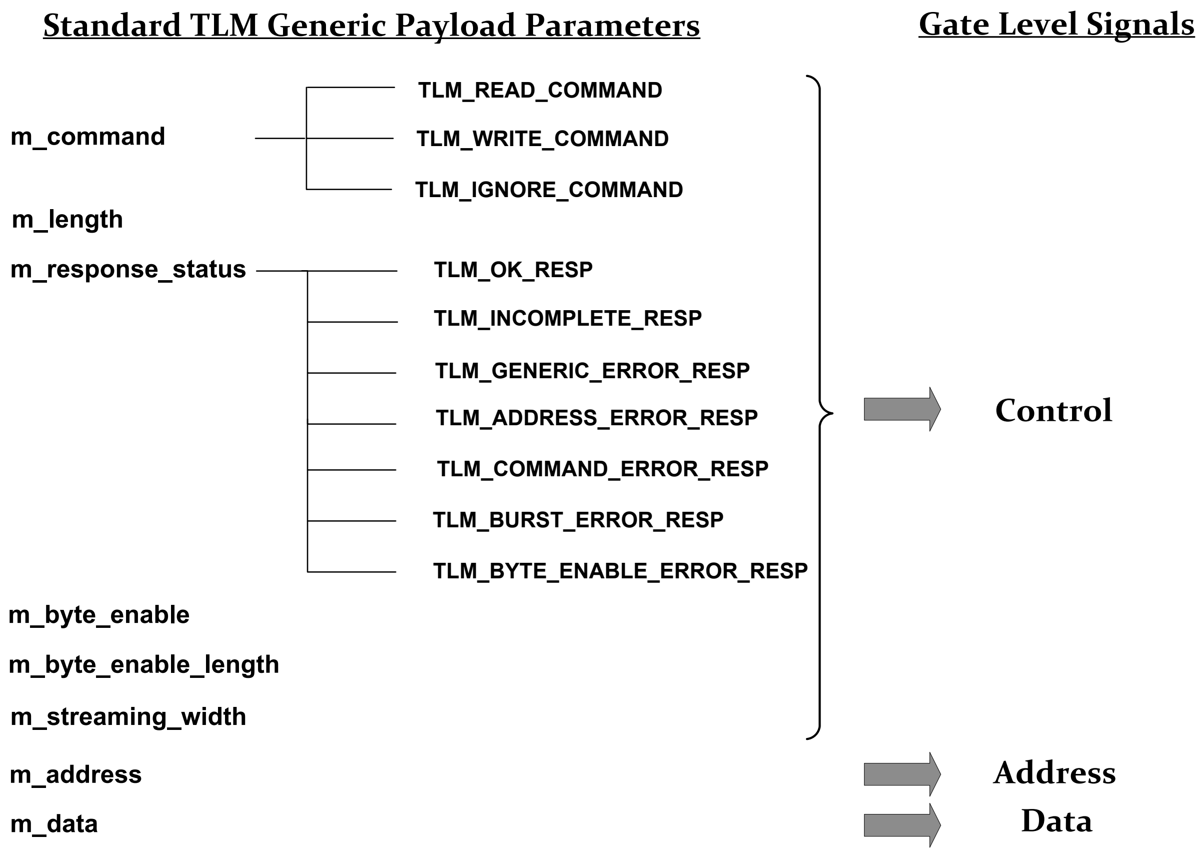


FIGURE 5.3: TLM to low (cycle-accurate) level signal mapping

At low level, the pattern generation classifies the signals to three main classes:

- control signals
- data signals
- address signals

TLM payload parameters are translated to the relevant low level signal classes, as shown in Fig. 5.3. At this point, each field would be mapped to relevant bits at DUA primary inputs. The low level input stimuli is generated automatically based on this mapping.

5.4.1.2 Simulation Control

The simulation control receives the low level stimuli from the pattern generation and initiates RTL simulation with two types of jobs:

- Autonomous cycles simulation (AC-SIM): The simulation control calculates the number of autonomous cycles passed since the last transaction has finished processing. This number is sent to the RTL simulation within an AC-SIM request. Upon receiving the request, the RTL simulation is run for the number of received cycles. The primary inputs are kept constant.
- Input stimuli simulation (IS-SIM): After simulating the autonomous cycles, the simulation control initiates the transaction simulation at the RTL by requesting the IS-SIM job. An IS-SIM request includes the input stimuli generated by the pattern generation.

Another function of the simulation control is to deal with the window-based simulation procedure, as will later be explained in section 5.5.

5.4.1.3 RTL Simulation

RTL Simulation includes the RTL DUA model and a loop monitoring module. As described before, AC-SIM and IS-SIM requests are two types of jobs received from the simulation control. If the requested job is AC-SIM, the RTL simulation initiates the loop monitoring module and starts the simulation. The loop monitoring procedure observes the internal states during the simulation, as will be described in detail in the following. If the requested job is IS-SIM, the RTL DUA simulation captures the actual DUA state and simulates the transaction. After the RTL simulation of the transaction is complete, the captured DUA state and the length of the transaction are sent to the simulation control.

Accelerated Autonomous Simulation The sequential state of the DUA during autonomous simulation is either stable or follows a loop pattern, as shown in Fig. 5.4. During AC-SIM jobs, if a sequential loop is detected, the simulation is skipped (fast-forwarded). The RTL simulation procedure with loop fast-forwarding is shown in Fig. 5.5. Upon reception of an input stimuli at time t_0, the simulation advances as follows:

Let t_{RTL} be the actual simulation time of the RTL model, and let S be a sequence of states, which is initialized with the current DUA state at t_{RTL}. The DUA state is determined by the content of all registers within the RTL DUA model.

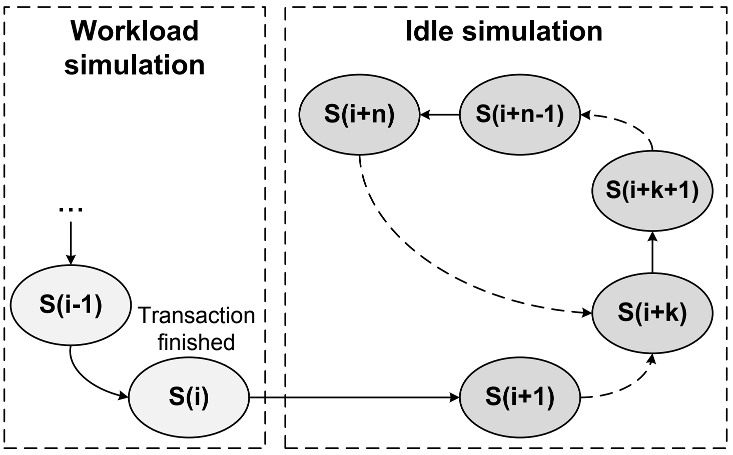


FIGURE 5.4: Autonomous sequential loop schema

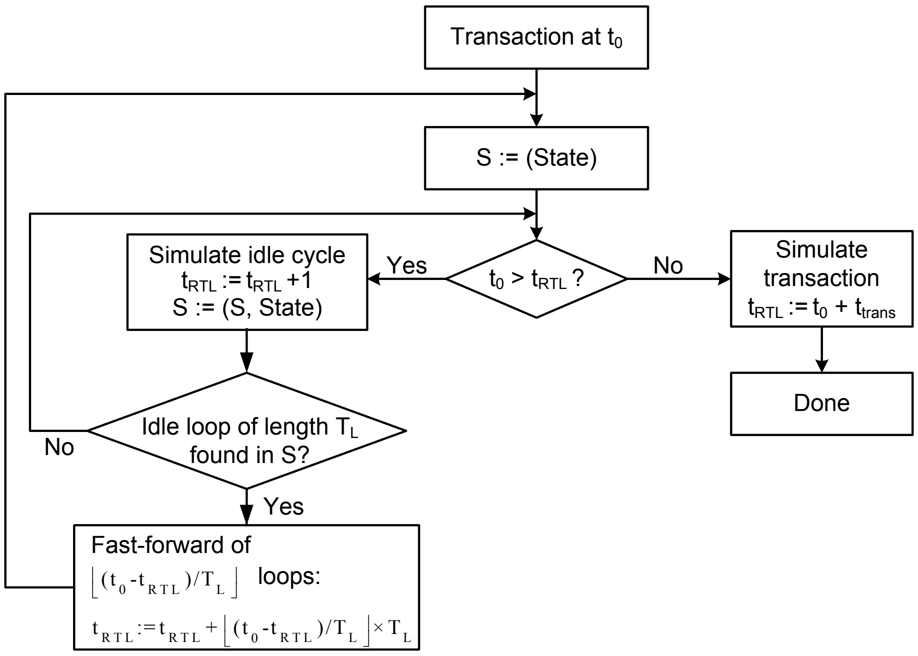

Transaction at t_0
S := (State)
$t_0 > t_{RTL}$?
Yes
No
Simulate idle cycle $t_{RTL} := t_{RTL} + 1$ S := (S, State)
Simulate transaction $t_{RTL} := t_0 + t_{trans}$
Done
Idle loop of length T_L found in S?
No
Yes
Fast-forward of $\lfloor (t_0 - t_{RTL})/T_L \rfloor$ loops:
$t_{RTL} := t_{RTL} + \lfloor (t_0 - t_{RTL})/T_L \rfloor \times T_L$

FIGURE 5.5: Simulation fast-forwarding procedure at RTL

The autonomous period of length $t_0 - t_{RTL}$ is simulated first, with DUA primary inputs kept constant. In each simulation cycle, the current DUA state is checked against S. If a match is found, an autonomous loop is detected. If there is no match, the current DUA state is appended to S and the next autonomous cycle is simulated.

If an autonomous loop of length T_L has been found, the simulation does not proceed cycle-accurately. Instead, the simulation time is fast-forwarded up to time $[t_{RTL} + \lfloor (t_0 - t_{RTL})/T_L) \rfloor \cdot T_L]$, effectively skipping the simulation of $\lfloor (t_0 - t_{RTL})/T_L) \rfloor$ identical autonomous loop iterations.

After the RTL simulation time has reached the current transaction time $(t_0 = t_{RTL})$, the input stimuli is simulated. The RTL simulation is stopped as soon as the input stimuli is acknowledged, and it remains on hold until the next request comes.

The proposed procedure requires that in each autonomous simulation cycle, the DUA state is captured and compared to all the states that have previously been stored. The autonomous loop detection is conducted in parallel to the RTL simulation.

Algorithm 5.4.1: LOOPDETECT$(limit, initialState, totalCycles)$

```
comment: detects the loop within the limit $limit$
$l := 0$
$S := \{\}$
state:= $initialState$
$remainingCycles := totalCycles$
while $(remainingCycles \geq 0)$
  while $(l \leq limit)$
    $simulate(state)$;
    for each $s_i \in S$
      do
        if $(state == s_i)$
          then
            $isloop \leftarrow true$;
            $exit$;
    $S := S \cup state$
    $l++$;
  remove the first element of $S$
  $remainingCycles--$;
```

Both the computational effort and the required storage space are confined by setting a limit for the autonomous loop depth. Algorithm 5.4.1 describes how loop detection within a specified limit is performed. The algorithm receives the limit, the initial state of the DUA and the total number of autonomous cycles as input. At each cycle, the current state is compared with the previous states to find a match. If no match is found, the current state is added to the end of the state sequence. If the limit is reached and no loop is detected, the first state in the state sequence is removed and the algorithm starts again until all the autonomous cycles are simulated or a loop is detected. The variable *RemainingCycles* keeps track of the cycles to be simulated.

As a loop is detected, the information corresponding the loop, i.e., the depth and the length of the loop, are captured. This information is used for simulating the autonomous cycles at gate level and performing loop fast-forwarding without the time demanding task of loop detection.

5.4.2 Low Level DUA Simulation

The low level DUA simulation is the final step to predict NFP observables. A simulation control schedules the received jobs for parallel logic simulation. To perform the NFP-aware gate level simulation, low level gates should be characterized with proper parameters to address the desirable observables. The gate characterization is performed by defining a *gate model* for every gate inside the low level NFP-aware simulator. Finally, the NFP monitoring observes all the DUA nodes and tracks the changes in the observables during the simulation. In the following, each component of the low level DUA simulation is discussed in detail.

5.4.2.1 Simulation Control

The low level simulation control receives the same two types of jobs from the intermediate DUA simulation:

- Autonomous cycles simulation (AC-SIM): An AC-SIM request at this level includes
  - the actual simulation time,
  - the initial state,
  - number of autonomous cycles,
  - depth of the autonomous loop (if any),
  - length of the autonomous loop (if any).
- Input stimuli simulation (IS-SIM): An IS-SIM request contains
  - the actual simulation time (the arrival time of the transaction),
  - the initial state,
  - number of the cycles to be simulated.

Based on the type of the received request, the low level simulation control issues the required signals for parallel logic simulation and schedules the stimuli on the proper simulation slot. If no free slot is found, the intermediate DUA simulation would be informed. At this point, the RTL can start another server to simulate the remaining jobs.

5.4.2.2 Parallel Logic Simulation

Parallel logic simulation consists of several instances of the gate level DUA model. A gate level DUA model is a post-synthesis gate level netlist, consisting of the primitive gates and the registers. For the sake of brevity, we assume that the gate level model is cycle-accurate. If models with post-synthesis timing annotation are available, the proposed approach can be extended with accurate gate level timing simulation.

An instance of the gate level simulator is an n-slot pattern-parallel logic simulator [Kwon99] which is extended for NFP-aware simulation. When a new transaction is received, one of the free pattern-slots is initialized with the initial state obtained from the RTL model and the simulation is run (Fig. 5.6).

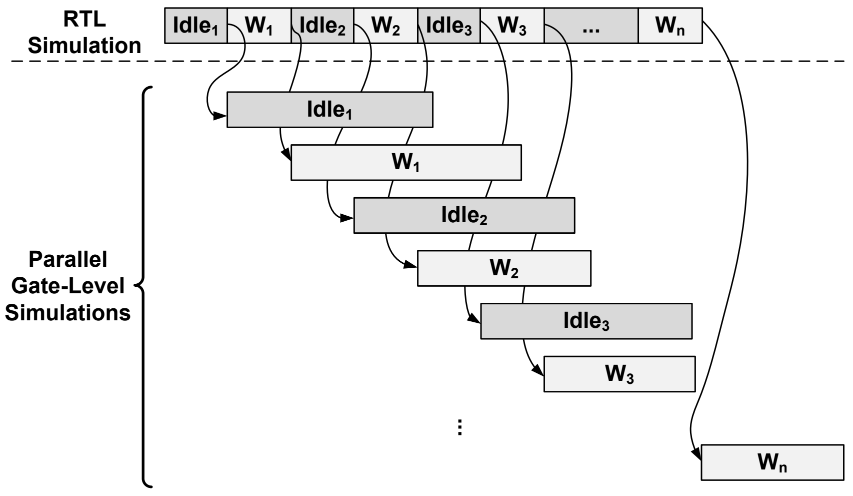


FIGURE 5.6: Parallel simulation procedure

If no pattern-slot is free, no more job requests would be accepted and the intermediate DUA simulation would be informed to start a new gate level simulation server on the same or different machine. This concept is shown in Fig. 5.7. This way, the performance of the parallel simulator depends on the number of the available machines.

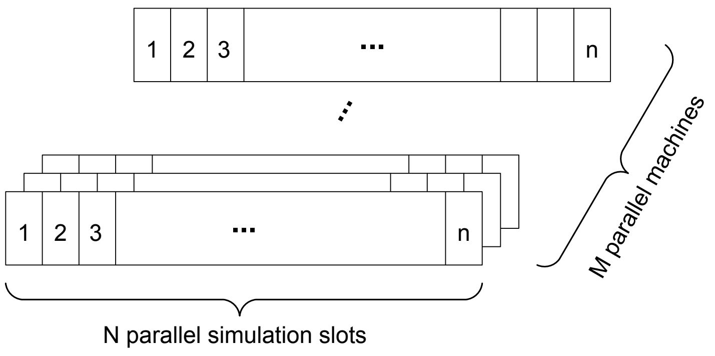


FIGURE 5.7: Parallel simulation architecture

To improve simulation performance, only a subset of observables that are relevant for NFP analysis is acquired during functional simulation. For some NFP models, it is possible to find a set of representative observables or NFPs that can be used for NFP prediction: For instance, to predict the delay degradation of the critical path due to NBTI and HCI aging, it may be possible to evaluate the delay degradation of only a subset of representative gates selected with the approach proposed in [Wang07c, Chen12].

5.4.2.3 NFP Monitoring

Relying on the NFP under analysis, observables are defined and modeled at different abstraction levels. To maintain the consistency between gate level simulation and observables at various levels, a gate model is proposed. The gate model uses the information about the internal gate structure to model the observables within the gate. For instance, Fig. 5.8 shows an inverter model and the relevant equations to estimate the NBTI stress factor. $GATE(T, c_i)$ is the input value of the transistor T at the i^{th} cycle. $SF(INV, 0)$ is the initial state. $SF(INV, c_i)$ is the stress factor for the gate INV at the i^{th} cycle while $SF(INV, (c_i, c_j])$ is the stress factor for the gate INV in the time interval $(c_i, c_j]$.

The proposed gate model can include the definition of several observables at different levels. For instance, the inverter gate model in Fig. 5.8 can also include the equations for switching activity required for power estimation:

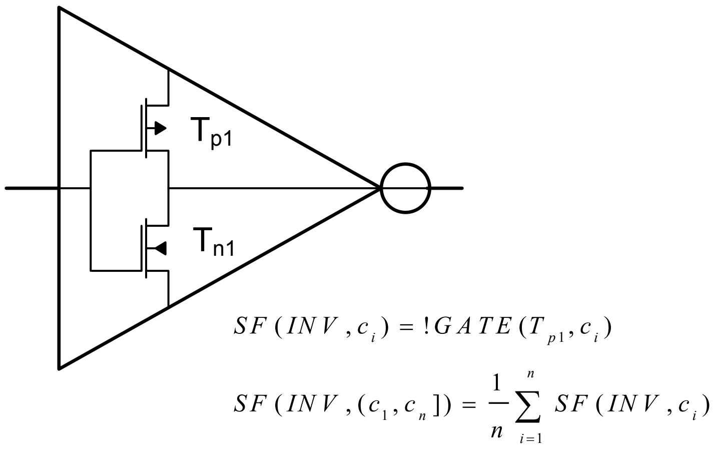

T_{p1}
T_{n1}
$SF(INV, c_i) = !GATE(T_{p1}, c_i)$
$SF(INV, (c_1, c_n]) = \frac{1}{n}\sum_{i=1}^{n} SF(INV, c_i)$

FIGURE 5.8: An inverter model for NBTI analysis

$$\begin{aligned}
SA(INV, 0) &= 0; \\
SA(INV, c_i) &= GATE(T_{p1}, c_i) \oplus GATE(T_{p1}, c_{i-1}) \\
SA(INV, (c_1, c_n]) &= \frac{1}{n}\sum_{i=1}^{n} SA(INV, c_i)
\end{aligned} \tag{5.1}$$

The gate model provides flexibility in evaluations. With a proper gate model, several observables can be estimated with a single simulation run.

5.5 Window-Based Multi-Level Simulation Procedure

As discussed in chapter 2, several NFPs require long-term prediction for the complete system lifetime. In this case, analytical NFP prediction is required. Analytical models include the parameters which affect the NFP behavior during its lifetime, including operational parameters and their dependence to the NFP under analysis. The operational parameters are dynamic, therefore, evaluating them at the end of simulation causes inaccurate NFP prediction, as will be discussed in chapter 6. Window-based simulation is the solution for efficient acquirement of operational parameters for analytical NFP prediction.

In this approach, the simulation time is split into several time frames, called *window*s. NFP observables are collected at the end of each window. The presented NFP evaluation approach, as described in chapter 6, is based on periodic evaluation of NFP models with the acquired observables and can well be synchronized with the window-based simulation. In the following, we show how the proposed multi-level NFP-aware simulation methodology can be used for the window-based simulation.

The window-based simulation proceeds as follows: Anytime a transaction appears on the bus interface of the TLM DUA, the system level workload generator sends it to the intermediate DUA simulation to see if it fits in the current window. The simulation continues as usual if the finishing time of the received transaction does not exceed the window size. Otherwise, the transaction is split. Fig. 5.9 depicts an example of the window-based simulation where horizontal dashed lines represent the window limit at different abstraction levels. As it can be seen in the figure, *Req*4 at transaction level is split at RTL to fit W_2.

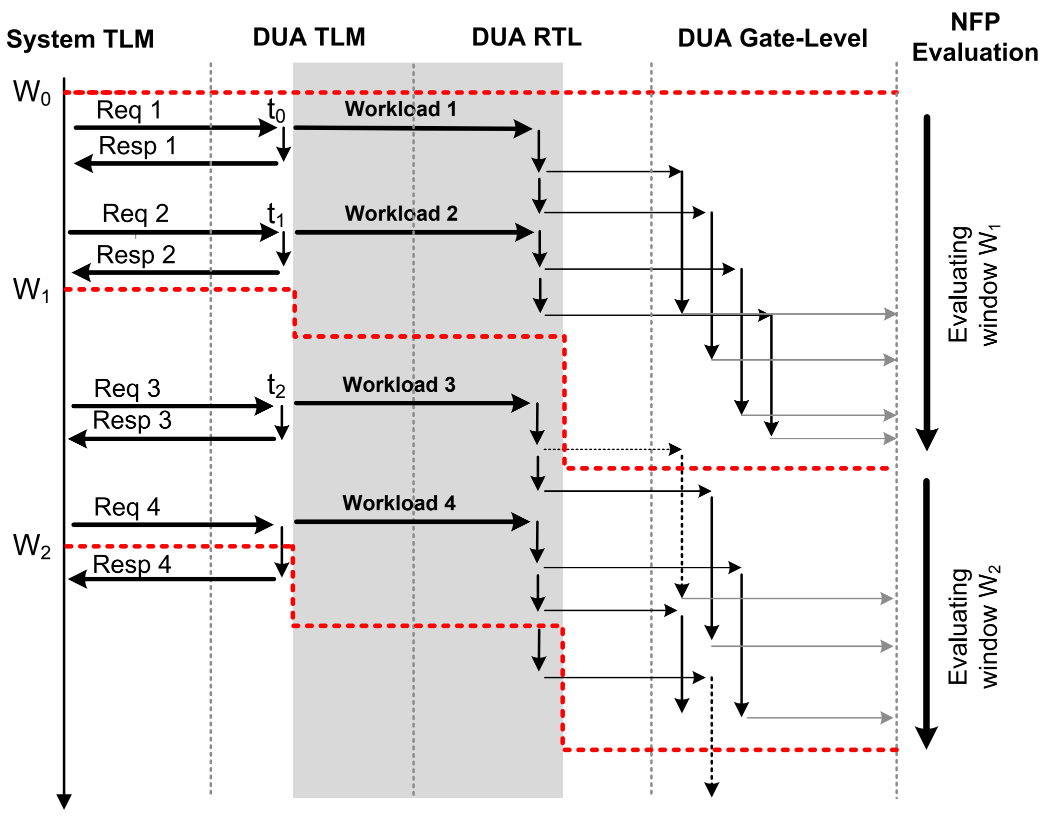


FIGURE 5.9: Window-based simulation example

Fig. 5.10 shows the window-based simulation procedure at RTL. As soon as a new transaction is received at time t_i, the simulation control calculates the number of autonomous cycles passed from the time the previous transaction is completed ($t_{i-1} + L_{T(i-1)}$ where $L_{T(i-1)}$ is the length of the previous transaction). The lighter box in the figure refers to this part. The simulation control checks if the number of autonomous cycles to be simulated exceeds the current window. W denotes the window size. W' is the number of cycles which are already simulated within the window. If the number of autonomous cycles fits in the current window, an AC-SIM job is initiated and sent to the RTL simulation and the variables are updated. Otherwise, the autonomous cycles are split. This procedure is repeated until all the autonomous cycles are simulated. At this point, the transaction is simulated (the darker box in Fig. 5.10). The transaction length (the number of cycles required to acknowledge the transaction) is not known in advance. Therefore, if the transaction exceeds the window at the middle of the simulation, it would be split. The simulation continues until the transaction is acknowledged.

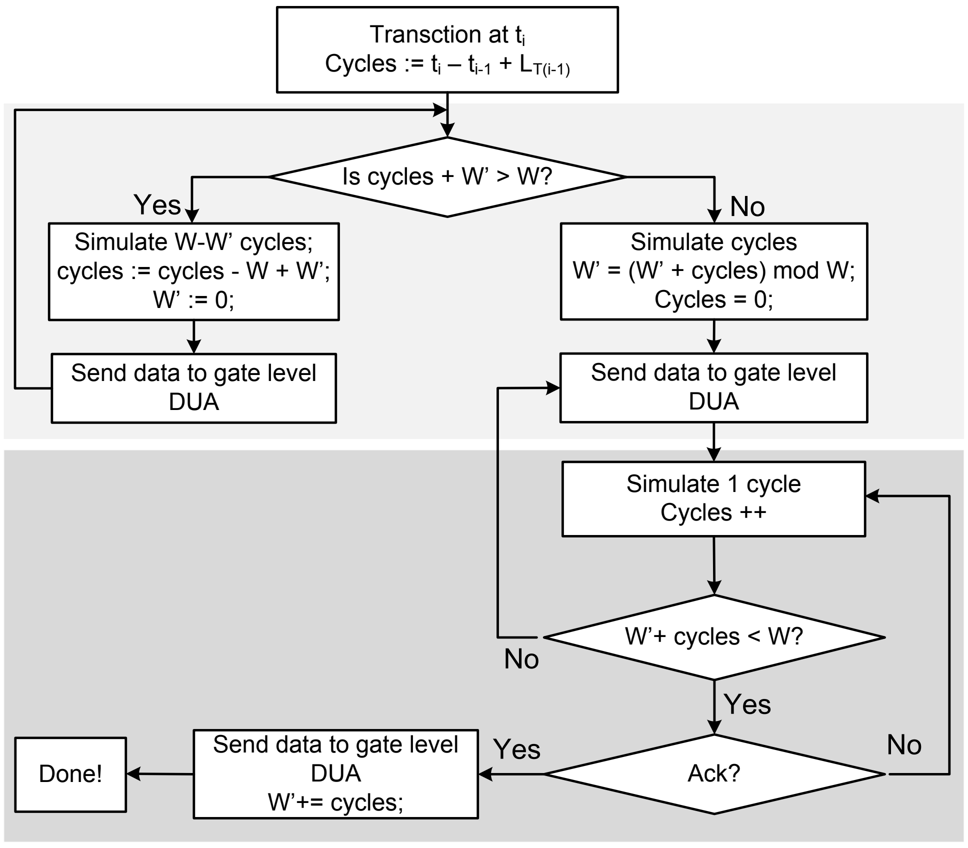

Transction at t_i
Cycles := $t_i - t_{i-1} + L_{T(i-1)}$
Is cycles + W' > W?
Yes
No
Simulate W-W' cycles; cycles := cycles - W + W'; W' := 0;
Simulate cycles W' = (W' + cycles) mod W; Cycles = 0;
Send data to gate level DUA
Send data to gate level DUA
Simulate 1 cycle Cycles ++
W'+ cycles < W?
No
Yes
Ack?
Yes
No
Send data to gate level DUA W'+= cycles;
Done!

FIGURE 5.10: Simulation control at RTL

At the low level DUA simulation, the low level simulation control keeps track of the simulated cycles. If the window limit is reached, the intermediate DUA simulation is signaled and the low level DUA refuses to accept more jobs. As soon as the window is fully covered with the gate level simulations and cumulative observables are available for the full window, the observables acquired from all the simulation jobs are accumulated. At this point, the NFP monitoring module collects the observables for each gate. Just after that, the low level simulation control resumes the simulation.

The size of the window is determined according to accuracy requirements. The non-linear behavior of observables creates a compromise between the size of the window and accuracy of the NFP evaluation. NFP models are partially linearized respecting the observables collected during an evaluation window. The models are fed with accumulated observables up to the end of the current window.

5.6 Summary

This chapter discussed the multi-level NFP-aware simulation methodology to predict NFP observables during lifetime of a system. Observable acquisition is performed at discrete time intervals, called windows. At the end of each window, the acquired observables are sent to the evaluation model and the simulation would reset. To increase the simulation performance, an optimization technique is introduced to fast-forward the simulation in presence of the simulation loops. Beside the proposed technique, the performance and speed can be further improved with parallel simulations at gate level. Parallel simulation requires the state of the system upon receiving any transaction. To fulfill this requirement, an intermediate, i.e., RTL, simulation is used to predict the state of the system as it receives the transaction. The stimuli is sent along with the state of the system to the gate level model to perform parallel NFP-aware simulation.

Chapter 6

Piecewise NFP Evaluation

6.1 Introduction

With the increasing importance of NFPs, it is necessary to consider them in the design process as early as possible [Vieh09]. The earlier the NFPs are analyzed and their effects on system behavior are studied, the more efficient the design process continues and many redesign efforts are avoided. The obstacle here is that non-functional properties are frequently multi-dimensional, still only partially understood in many domains. NFP prediction demands a comprehensive study of the property under analysis in several domains. This study results in a model which includes the behavior of the NFP and the parameters which affect it in those domains.

An accurate NFP modeling requires the current state of the system, all the dependencies and technological and environmental conditions to be included in the model. NFP changes are dynamic and must be considered not once, but continuously during the system operation. The models must be evaluated progressively throughout the system lifetime to achieve high accuracy. The highest accuracy is obtained by cycle-accurate evaluation. However, model inefficiencies may prohibit the accurate or continuous NFP evaluations [Sidd11], e.g., several models do not allow partial evaluation at desired time points.

To overcome model's shortcomings, several works in literature aim at improving NFP models by including more details. This work, instead, proposes a method to evaluate the available models in a more efficient and accurate way.

The majority of NFPs result from relatively slow, long-term physical processes. For example, measurable delay degradation due to aging effects can be observed after hours, days, or months of transistor operation. The changes in chip temperature are not instantaneous, but can be observed after several hundreds of micro or milliseconds upon a change in power consumption or cooling conditions. This implies that a cycle-accurate NFP evaluation is not only computationally expensive, but also unnecessary for many applications.

The piecewise evaluation methodology presented in this chapter splits the operation time of the system into equal timesteps, called *window*s. The accuracy of NFP evaluation depends on the window size. The NFP models are evaluated once per window, considering average system conditions within that window. This stays in contrast to traditional approaches, which evaluate NFPs based on average system conditions [Lore09, Wang10, Wang07c]. An evaluation window may span from several hundreds to several hundred thousands clock cycles, depending on the NFP model response and the required evaluation accuracy. Larger evaluation windows may result in reduced accuracy, but higher evaluation speed. The window size may be subject to dynamic runtime adaptation to adjust the accuracy.

This chapter focuses on evaluating NFP models. Section 6.2 proposes a general formal modeling approach for non-functional properties. The piecewise evaluation technique proposed in section 6.3 partially linearizes NFP models to overcome the model's inefficiency, considering the effect of runtime variations on the NFPs under analysis. Section 6.4 discusses the sources of error and solutions to optimize the window size for a given NFP model. Finally, section 6.5 gives a summary of the method and possible improvements.

6.2 Modeling Non-Functional Properties

Several parameters affect the behavior of an NFP. Chapter 3 discussed models for architectural vulnerability and performance degradation due to different aging mechanisms. The presented models show the variety of parameters which should be included in an NFP model. For accurate NFP modeling, the sources and characteristics of these parameters should be well known and studied. This section provides a general, formal notation for NFP models. The general model facilitates NFP evaluation as comes in the following sections.

Let y be a dynamic, quantifiable NFP under analysis. The general function for evaluating $y^{(t)}$, $y \in \mathbb{R}$ at time t is expressed as

$$\begin{aligned} y^{(t)} = f(&\lambda_1, \lambda_2, \ldots, \lambda_p, \\ &\alpha_1(.), \alpha_2(.), \ldots, \alpha_q(.), \\ &\gamma_1(.), \gamma_2(.), \ldots, \gamma_r(.), t) \\ f, \alpha_i, \gamma_i &: \mathbb{R} \to \mathbb{R}, \quad p, q, r \in \mathbb{N}, \quad \lambda_i \in \mathbb{R} \end{aligned} \tag{6.1}$$

where $t \in \mathbb{R}^+$ is the time. λ_i's $(1 \leq i \leq p,\ i \in \mathbb{N})$ are set of *process parameters*. Process parameters include all the process-dependent constants defined by the technology. Transistor's gate length (l), supply voltage (V_{dd}), oxide thickness (T_{ox}), clock frequency (T), etc. are all examples of λ. λ's can be obtained after the design is mapped to a particular technology. For simplicity, they are noted as Λ from now on in this text.

α_i's $(1 \leq i \leq q,\ i \in \mathbb{N})$, called *observable*s in this text, are workload dependent parameters and depend on the functional properties of the design, as described in chapter 5.

γ_i's $(1 \leq i \leq r,\ i \in \mathbb{N})$ are set of NFPs posed by the operating environment and follow the general form of NFP models as described in equation 6.1. γ_i's can also include the y's previous state.

An accurate NFP evaluation requires all these parameters to be considered in the model. To efficiently evaluate NFP models during the system lifetime, we cluster the models into three classes of recursive, differential, and memoryless forms. In the following, these three classes are discussed in detail.

6.2.1 Differential NFP Models

Several NFPs are modeled by ordinary or partial differential equations. An NFP model in the form of first order ordinary differential equation is usually expressed as

$$\frac{dy}{dt} = f(t) \quad f : \mathbb{R} \to \mathbb{R} \tag{6.2}$$

where t is the time and is involved as an independent variable as in most dynamic NFPs.

A partial differential equation may appear in different orders and dimensions. A second order partial differential equation in two dimensions has the general form of [Anti02]

$$a\frac{\partial^2 f}{\partial x^2} + b\frac{\partial^2 f}{\partial x \partial y} + c\frac{\partial^2 f}{\partial y^2} = 0$$

where a, b and c are constants.

6.2.1.1 Example: Interconnect Thermal Modeling

Heat is generated in the substrate as well as the interconnections. The major source of heat generation is the power dissipation of devices that are embedded in the substrate. Power dissipation increases by Joule heating (or self-heating) caused by the flow of current in the interconnect network. Although interconnect Joule heating constitutes only a small fraction of the total power dissipation in the chip, the temperature rise in the interconnections can be significant. This is due to the fact that interconnects are located away from the silicon substrate and the heat sink by several layers of insulating materials which have lower thermal conductivities than that of silicon.

The one-dimensional heat diffusion equation in metal interconnection under the steady state can be written as [Pedr06]

$$\frac{\partial^2 T_{metal}(x)}{\partial x^2} = \frac{1}{k_m}((\frac{k_{ins}}{t_m t_{ins}} - \frac{\rho_i \beta I_{rms}^2}{w^2 t_m^2})T_{metal}(x) - \frac{k_{ins}}{t_m t_{ins}} T_{chip}(x) - \frac{\rho_i I_{rms}^2}{w^2 t_m^2}) \tag{6.3}$$

This differential equation is the basis of the interconnect temperature calculations.

6.2.2 Recursive NFP Models

A *recurrence (difference)* relation is an equation that recursively defines a sequence. Once one or more initial terms are given, each further term of the sequence is defined as a function of the preceding terms. Recurrence equations are frequently used to refer to any recurrence relation. They usually appear in dynamic systems and involve an integer function $f(n)$ in a form like [Chen03]:

$$f(n) - f(n-1) = g(n)$$

where $g(n)$ is some integer function. The above equation is the discrete analog of the first order ordinary differential equation:

$$f'(x) = g(x)$$

A *recursive* NFP model is an NFP model in the form of a recurrence equation. It relies on the NFP's previous state and have the following form:

$$\begin{aligned}
y^{(t_0)} &= g(t_0, .) \\
y^{(t_n)} &= f(\Lambda, \alpha_1^{(t_n)}(.), \ldots, \alpha_q^{(t_n)}(.), \\
&\qquad \gamma_1^{(t_n)}(.), \gamma_1^{(t_n)}(.), \ldots, \gamma_{r-1}^{(t_n)}(.), y^{(t_{n-1})}, t_n) \\
f, \alpha_i^{(t_k)}, \gamma_i^{(t_k)} &: \mathbb{R} \to \mathbb{R}, \quad q, r \in \mathbb{N}
\end{aligned} \tag{6.4}$$

where t_0 is the initial time. y is the NFP under analysis. $y^{(t_0)}$ is the initial condition for the NFP, i.e., the NFP value at time t_0. $\alpha_i^{(t_n)}$'s are observables at time t_n, i.e., the n^{th} timestep. The NFP value at time t_n is calculated from its previous value evaluated at time t_{n-1} and average system conditions in the time interval $(t_{n-1}, t_n]$. Here, t_{n-1} corresponds simply to the time interval in which we wish to do the evaluation.

6.2.2.1 Example: Dynamic Energy

Dynamic energy for a transistor corresponds to the power dissipated by the transistor during the time interval $[t_1, t_2]$. This power is spent in charging the capacitances associated with the transistors and wires.

Dynamic energy is a function of time (t), supply voltage (V_{DD}), operating frequency (f), load capacitance of the node (C_{load}), and the switching activity (β). It is computed as [West85]

$$E_{dyn}(\Delta t_n) = V_{DD}^2 \cdot f \cdot C_{load} \cdot \beta(\Delta t_n) \cdot \Delta t_n \tag{6.5}$$

where, in general, $\Delta t_k = t_k - t_{k-1}$. Using equation 6.5, the total energy at time t_n can be expressed as

$$E_{dyn}(t_n) = E_{dyn}(t_{n-1}) + V_{DD}^2 \cdot f \cdot C_{load} \cdot \beta(\Delta t_n) \cdot \Delta t_n \tag{6.6}$$

where t_{n-1} is the energy at the previous time point and $\beta(\Delta t_n)$ is the switching activity of the transistor in the time interval $[t_{n-1}, t_n]$.

Here, we have neglected the short-circuit component of the dynamic energy, which is due to the current that flows from the power supply to the ground when the devices are switching and both the pull-up and the pull-down network of the gate are conducting. This component of energy is generally small and can be safely neglected. However, it is important to note that if a design is not sufficiently optimized and has large transition times, then the short-circuit power dissipation and hence the energy can form a significant fraction of the total energy [Sriv05].

6.2.3 Memoryless NFP Models

Memoryless NFP models describe a single interval of NFP evaluation ($[0,t]$). The formula is evaluated with the average system conditions at the end of system operation, i.e., in the time interval $[0,t]$.

Memoryless NFP models cannot capture the changes in system's conditions, therefore, they may not provide enough accuracy for NFP prediction. The proposed piecewise evaluation method is used to express these models recursively to facilitate their accurate evaluation during the system lifetime.

6.2.3.1 Example: Delay Degradation Due to NBTI

The delay degradation of a PMOS transistor due to the NBTI effect (cf. section 3.3.1) in the time interval $(0,t]$ can be approximated with the following equation [Noda10]:

$$\Delta d(t,\alpha) = A\alpha^n t^n d_0 \tag{6.7}$$

where α is the probability that $V_{GS} = -V_{DD}$ (stress factor), d_0 is the initial transistor delay, and A and n are the technology parameters.

6.2.3.2 Example: Failure Rate Modeling

The failure rate determination for a collection of ULSI (Ultra Large Scale Integrated) chips is of primary reliability importance. The failure rate λ is usually given by [Doer07]

$$\lambda(t) = \frac{f(t)}{1-F(t)} \tag{6.8}$$

where $f(t)$ is the failure probability density function. $F(t)$ is the cumulative failure probability. FIT (Failure In Time) is the accepted unit for failure rate. One FIT specifies one failure in a billion hours.

6.3 Piecewise Evaluation of General NFP Models

The main idea behind the piecewise evaluation approach is to evaluate NFP models periodically and at the end of each evaluation window. To reach this goal, the NFP model should allow continuous evaluation at different time steps. Piecewise evaluation provides a method to change an NFP model to its equivalent recursive form. At the end of each window t_i, the equivalent NFP model is fed with proper parameters calculated for that window, namely the average observables and interdependent NFPs in the time interval $(t_i - W, t_i]$, where W is the length of the window. The accuracy of the evaluated NFP corresponds to the accuracy of the model, the accuracy of the provided parameters, and the frequency of evaluation, i.e., the size of the evaluation window. The accuracy of predictions is adapted by adjusting the window size.

In the following, we describe the piecewise evaluation method for typical types of NFP models.

6.3.1 Recursive NFP Models

Recursive NFP models are most suitable for piecewise evaluation, as they predict the NFP based on its previous value. To evaluate a recursive model in a piecewise manner, the simulation time is split into windows of size W. The NFP value at time t_i is calculated from its previous value evaluated at time $t_i - W$ and the average system observables in the time interval $(t_i - W, t_i]$. t_i is equal to $t_0 + kW$ where k is a constant. The recursive NFP can be rewritten as

$$\begin{aligned}
y^{(t_0)} &= g(t_0, .) \\
y^{(t_n)} &= f(\Lambda, \alpha_1^{(t_n)}(.), \ldots, \alpha_q^{(t_n)}(.), \\
&\qquad \gamma_1^{(t_n)}(.), \gamma_2^{(t_n)}(.), \ldots, \gamma_{r-1}^{(t_n)}(.), y^{(t_n - W)}, t_n) \\
&f, \alpha_i, \gamma_i : \mathbb{R} \to \mathbb{R}, \quad q, r \in \mathbb{N}
\end{aligned} \tag{6.9}$$

Fig. 6.1 shows how a recursive NFP model is evaluated using the piecewise evaluation method. At the end of each window at time t_n, the models for the NFP under analysis (y) as well as interdependent NFPs ($\gamma_1, \ldots, \gamma_{r-1}$) are fed with proper process parameters and observables evaluated for the current window. The NFP model is also fed with its previous value at time $t_n - W$, where W is the size of the window.

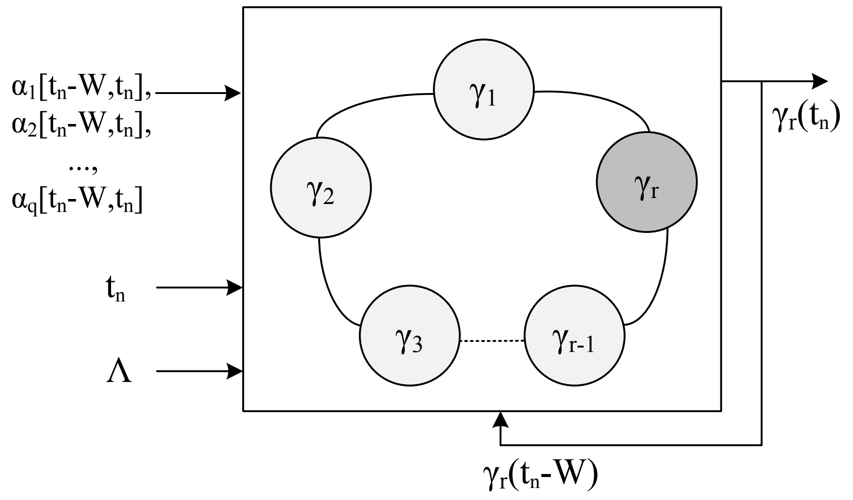


FIGURE 6.1: Piecewise evaluation of recursive NFP models

6.3.2 Differential NFP Models

NFP models in the form of differential equations are evaluated using partial linearization. Several linearization methods can be found in literature (e.g., [Anti02, Poly03]) among which the so-called *numerical integration methods* are the most straightforward numerical solutions. In these methods, the spacing h_j at different steps is assumed to be constant and hence $t_j = t_0 + jh$. Further, the computed value of the solutions at the new point depends only on a fixed number of the previous values. This implies that the piecewise evaluation technique can well be used to evaluate NFP models using numerical integration methods.

As an example, we discuss a simple numerical integration method, called Explicit Euler method [Anti02], for piecewise evaluation of ordinary differential equations. Explicit methods essentially extrapolate the value of the solution at the next point using previous points. There are more elaborate methods which may provide higher accuracy at the cost of increased evaluation time, namely Heun and Runge-Kutta methods. More information about numerical integration methods can be found in [Anti02].

Problems arising in ordinary differential equations can be usually reduced to a system of first-order differential equations [Anti02]. For example, a differential equation of order m

$$\frac{d^m y}{dt^m} = f(t, y, \frac{dy}{dt}, \ldots, \frac{d^{m-1}y}{dt^{m-1}})) \tag{6.10}$$

can be reduced to a system of m first-order equations by defining new variables $y_1 = y$, and $y_{j+1} = \frac{d^j y}{dt^j}$ for $j = 1, \ldots, m-1$, to get

$$\begin{aligned} \frac{dy_j}{dt} &= y_{j+1}, (j = 1, 2, \ldots, m-1), \\ \frac{dy_m}{dt} &= f(t, y_1, y_2, \ldots, y_m). \end{aligned} \tag{6.11}$$

Accordingly, we only consider solutions of a system of the latter form:

$$\frac{dy_j}{dt} = f_j(t, y_1, y_2, \ldots, y_n), \quad (j = 1, 2, \ldots, n)$$

To estimate a first order differential equation, we should first specify the initial values:

$$g_i(y_1(t_0), y_2(t_0), \ldots, y_n(t_0)) = 0 \qquad (j = 1, 2, \ldots, n)$$

where t_0 is the starting point in which the initial values are defined. Explicit Euler method transforms the ordinary differential equation to a recurrence equation. In this method, the NFP model is discretized as follows:

$$\begin{aligned} y^{(t_0)} &= g(t_0, .) \\ y' &= f(y, t) \\ y^{(t_n)} &= y^{(t_n - W)} + W f(y^{(t_n - W)}, t) \end{aligned} \tag{6.12}$$

where W is the window size. Fig. 6.2 shows an example of linearization using Euler method.

Since the explicit Euler method approximates a continuous function at discrete points, it is bound to have some truncation error. If the value of the NFP function at the end of the previous window is known, then the truncation error is simply the error in integration over

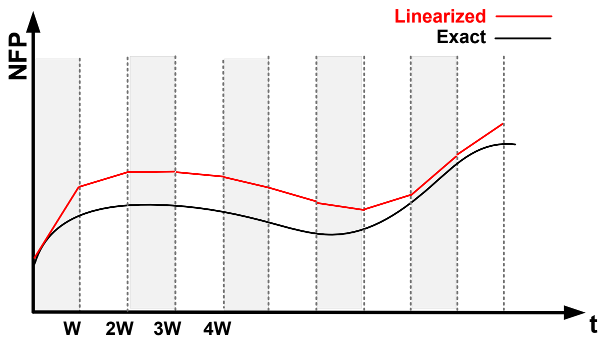


FIGURE 6.2: Linearization example using Euler method

the last step (the local truncation error[1]) and it can be easily estimated. Using Tailor series expansion [Abra64], the local truncation error is expected to be in the form $\frac{1}{2}W^2y"(\xi)$ [Anti02].

6.3.2.1 Example

For the sake of simplicity, we apply the Euler method for piecewise evaluation of a sample model. Assume the first order problem

$$\begin{cases} \frac{dx}{dt} & = -2\,tx + 3\,y^2 \\ \frac{dy_j}{dt} & = -3\,x^2\,(1-y) \end{cases}$$

represent the NFP y with the initial value of $(-1,2)$ at $t_0 = 0$. The value of y at the end of each window with the size of W is calculated as

$$t_i = t_{i-1} + W$$
$$x_i = x_{i-1} + W\,x'(t_{i-1}) = W(-2\,t_{i-1}x_i + 3\,y_{i-1}^2)$$
$$y_i = y_{i-1} + W\,y'(t_{i-1}) = W(-3\,x_{i-1}^2\,(1-y_{i-1}))$$

[1]Local truncation error is defined as the error introduced in the i^{th} step, assuming that the previous values of y_{i-k} are exact.

6.3.3 Memoryless NFP Models

A memoryless NFP model evaluates the NFP under analysis with the average system conditions in a single time interval, i.e., $[0,t]$. As the evaluation is performed once at the end of the system lifetime, iterative capturing of the system conditions during this time is prohibited. To overcome this challenge, the static NFP model is transformed to a piecewise form as follows: To evaluate the NFP y at the end of each window of size W, first the initial values at the time t_0 are specified. Afterward, at the end of each window, the equivalent time, t', is found such that for the observables and the interrelated NFPs at time $t_0 + Wk$, the NFP value is the same as for the previous window, $t_0 + W(k-1), k \in \mathbb{N}^+$:

$$\begin{aligned}
&y^{(t_0)} = g(t_0, .) \\
&y^{(t_n)} = f(\Lambda, \alpha_1^{(t_n)}(.), \alpha_2^{(t_n)}(.), \ldots, \alpha_q^{(t_n)}(.), \\
&\qquad\qquad \gamma_1^{(t_n)}(.), \gamma_2^{(t_n)}(.), \ldots, \gamma_r^{(t_n)}(.), t' + W) \\
&\quad f, \alpha_i, \gamma_i : \mathbb{R} \to \mathbb{R}, \quad q, r \in \mathbb{N}
\end{aligned} \tag{6.13}$$

where t' satisfies

$$\begin{aligned}
&y^{(t_n - W)} = f(\Lambda, \alpha_1^{(t_n)}(.), \alpha_2^{(t_n)}(.), \ldots, \alpha_q^{(t_n)}(.), \\
&\qquad\qquad \gamma_1^{(t_n)}(.), \gamma_2^{(t_n)}(.), \ldots, \gamma_r^{(t_n)}(.), t') \\
&\quad f, \alpha_i, \gamma_i : \mathbb{R} \to \mathbb{R}, \quad q, r \in \mathbb{N}
\end{aligned} \tag{6.14}$$

$y^{(t_n - W)}$ is the NFP value calculated in the previous evaluation window. Solving equation 6.14 for t' and substituting it in equation 6.13 evaluates the NFP under analysis at time t_n.

Fig. 6.3 shows an example of the piecewise evaluation for an arbitrary memoryless model. For the sake of simplicity, we assume that y is affected only by the time-dependent parameter, α. At the time t_0, $y^{(t_0)}$ is initiated with 0. At the time t_1, $\alpha = k_2$ and the memoryless model is evaluated with the actual α. At the time t_2, α changes to k_1. As the model is memoryless, we can not simply evaluate it with the new α. Therefore, the time t'_1 is found, such that $y = y^{(t_1)}$ and $\alpha = k_1$. The value for $y^{(t_2)}$ is obtained by evaluating the model at the time $t'_1 + W$ with $\alpha = k_1$.

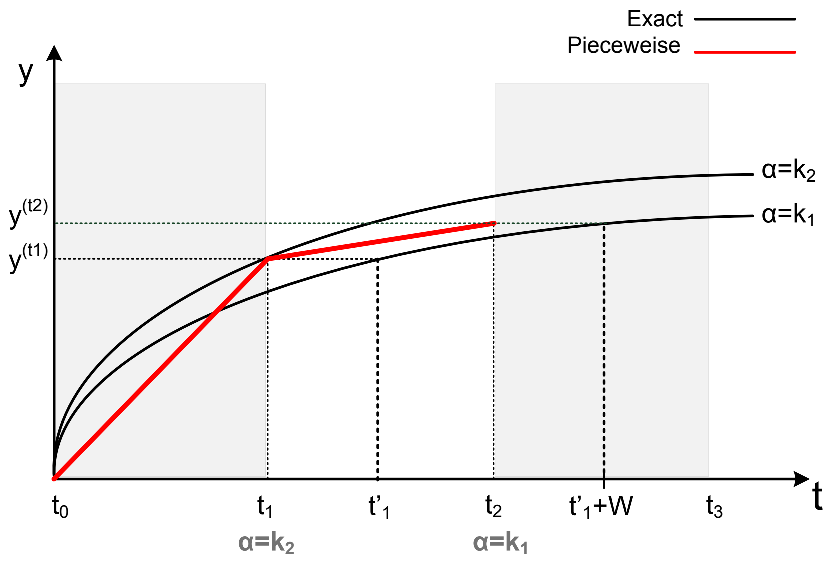


FIGURE 6.3: An example for piecewise evaluation of a memoryless NFP model

6.3.3.1 Example: NBTI Aging

As shown in section 6.2.3, the delay degradation of a PMOS transistor due to the NBTI effect is described by the memoryless equation [Noda10]:

$$\Delta d(t,\alpha) = A\alpha^n t^n d_0 \tag{6.15}$$

where α is the low-level observable, d_0 is the initial delay, and A and n are technology parameters.

The equation is evaluated piecewise at the time t_i as follows:

$$\Delta d(t_i,\alpha_i) = \Delta d(t'_{i-1} + W, \alpha_i) \tag{6.16}$$

where α_i is the stress factor in $[0,t_i]$ and t'_{i-1} satisfies:

$$\Delta d(t'_{i-1}, \alpha_i) = \Delta d(t_i - W, \alpha_{i-1})$$

where $\Delta d(t_i - W, \alpha_{i-1})$ is the delay degradation calculated in the previous evaluation window. According to equation 6.15, t'_{i-1} is calculated as

$$A \cdot \alpha_i^n \cdot (t'_i)^n \cdot T_0 = \Delta d(t_i - W, \alpha_{i-1})$$

Thus:

$$t'_{i-1} = \sqrt[n]{\frac{\Delta d(t_i - W, \alpha_{i-1})}{A \cdot T_0}} \cdot \frac{1}{\alpha_i} \tag{6.17}$$

From equation 6.16, we obtain the recursive function for delay degradation at time t_i:

$$\Delta d(t_i, \alpha_i) = A \cdot \alpha_i^n \cdot T_0 \cdot (\sqrt[n]{\frac{\Delta d(t_i - W)}{A \cdot T_0}} \cdot \frac{1}{\alpha_i} + W)^n \tag{6.18}$$

6.4 Optimal Window Size

The piecewise NFP evaluation methodology predicts the NFP at the end of each window based on the value of the previous window. There are two sources of truncation error using this approach. Assuming that $y^{(t_{i+1})}$ stands for the value of NFP y at $(i+1)^{\text{th}}$ window. First, there is some truncation error introduced in the $(i+1)^{\text{th}}$ window (the local truncation error). Second, the computed value of $y^{(t_i)}$ itself can not be exact, thus introduces an additional error while computing $y^{(t_{i+1})}$. In most cases, it is comparatively easy to estimate this local truncation error. The second error source is essentially the propagation of the local truncation error. This contribution is rather difficult to estimate [Anti02].

Truncation error depends on the size of the window. Most methods don't yield a meaningful solution for extremely large windows (depending on the NFP). However, small windows demand intensive calculations. Knowing the tolerated error for the calculations, the window size should be chosen in a way not to exceed the error bound.

The general error ε for one window with the size W can be calculated as

$$\begin{aligned}\varepsilon = &\sum_{i=1}^{W} f(\Lambda, \alpha_1^{(i)}(.), \ldots, \alpha_q^{(i)}(.), \gamma_1^{(i)}(.), \ldots, \gamma_r^{(i)}(.), i) - \\ &f(\Lambda, \frac{\sum_{i=1}^{W} \alpha_1^{(i)}(.)}{W}, \ldots, \frac{\sum_{i=1}^{W} \alpha_q^{(i)}(.)}{W}, \frac{\sum_{i=1}^{W} \gamma_1^{(i)}(.)}{W}, \ldots, \frac{\sum_{i=1}^{W} \gamma_r^{(i)}(.)}{W}, W) \\ &f, \alpha_i, \gamma_i : \mathbb{R} \rightarrow \mathbb{R}, \quad q, r \in \mathbb{N}\end{aligned} \tag{6.19}$$

As the error function highly depends on the model, there is no known, general way to calculate the error of an arbitrary NFP model. However, some methods from numerical analysis can be reused to find the optimal window size which produces a tolerated error. As an example, we discuss how to obtain an optimal window size in the explicit Euler method described in section 6.3.2. Other techniques can be found in [Atki08].

Using the explicit Euler method, we know that the error function is in the form of

$$\frac{1}{2}h^2 y\text{''}(\xi) + O(h^3)$$

However, the value of $y\text{''}(\xi)$, even approximately, is not known to us. To obtain the correct window size, we consider that one step of Euler method with the window size of h consists of doing two steps of Euler of size $\frac{h}{2}$. Using the window size of h, we calculate the NFP function at point $t_n + h$ [Atki08]:

$$A_1 = y^{(n)} + hf(t_n, y_n)$$

So, A_1 can be written as

$$A_1 = \phi(t_n + h) + \frac{1}{2}h^2 f\text{''}(\xi) + O(h^3)$$

where $\phi(t_n + h)$ is the exact value of NFP function at $t_n + h$. Instead, evaluating the NFP function with the window size of $\frac{h}{2}$ at the points $t_n + \frac{h}{2}$ and $t_n + h$, we have

$$A_2 = y^{(n)} + \frac{h}{2}f(t_n, y^{(n)}) + \frac{h}{2}f(t_n + \frac{h}{2}, y^{(n)} + \frac{h}{2}f(t_n, y^{(n)}))$$

The first step took us from $y^{(n)}$ to $y^{(mid)} = y^{(n)} + \frac{h}{2}f(t_n, y^{(n)})$ and the second step took us from $y^{(mid)}$ to $y^{(mid)} + \frac{h}{2}f(t_n + \frac{h}{2}, y^{(mid)})$. The local truncation error introduced in the first and second half window is $\frac{1}{2}(\frac{h}{2})^2 y\text{''}(\xi) + O(h^3)$ where $O(h^3)$ is different for each half. All together, we can say that

$$A_2 = \phi(t_n + h) + \frac{1}{2}h^2 y\text{''}(\xi) + O(h^3)$$

The difference is

$$A_2 - A_1 = \frac{1}{2}h^2 f\text{''}(\xi) + O(h^3)$$

The error $\frac{1}{2}h^2 f\text{''}(\xi)$ can be estimated using $A_2 - A_1 + O(h^3)$. In other words, the current error rate is about

$$r = \frac{|A_2 - A_1|}{h} \approx \frac{1}{2}hf\text{''}(\xi)$$

per unit increase of t. If $r > \varepsilon$, we reject A_2 and repeat the current step with a new trial window size h' chosen so that $\frac{1}{2}|f\text{''}(\xi)|h' \approx \frac{r}{h}h' < \varepsilon$.

6.5 Summary

This chapter presented the piecewise evaluation method for long-term NFP analysis. The technique overcomes the NFP model inefficiencies by finding the equivalent iterative form of a given NFP model. NFP models are classified to recursive, differential, and memoryless forms. The method evaluates the piecewise equivalent of the given NFP model periodically by dividing the system lifetime to windows and performing the evaluation at the end of each window. The accuracy of evaluations is adjusted by choosing the correct size for the window.

Part III

Applications

Chapter 7

Architectural Vulnerability Analysis

7.1 Introduction

Radiation-induced soft errors have emerged as a key challenge in computer system design. The variability and defect mechanisms in nano-scale CMOS are complex [Roy06] and require the soft error effects on the circuit to be considered also during the functional operation [Bork05]. Only a subset of the errors observed at logic level leads to failures at system level [Leve04], but those that do must be accurately analyzed. The analysis of these interactions at early design stages gives an important feedback for reliable [Leve03, Jhum05] and secure systems [Roth04].

The impact of faults also depends on application scenarios [Watt04, Cano06]. Each scenario occupies and utilizes the hardware components differently. A complete SoC simulation models the effect of errors on a system considering the application. However, it is usually not feasible to run the gate level simulation of a complex design either for its size or model availability. Instead, the multi-level simulation techniques are used. These techniques use models at different abstraction levels: Models with high accuracy for the fault injection and highly abstract models for the evaluation of the consequences [Leve04]. Only errors observable at component boundaries are propagated at the high abstraction level without loss of accuracy. This allows to retain the advantages of structural modeling at much higher simulation speed.

This chapter uses the proposed NFP-aware simulation approach to efficiently implement concurrent multi-level fault simulation across gate and transaction level in an integrated simulation environment. The work is based on a structural fault model with an efficient concurrent fault simulator at gate level. The fault effects observable at gate level boundaries are propagated concurrently at transaction level, allowing a realistic fault evaluation at this level. The employed rollback mechanism is simple to use with the existing models and the transaction level simulators.

The following section presents the terminology used for multi-level vulnerability analysis. Section 7.3 presents the proposed methodology and the integration of the gate level sequential fault simulator within the TLMs. Section 7.4 explains the fault lifting approach from the gate to transaction level. Section 7.5 describes the low level DUA simulation in detail. Section 7.6 discusses the concurrent transaction level fault propagation approach and its implementation. Finally, section 7.8 presents the experimental results for the case studies and discusses the consequences of the system level reliability evaluation with the presented method.

7.2 Terminology

A *fault* is an abstraction of one or more physical defects affecting a circuit. A *fault model* is the formal abstraction of a class of defects. Fault models are essential to the tractability and automation of design for testability and design for reliability. An *error* is the consequence of a fault that has manifested in the circuit state captured in the memory elements. A *failure* is the inability of the system to accomplish its target mission when subject to a fault.

Fault simulation of a gate level model determines which faults cause errors at the outputs of that model. *Fault lifting* is the mapping of gate level errors due to the structural faults to a transaction level fault model. A *mutation* is an instrumentation for fault injection that becomes part of a (high level) model and modifies its behavior according to a fault model [Belt09]. *Mutation-based testing* is a common approach in the functional verification of high level hardware-software systems. In the approach presented here, the mutation is chosen to map well to structural faults and not to design bugs. The instrumentation is a part of the gate level simulation control (cf. section 7.5.2).

7.3 Multi-Level Vulnerability Analysis Overview

Fig. 7.1 presents an overview of the proposed multi-level vulnerability analysis approach. The system is simulated at transaction level. The workload on the input bus of the DUA is sent to the low level DUA simulation. As discussed in chapter 5, the pattern generation fills the abstraction gap between gate and transaction level by translating the transactions into the pin- and cycle-accurate protocol of the gate level component. It is also extended to lift faults from gate to transaction level, which will be explained in the next section.

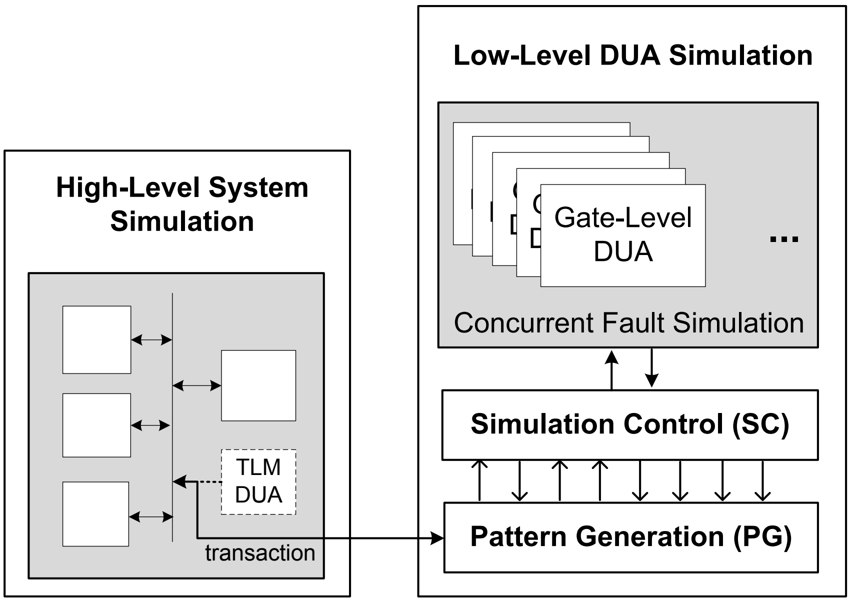


FIGURE 7.1: Overview of the multi-level vulnerability analysis approach

As we target soft errors, a concurrent fault simulation is used instead of the original parallel logic simulation with state prediction. The generated gate level input stimuli is sent to the simulation control to be injected to the gate level DUA model. The simulation results of the input stimuli are sent to the pattern generation to create the high level transaction and send it to the high level system simulation to proceed.

Fig. 7.2 depicts the principle of the proposed approach. The system and the target application are modeled at transaction level. For the hardware blocks and cores to be investigated, the gate level fault simulator instances are created using gate level models. The system is simulated at transaction level until a transaction with the component subject to fault simulation is requested. Upon the request, the fault simulation proceeds at gate level. If a fault causes an observable error, this error is lifted and the functional error propagation is performed at the transaction level. The result of the error propagation is then evaluated at high level and it is determined whether the fault eventually results in a system failure. For fault classification, the model from chapter 3.2 is used.

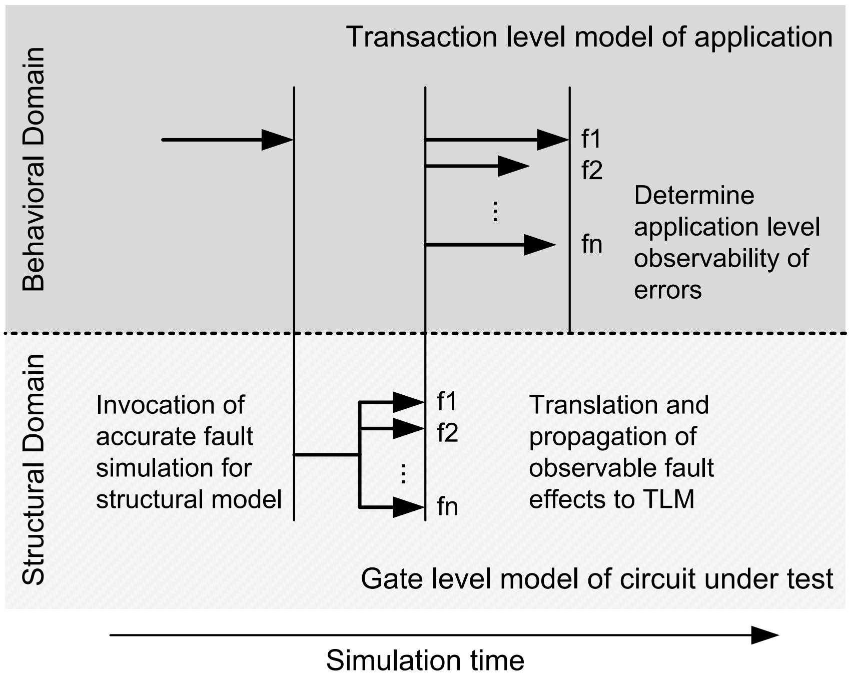

Behavioral Domain
Transaction level model of application
f1
f2
fn
Determine application level observability of errors
Structural Domain
Invocation of accurate fault simulation for structural model
f1
f2
fn
Translation and propagation of observable fault effects to TLM
Gate level model of circuit under test
Simulation time

FIGURE 7.2: The fault simulation sequence

7.4 Fault Lifting to TLM

For the fault simulation with high accuracy, the mapping, binding and timing of the TLM must be taken into account. *Mapping* is the process of selecting an actual implementation for the functional system model. It determines what the hardware components require to implement the system's functionality. *Binding* involves scheduling the steps required to provide the system's functionality and assigning them to the hardware components selected during mapping. Since TLMs are often used for design exploration, it is obvious that in early design steps, not all the system components may be mapped. However, error masking often exists, even in the function provided by the system itself or in the application running on the system. This inherent masking of operations in a data-flow graph is called *transparency* [Hill04]. The method presented here takes full advantage of the transparency in the behavior.

In the multi-level approach, only the component subject to fault injection is required to have a gate level model. Hence, only a partial mapping of the entire system is required and binding may be limited to the components subject to fault injection. As a result, the approach can be used early to evaluate mapping and binding decisions and explore design alternatives with respect to reliability.

The timing accuracy of the transaction level models can range across several orders of magnitude and the designer has a great freedom in modeling the timing aspects. On the other hand, the RT and gate level models are usually at least cycle-accurate. Obviously, there may be structural faults that can impact the temporal behavior of a sequential component and for example lead to longer completion times of certain operations. System failures due to such faults may be masked in a loosely timed TLM. In order to increase the accuracy for this type of failures, adaptive timing accuracy [Rade08] is used in the direct surrounding of the component subject to fault injection.

The presented vulnerability analysis approach uses the TLM fault model [Belt09] to lift the gate level errors to the TLM. Instead of random mutations, it accurately reflects the effects of transient faults on transactions issued to and from the component subject to fault injection. The following mutations are used:

- corruption of a parameter such as address, payload, transaction state or delay;
- transactions falsely issued by the fault-simulated component.

For this purpose, there must be a correspondence between the errors at the gate level and a mutation at the behavioral model. A structural fault within a gate level module can be observable at the module's outputs. These errors can be classified into errors affecting the data, address or control outputs of the gate level module. As discussed before (cf. Fig. 5.3), an error at the gate level can correspondingly affect one of the parameters at TLM level. For example, using TLM 2.0, an error in the "read request" signal of the memory at gate level can result in the corruption of the TLM_READ_COMMAND attribute of a transaction being issued. On the other hand, a fault in the datapath that results in an erroneous data output may affect the *m_data* parameter of the TLM payload.

In order to deal with the faults that affect the time behavior (e.g., the transaction delay), the TLM model must be timing aware. Differences in transaction duration caused by faults can impact the system performance or cause timeouts to be triggered. For blocking TLM calls with *b_transport*, the delay is passed along with the transaction. Errors on the control lines that affect the delay are translated to the TLM delay parameter. Timeouts are reported to the originator through the TLM_INCOMPLETE_RESPONSE. For non-blocking calls with *nb_transport*, the actual simulator time is advanced before the response is communicated.

7.5 Low Level DUA Simulation

This section describes the individual components of the low level DUA simulation in detail.

7.5.1 Pattern Generation

Pattern generation is responsible for protocol translation between transaction and gate level. An accurate protocol translation from the transaction to the pin- and cycle-accurate protocol of the gate level model is achieved by decomposing each transaction. The complex values are mapped to the binary values, and additional control signals are provided at gate level which are not explicitly represented at transaction level (e.g., the reset or write-enable signals) as explained in section 5.4.1.1.

The pin- and cycle-accurate values are processed by synchronous fault simulation of the gate level model, where in each simulation cycle, a new data vector is passed to the simulator. The simulation results for each cycle is evaluated. If the error is observable at the gate level model boundary, it is propagated at transaction level as detailed in section 7.6.

7.5.2 Simulation Control

Simulation control is responsible for fault lifting. Faults can lead to transaction properties that are not part of the fault-free specification of the communication protocol. For example, faults can cause transaction types such as burst transfers that are not part of the fault-free communication. Hence, the simulation control must model more communication aspects for the fault simulation case than for the fault-free case.

The unknown values cannot easily be represented in a regular TLM. Therefore, the simulation control replaces them by random values or by values for the worst or best case. It depends on whether a pessimistic or optimistic bound of system reliability is to be evaluated. The exact strategy depends on the function of the given component. Beside detected and undetected, low level simulators introduce a third fault class called *possibly detected*. A similar probabilistic consideration of the unknowns must be done by the simulation control. In case of the worst case analysis, if unknown values appear at the gate level boundary, they are propagated at the transaction level several times: By replacing them with zeroes, ones, random values, and values that are the inverse of the good value simulation. If any propagation results in failure of the application, the fault that resulted in the unknowns is classified as possibly detected.

A special consideration in the simulation control is the timing deviation and uncertainty. The simulation control must keep track of the fault simulation time, at which errors occur. To detect faults that cause the gate level module to exceed the response time of the good simulation, it must advance the fault simulator time beyond the fault-free simulator time. The upper bound for this is the timeout specified by the bus model. Again, the simulation control must model details that go beyond the specification of the DUA and model the allowed specification of the communication mechanism and its correspondence to errors. For example, transaction ordering errors can occur subject to faults, even if the good simulation assumes some specific ordering. Furthermore, since faults can lead to unknowns on control signals, the exact duration of a transaction can even become undetermined.

7.5.3 Concurrent Fault Simulation

During the gate level fault simulation, a large degree of parallelism can be exploited by efficient evaluation of faults, patterns and gates in parallel [Koch10]. Here, the concurrent fault simulation algorithm [Ulri73] is used to achieve high efficiency. It simulates several faults in parallel, such that gains are obtained by common sensitization criteria amongst faults.

The precision of gate level designs allows to model multiple aspects of a system that are usually not considered at transaction level, for example multi-valued logic, multiple clock phases and reset signals. In a fault-free system, the multi-valued logic is easily translated (e.g., for a well-behaved bus). Multiple clock phases are deterministic if their relationship is known. Reset signals flush any unknown values from the gate level model.

In a gate level component that is subject to structural fault injection, these modeling aspects may be visible at the component boundary: Some faults affect buses and cause conflicts that should be considered at transaction level. Multiple clock phases that were previously in a known relationship become undetermined and lead to race conditions. Any strike on reset signals, due to their high fan-out, result in any combination of uninitialized latches or flip-flops that show up as unknowns at the gate/transaction level boundary.

7.6 Optimizations at Abstraction Boundaries and in TLM

The gate level fault simulation determines the observability of fault effects at the primary outputs of the gate level model. It can take full advantage of a plethora of techniques that significantly improve the computational efficiency such as concurrent simulation of faults, special data structures and algorithmic optimizations [Lee92]. To determine if a fault has any undesirable impact on the system functionality, its effect (error) is propagated in the system and evaluated within the application context. This section introduces an efficient, parallel error propagation and evaluation method at transaction level.

7.6.1 Error Injection Mechanism

An observable error at the boundary between the gate and the transaction level is injected in an atomic transaction and further propagated and evaluated in transaction level simulation. The specific mutation of a transaction is determined by the simulation control whenever the gate level simulator requests fault propagation. The simulation control determines mutations based on the information provided by the gate level fault simulator.

In order to keep the simulation effort low and classify faults quickly, initially just a subset of the outputs at gate level is evaluated to determine the type of the mutation. For instance, if at a given time, an output specifying data validity of the corresponding port is deasserted in both the fault-free and the faulty machines, the data provided by the port does not need to be verified. In this case, no fault propagation at transaction level follows. Fault propagation is also given up if the error is certain to be masked by the bus protocol. For example, error propagation is not requested if a fault affects only the bus address bits that are masked out by the bus masking bits. Such faults are classified as benign already in the simulation control to avoid superfluous error propagation.

7.6.2 Evaluation of System Failure Conditions

A system failure is defined as a deviation of the system behavior from its specification. The expected behavior included in the test scenarios from functional verification is reused in our fault simulation approach to construct the simulation control and to evaluate the overall system behavior. In a holistic model including the environment (e.g., a stability controller within a vehicle), certain system properties can be verified under faults.

If the component subject to fault simulation is self-testing or self-checking [Lala01], this mechanism is used for error detection and fault classification by checking the output for the non-codewords. Such errors are communicated with an appropriate transaction response and lead to an early abort of error propagation. Similarly, assertions from functional verification, which usually compose built-in model instrumentation, are also reused. Assertions implement sanity checks to find faulty states and control flow violations. At system level, they check for the “instance out-of-bounds” exceptions.

To speed up the fault simulation, the transaction level fault propagation is halted as soon as there is enough information for fault classification. In case of signal processing applications, a checksum is calculated from the output data stream. The checksum is then evaluated and compared at the intermediate checkpoints. Assertions used for functional verification are reused to detect invalid behavior earlier than the checksum mechanism.

7.6.3 Concurrent Error Propagation

To efficiently propagate a large number of errors, it is important to have an effective means of reverting to the good machine state and undoing the changes made by the propagation. In gate level fault simulators, this is achieved by keeping track of the changes on a stack or by using tags or group IDs to identify data that differs from the good machine state. However, this is not feasible in TLM simulations, since the models consist mostly of functional abstractions in the form of host-compiled codes. Beside code modification, the existing error injection approaches for TLM work with instrumentation of the compiled simulation binary [dSF09] or directly with the TLM simulation kernel [Na13]. However, with all these methods, one simulator session can only be used for a single injection.

The proposed error injection method is based on the concept of concurrent fault simulation with one fault-free machine and several faulty machines evaluated in parallel. The fault-free machine is running as the main process. Faulty machines are created quickly as sub-processes using operating system facilities. Since processes are protected from each other, the cost of a rollback amounts to terminating the child process that executes the faulty machine. Beside its low cost, the approach is truly concurrent on host computers with multiple cores.

This approach is easily implemented on top of any existing transaction level model. No changes to the simulation kernel are necessary and intellectual properties can be used as is. The evaluation of system failure or success can be done entirely in the faulty machine. Only for the fault classification mentioned before, communication between the good and the faulty machine processes must be established. However, the classification is easily enumerated and it can be communicated cheaply using the process return value upon termination of the faulty machine process.

7.7 Implementation

The multi-level fault simulation algorithm has been implemented based on the sequential gate level fault simulator *Hope* [Lee92] and the OSCI SystemC 2.2 and TLM 2.0 libraries. A C++ wrapper is implemented for the Hope fault simulator to integrate it with the object oriented

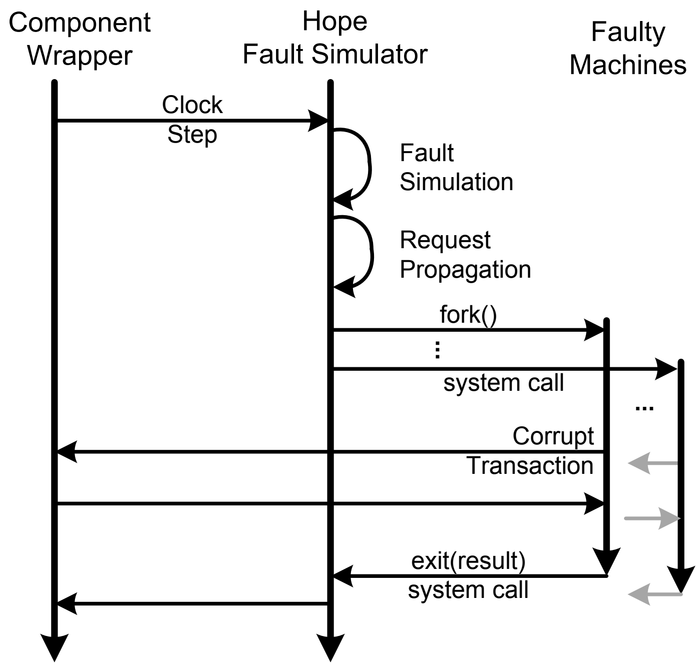


FIGURE 7.3: Steps in multi-level fault simulation

SystemC simulation environment. In the Hope wrapper, relevant data structures and methods were exposed to obtain fault detection information. In addition, methods were added to initiate error propagation for faults visible at the gate level boundary. The pattern generation and simulation control are both implemented inside the wrapper. Separate instances of the Hope fault simulator are dynamically created for the considered gate level models. While the algorithmic optimizations in Hope target the stuck-at fault model, they can be extended to other structural fault models using the concept of conditional stuck-at faults [Wund09].

Fig. 7.3 shows the interaction between the core wrapper, the Hope fault simulator instance and the faulty machine at transaction level. The gate level fault simulator is a part of the good machine. Faulty TLM machines are created as necessary using the POSIX $fork()$ command. This allows to quickly create a faulty machine since Unix implements process forks with copy on write. Consequently, the fault-free and the faulty machines share the same memory regions until a memory page is modified in the faulty machine. Overall, the mechanism is transparent for many system models, but some care must be taken for file handles opened for writing in the simulation environment. The file handles should be closed in faulty machines.

7.8 Evaluation

The proposed multi-level fault simulation approach is evaluated concerning the fault classification accuracy and performance. We target the higher bound of fault detectability, and thus follow the approach of multiple propagation for unknown values (cf. section 7.5). The experiments are performed on three classes of applications executed on an AMBA based SoC with a LEON3 processor. The SoC, as shown in Fig. 7.4, contains hardware accelerator cores for Triple-DES (Data Encryption Standard), AES (Advanced Encryption Standard), as well as for 2D-DCT (two-dimensional discrete cosine transformation). The AES core is equipped with the built-in self-test (BIST) facilities.

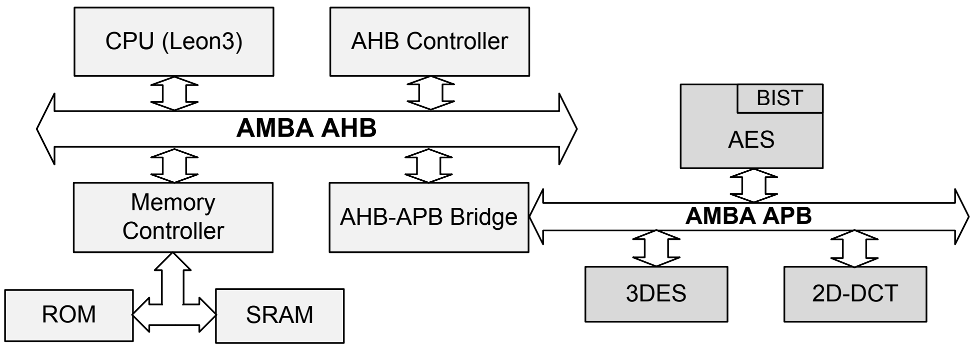


FIGURE 7.4: SoC with Triple-DES, AES and DCT accelerators

Except for the validation, the experiments were run on a multiprocessor system with 8 Intel Xeon CPUs (2.8 GHz). The memory usage did not exceed 250 MB in any of the experiments.

7.8.1 Validation

The proposed approach is validated in a traditional fault injection environment based on a state-of-the-art commercial simulator. The SoC is modeled at RTL, except for the core subject to fault injection which is modeled at gate level.

In each simulation run, a single transient fault is evaluated. The simulation is run until the result of the application is produced. A time-out is set in order to detect faults that lead to deadlocks and unacceptable delays. The simulation outcome is evaluated by the fault injection environment and the fault is classified accordingly. Due to the high computational cost, a random sample of 3000 faults per core is investigated this way, so that the per-application validation effort doesn't exceed two weeks.

Each fault is classified according to the categories from section 3.2. In the validation experiments, the following cases are discerned:

- Covered: The classification from the proposed method agrees with the validation experiment.
- False corrupt: The fault causes a silent data corruption (SDC) in the proposed method, but is benign or causes a detected, unrecoverable error (DUE) in the validation experiment.
- False benign: The fault is benign in the proposed method, but causes an SDC or DUE in the validation experiment.
- False detected: The fault results in a DUE in the proposed method, but is benign or causes an SDC in the validation experiment.

Validation experiments of the proposed method and the reference simulator were conducted on a farm of workstations equipped with AMD Athlon 64 Dual Core Processors (2.4 GHz) and 4 GB of RAM.

7.8.2 Triple-DES Encryption Application

The first case study is based on an encryption application utilizing the Triple-DES core in the SoC from Fig. 7.4. It encrypts a string of 64-bit words using a 64-bit key. The software part of the application is responsible of the block-wise transfer of data to the core and the read-back of results. This application is chosen as an example that exhibits almost no inherent masking.

The Triple-DES dedicated core has been obtained from OpenCores[1] and synthesized for the LSI10k generic library. It contains 19,917 logic cells and 53,010 stuck-at faults. In the following, we present the results for the system level effects of faults in the Triple-DES core obtained by the proposed multi-level approach and discuss its performance and accuracy.

[1]http://www.opencores.org

Table 7.1 presents the system level fault masking in four scenarios. The first column specifies the type and the length of the input data set that is encrypted. The encryption keys were chosen randomly for each scenario. In the second column, we give the number of sensitized faults, i.e., faults that produce an observable change on the core boundaries but do not necessarily lead to errors at system level. The third and fourth column provide the number of SDCs and benign errors, respectively. As there is no error detection mechanism, DUEs would not occur (cf. section 7.6.2). In all scenarios, more than 99% of the faults that were sensitized led to an SDC at system level. This is explained by the fact that the results from the core are directly transferred to the system output, so, no data error masking takes place. The remaining 193 faults cause errors only during inactivity of the "data ready" signal and hence they are benign.

TABLE 7.1: Validation results for Triple-DES application (random sample of 3,000 faults)

Scenario	Faults sensitized	SDC	Benign errors
(1)	(2)	(3)	(4)
English 3.5 KB	32916	32723	193
Italian 21 KB	33247	33054	193
Latin 20 KB	32901	32708	193
Random 8 KB	32953	32760	193

Table 7.2 gives an insight into the performance of the presented approach on the 8-core machine. Column "Num. sim. contexts" gives the number of fault propagations performed using *fork*. The third and fourth column provide the CPU-time spent for the concurrent fault simulator and for the execution of the TLM model, respectively. The time needed for the child process creation and termination is included in the latter. The last column provides the overall run-time of our approach. The Hope CPU-time proved to be one fifth to one half of the total execution time.

TABLE 7.2: Validation results for Triple-DES application (random sample of 3,000 faults)

Scenario	Num. sim. contexts	Gate level Hope CPU-time	TLM+Sys CPU-time	Overall run-time
(1)	(2)	(3)	(4)	(5)
English 3.5 KB	37983	1m 16s	4m 18s	5m 34s
Italian 21 KB	37396	5m 56s	5m 10s	11m 6s
Latin 20 KB	37809	5m 41s	5m 20s	11m 1s
Random 8 KB	37777	2m 21s	4m 23s	6m 44s

The validation was performed on a random sample of 3000 faults from the full set of 53,010 faults. A string of 3,576 ASCII characters was encoded using various keys. According to the classification from section 3.2, all the sampled faults were categorized as “covered” by the proposed multi-level method, i.e., no fault was mispredicted. The run-times are summarized in table 7.3. The first column lists the type of the key used for encryption, and the subsequent columns provide the comparison between the CPU time of the validation experiments (RT/gate level) and the proposed approach (TLM/Hope), both performed on the same Athlon machine. We achieved a perfect match under an average speed up of about 13,200x.

TABLE 7.3: Validation results for Triple-DES application (random sample of 3,000 faults)

Scenario	CPU-time	
	RTL/gate	TLM/Hope
All “0”	233h	67.1s
All “1”	243h	88.6s
Sequence	234h	51.4s
Random	242h	53.0s

7.8.3 AES Encryption Application

The second case study is based on the self-testable AES core within the SoC from Fig. 7.4. The core is able to encrypt a string of 64-bit words using a 64-bit key. The BIST functionality provides a 64-bit signature that is unique for the core—any deviation in the signature indicates that the core is faulty. The software part of the application is responsible for the block-wise transfer of data to the core and the read-back of results. Similar to the Triple-DES application, AES exhibits little inherent masking.

The self-testable AES core has been obtained from the authors of [DN10]. It has been synthesized for LSI10k library and contains 22,985 logic cells and 53,850 stuck-at faults.

Tables 7.4 and 7.5 present the system level fault masking and are analogous to tables 7.1 and 7.2, respectively. The first four applications are equivalent to those discussed in the previous case study. The following four applications are similar to the first four, except that they begin with a BIST run. If a fault is detected by the BIST, it is classified as DUE. If a fault is not classified as a DUE and results in failure of the application, it is considered as an SDC. If a fault is not detected by BIST and does not influence the application, the fault is benign. In each scenario, more than 96% of the faults that were sensitized were either detected by BIST

(DUE), or resulted in SDC. BIST was proven to detect 95% of the faults from the full set of 53,850 stuck-ats. It failed to detect at least 1411 faults (2.6%) that led to a failure in the sixth application. The undetected faults were found to reside either at the primary inputs or at the gates in the direct neighborhood thereof. As the BIST runs with a constant seed that is applied internally, the faults at the data and key inputs do not propagate and thus are not detected.

TABLE 7.4: Fault masking results for AES application

Scenario (1)	Faults sensitized (2)	SDC (3)	DUE (4)	Benign errors (5)
English 3.5 KB	45836	44274	-	1562
Italian 21 KB	46213	44601	-	1612
Latin 20 KB	46057	44478	-	1579
Random 8 KB	46071	44487	-	1584
BIST, Eng. 3.5 KB	52973	1334	51146	493
BIST, Ita. 21 KB	53076	1411	51146	519
BIST, Lat. 20 KB	52986	1328	51146	512
BIST, Rnd. 8 KB	53053	1398	51146	509

Table 7.5 provides performance details for the previously discussed scenarios. As a single BIST-run takes 512 cycles and faults can be dropped only after the run is complete, the last four scenarios require a considerable effort at the low level, which approaches half of the overall run-time. However, as a BIST run results in massive fault dropping, the overall run-time is little affected by the length of the application itself. For this reason, even though the applications differ in the length of the text subject to encryption, they exhibit similar run-times.

TABLE 7.5: Run-time results for AES application

Scenario (1)	Num. sim. contexts (2)	Gate level Hope CPU-time (3)	TLM+Sys CPU-time (4)	Overall run-time (5)
English 3.5 KB	45670	24s	1m 59s	2m 23s
Italian 21 KB	45676	48s	2m 7s	2m 55s
Latin 20 KB	45646	47s	2m 8s	2m 55s
Random 8 KB	45562	31s	2m 0s	2m 31s
BIST, Eng. 3.5 KB	54639	10m 12s	10m 3s	20m 15s
BIST, Ita. 21 KB	54716	10m 18s	9m 4s	19m 22s
BIST, Lat. 20 KB	54633	10m 13s	8m 30s	18m 43s
BIST, Rnd. 8 KB	54703	10m 13s	8m 40s	18m 53s

Like in the previous case study, validation is performed by encoding a string of 3,576 ASCII characters with various keys. A random sample of 3000 faults was chosen from the full set of 53,850 faults. For all the sampled faults, our approach provided the correct classification and all faults were categorized as “covered” with respect to section 7.8.1. The execution times for the validation experiments are summarized in table 7.6, which is analogous to the previously discussed table 7.3. We achieved a perfect match under an average speed up of about 6,400x.

TABLE 7.6: Validation results for AES application (random sample of 3,000 faults)

Scenario	CPU-time RTL/gate	CPU-time TLM/Hope
All “0”	278h	21.9s
All “1”	281h	22.2s
Sequence	288h	22.1s
Random	278h	22.2s
BIST, All “0”	298h	4m 59s
BIST, All “1”	306h	4m 58s
BIST, Sequence	290h	5m 6s
BIST, Random	295h	5m 9s

7.8.4 JPEG Encoder Application

In case of the JPEG encoding application, we study the strong impact of error masking. The baseline JPEG encoding algorithm can be decomposed into four steps:

1. color transformation
2. two-dimensional discrete cosine transform (2D-DCT)
3. quantization
4. lossless compression

JPEG encoding is performed by the SoC architecture from Fig. 7.4. As the 2D-DCT is the most computationally expensive operation, it is accelerated by the hardware core. All other operations are performed by the LEON3 processor. The 2D-DCT core has been obtained from OpenCores and synthesized for the LSI10K library. It contains 28,001 logic cells and 78,914 stuck-at faults. In the following, we study the performance and accuracy of our approach for several case studies with various images.

Table 7.7 describes the effect of the faults in the 2D-DCT core. The first column specifies the type and pixel dimensions of the image that is encoded in each scenario. The subsequent columns are analogous to the previously discussed table 7.1. Compared to the Triple-DES and AES applications, there is a much larger proportion of sensitized faults that lead to benign errors, i.e., faults that are masked by the application, although their effect is observable on the core boundaries. Among the sensitized faults, from 16% up to 36% of the faults are benign. This is due to the error masking property of the quantization step.

TABLE 7.7: Fault masking results for JPEG application

Scenario (1)	Faults sensitized (2)	SDC (3)	Benign errors (4)
White 8x8	25297	16295	9002
Black 8x8	22729	14420	8309
Noise 8x8	48794	34064	14730
Fruits 64x48	64797	54141	10656

The run-time results for the approach running on the previously mentioned 8 core machine are gathered in table 7.8. The number of simulation contexts clearly depends on the image size, as for each 8x8 pixel block, the effects of all the sensitized faults that were not yet classified as SDC have to be analyzed. Due to the masking property of the JPEG application, a large number of error propagations occurs before the associated fault is classified as SDC and can be dropped. Due to the fault dropping, the run-time is not linear with the image size. For the image composed of 48 pixel blocks, the run-time increases just 7 times compared to the scenario with a single block.

TABLE 7.8: Run-time results for JPEG application

Scenario (1)	Num. sim. contexts (2)	Gate level Hope CPU-time (3)	TLM CPU-time (4)	Overall run-time (5)
White 8x8	25927	56s	1m 15s	2m 11s
Black 8x8	25399	57s	1m 16s	2m 13s
Noise 8x8	47892	1m 14s	3m 48s	5m 02s
Fruits 64x48	279563	9m 11s	21m 5s	30m 15s

The validation experiments were conducted in a setting identical to the one used for Triple-DES. Due to the high computational effort, the validation was run for the scenarios with a single 8x8 pixel image. The results are summarized in table 7.9. It is analogous to table 7.3 except for the two additional columns that give the number of “benign faults” and the number of faults categorized as “false corrupt” (cf. section 7.8.1) among the 3,000 faults in the sample. From 52% up to 82% of the sampled faults were found benign, which is attributed to the error masking property of the JPEG quantization step. The effects of 2 to 7 faults per scenario were mispredicted and classified as “false corrupt”, which is pessimistic. They were found to either result in a period of an unknown value on the “data ready” signal while the signal should have been inactive, or generate additional active pulses on this signal after the data becomes invalid.

TABLE 7.9: Validation results for JPEG application (random sample of 3,000 faults)

Scenario	Faults benign	False corrupt	CPU-time RTL/gate	CPU-time TLM/Hope
White 8x8	2377	6	115h	29.9s
Black 8x8	2463	7	117h	31.3s
Sequence 8x8	2067	2	119h	36.9s
Noise 8x8	1580	2	148h	42.8s

In the validation experiments, these faults were classified as benign only due to the short length of the application and favorable synchronization. Under unfavorable circumstances, they could in fact cause SDCs. However, even if we assume the validation experiments to be the golden reference, we achieve a match for 99.8% of the faults under an average speed up of 12,700x.

7.9 Summary

This chapter used the multi-level NFP-aware simulation approach to present a fault simulation methodology for vulnerability analysis. The method allows consideration of the structural faults in a multi-level simulation environment at gate and transaction level. The simulation time is improved by four orders of magnitude using an efficient concurrent fault simulator at gate level and concurrent error propagation at transaction level. The methodology and the error propagation mechanism allow reusing the TLM models from design space exploration. The accuracy of the precise gate level simulation is achieved.

Chapter 8

Application on Aging Prediction

8.1 Introduction

As the technology nodes scale further, aging processes arise as another non-functional bottleneck that requires consideration in early design phases [Kim08, Alam05, Kean10]. Several approaches at different abstraction levels propose solutions to ensure reliability and performance in the presence of circuit aging.

Traditionally, guardbanding is used to overcome delay degradation due to aging [Chan11, Agar08]. In this approach, operating frequency is reduced or supply voltage is increased based on the worst degradation that may occur due to the worst combination of temperature, voltage and workload over the specified lifetime of the system. If the guardbands are too pessimistic, throughput and power cost of the design are incurred, and the product is less competitive. If circuit degradation is underestimated, the circuit may fail before expected and force expensive redesigns [Agar08, Chan11, Lore10]. Another method is to use on-line circuit failure prediction using on-board aging monitors. However, the correct placement of monitors is crucial for on-time failure prediction [Agar08]. Apart from the selected aging mitigation approach, an advance knowledge of critical paths and the aging-related performance degradation is required to realize an efficient, aging-aware hardware design.

Early aging analysis enables an accurate estimation of the aged circuit performance before manufacturing. If no information about the workload is available, the worst case aged circuit characteristics can be obtained. With the workload information, the degradation can be estimated more accurately, as it strongly depends on the input signal over time [Lore10].

Among aging mechanisms, NBTI and HCI have gained more weight in the nanoscale era [Giel08]. They cause systematic reduction in transistor parameters (e.g., drain current, threshold voltage, capacitance, etc.) and have been a persistent reliability concern for CMOS technology generations below 130 nm node [Alam07].

This chapter uses the window-based simulation with the piecewise evaluation approach to develop a flexible, numerical simulation engine for aging prediction early in the design phase. The aging-related observables are acquired using the window-based, NFP-aware simulation (cf. chapter 5), while the piecewise evaluation approach estimates aging degradation (cf. chapter 6). Environmental changes are also considered dynamically during aging prediction. The proposed approach can be used to evaluate the impact of aging mitigation techniques under various operating conditions, including different voltage scaling, power gating, and activity management scenarios. It can also be used to provide complementary information for low level approaches, as well as online techniques.

The rest of this chapter is organized as follows: Section 8.2 presents the state of the art for aging prediction techniques. Piecewise aging evaluation approach for NBTI and HCI aging mechanisms and power and temperature as interdependent NFPs are discussed in section 8.3. It also explains how the interrelation between these parameters are modeled. Observable collection is discussed in section 8.4. Section 8.5 discusses a case-study and shows the experimental results. At the end, section 8.6 concludes the chapter with a short summary.

8.2 State of the Art

Most of the state-of-the-art reliability simulation methods try to emulate the degradation process of aged devices in a repetitive scheme. They are based on physical failure mechanisms and contain major wear-out models mostly for HCI and NBTI mechanisms. A set of parameters for each of these failure mechanisms are identified and algorithms for extracting them in a given technology are developed using accelerated tests. A circuit simulator, such as SPICE, is employed to calculate the electrical parameters of fresh and degraded devices to predict their degradation or failure [Bern06].

A fundamental model and methodology for reliability simulation were first proposed and implemented in the Berkeley reliability tools (BERT) [Tu93]. It predicts the degradation due to HCI. The circuit is simulated to capture the relevant current and voltage waveforms at transistor terminals. The waveforms are applied to generate the degraded transistor model for each transistor. The degraded circuit performance is obtained by a second simulation with the aged transistors. The main advantage of BERT is accuracy and SPICE modeling technology compatibility. However, the design parameters have to be correctly extracted. Once the degraded circuit is produced, the behavior under a new application or a set of other operating parameters cannot anymore be analyzed [Bern06].

The authors in [Wang07b, Paul05] propose an NBTI prediction approach at transistor level. Additionally, the authors in [Chak04] use a numerical framework to model NBTI at this level. Bernstein et al. [Bern06] propose a failure-rate based simulation methodology for NBTI, EM, HCI and TDDB together at transistor level (SPICE).

Transistor level techniques require the circuit model at this level and can simulate up to several thousand transistors. Furthermore, realistic input vectors are needed which might not be available [Lore10]. This implies that these methods are not proper solutions for complex digital circuits and cannot efficiently be used early in the design flow. However, they can be used to characterize gate models [Lore12].

Gate level aging simulation methods, such as [Wu00, Wang07d, Wang10], are mostly look-up table (LUT)-based approaches. An LUT containing the aged gate delays is built using transistor level tools such as SPICE for a certain use profile. A use profile consists of a set of environmental conditions (temperature, supply voltage, duty cycle of the input signals, process conditions, etc.) over the lifetime. Whenever the use profile changes, the entire cell library must be re-characterized. Furthermore, if the workload of a cell should be considered, a lot of different aged LUTs must be generated for each cell.

The LUT is modeled as a function of the input signals, the output load capacitances and the transistors' threshold voltages. The delay values are usually estimated for a short simulation run and extrapolated for the lifetime. For example in [Wang07d, Wang10], the gate delays are extrapolated from SPICE simulation with the Chebyshev polynomial series.

Some works such as [Noda10] use the stress factor, α, to estimate NBTI degradation at gate level. α is obtained for each transistor using random pattern simulation. However, dynamic changes of α during system operation are ignored. Moreover, the effect of temperature variation on NBTI is not considered in these works.

Beside the above mentioned approaches, a gate model is proposed in [Lore12, Lore09] to perform an aging-aware static timing analysis (STA) at gate level. NBTI and HCI are considered as target aging mechanisms and either the worst-case values or the probabilistic methods can be used to obtain the observables. The presented approaches are flexible in terms of degradation equations and consider that the transistors of a gate degrade individually. Lorenz et al. [Lore10] use the same approach at the macro-cell level and obtain a considerable speed up (average 27X for ISCAS'85 benchmark) in comparison to the gate level STA, but with the same accuracy. However, the gate profile in any of the approaches does not consider environmental variations during the simulation.

At microarchitectural level, the authors in [Abel07] introduce techniques to reduce the NBTI effect on microprocessors. Oboril et al. [Obor12] propose an aging framework that integrates a performance analysis tool with temperature and power estimation tools for accurate aging prediction. To deal with the lack of structural data at microarchitectural level, it is assumed that all the transistors have the same stress factor.

Some works tried to offer high level, accurate predictions using multi-level approaches. A switch/gate level approach to predict NBTI degradation is proposed in [Salu08]. A SPICE level simulation is used to identify the stress on PMOS devices under varying input conditions for various gate types. The degradation is expressed using the power law:

$$\Delta V_{th}(t) = \alpha t^n$$

The parameters α and n are statistically determined using gate level simulation of a small fraction of circuit lifetime. The effect of environmental variations on NBTI are not considered in the predictions.

New-Age [DeBo09] is an NBTI framework for microarchitectural components. It uses an RTL model of the system to perform NBTI-aware timing analysis. The switching activity and delay of each gate is estimated using the average system simulation. The switching activity is used for power estimation and in turn temperature analysis. The best- and worst-case analysis is performed to predict the degradation. Power and temperature analysis are done outside the framework; therefore, adapting them to different designs may require an additional effort. The framework cannot be used at early design phases while a holistic system simulation is required for the timing analysis phase.

8.3 Piecewise Aging Evaluation

To study the performance degradation due to aging, the first step is to find proper models. The models must describe respective aging mechanisms accurately by including the technology parameters and interdependent NFPs.

In the following, models for NBTI and HCI degradation are discussed and the relevant parameters and interrelated NFPs are extracted. NBTI and HCI are affected by temperature, while temperature and power have a mutual effect on each other. Therefore, models for temperature and power are also presented and the relevant observables are discussed in this section.

8.3.1 Piecewise Aging Models

The performance degradation due to aging is usually studied by changes in the threshold voltage (V_{th}) of the transistors. NBTI and HCI increase the V_{th} by time which causes delay in the critical paths of the circuit. Chapter 3 discussed models for V_{th} increase under NBTI and HCI mechanisms. In the following, we present the piecewise equivalent of the aforementioned models.

8.3.1.1 Piecewise NBTI Model

The NBTI effect (cf. section 3.3.1) causes a gradual positive shift in PMOS V_{th} when the transistor is stressed, i.e., when negative voltage is applied to the PMOS gate. Removal of the stress can partially anneal the interface traps resulting in partial recovery [Wang07e, Rang03, Alam03]. Due to the V_{th} degradation, NBTI results in poor drive current and lower noise margin which shortens the device lifetime and degrades the overall circuit performance [Bhar06, Wang07b, Alam03]. NBTI is exacerbated by temperature escalation [Alam05, Chak04]. Temperature linearly changes with the total power dissipated by the chip [Pedr06]. Thereby, an accurate NBTI prediction should allow for temperature variations and power consumption.

For NBTI modeling, equation 3.14 (cf. section 3.3.1) is rewritten in the piecewise form to give an upper bound of V_{th} at the i^{th} window with the size of W cycles (or at time t_i where $t_i = W \times i$):

$$\begin{aligned} \Delta V_{th}(i) &= \frac{\sqrt{K_v(i)^2 \cdot \alpha(i) \cdot T_{clk}}}{1-\beta(i)^{\frac{1}{2n}}} \\ K_v &= (\frac{q \cdot t_{ox}}{\varepsilon_{ox}})^3 \cdot K_1^2 \cdot C_{ox} \cdot (V_{gs} - V_{th}) \cdot \sqrt{C(i)} \cdot \exp(\frac{2E_{ox}}{E_0}) \end{aligned} \tag{8.1}$$

$\alpha(i)$ is the transistor stress factor during the last evaluation window. α shows the fraction of time in which the transistor is stressed and is identified as a low level observable. α acquisition is discussed in section 8.4. $C(i)$ is the diffusion constant expressed with the following equation:

$$C(i) = T_0^{-1} \cdot \exp(\frac{-E_a}{k \cdot T(i)})$$

where $T(i)$ is the temperature at time t_i. Temperature variation depends on several parameters including the ambient temperature, the material and the cooling system of the chip. Therefore, it is identified as an independent NFP and is modeled separately in the aging prediction methodology.

The missing parameter $\beta(i)$ is calculated in a piecewise manner as

$$\beta(i) = 1 - \frac{2K_1 \cdot \sqrt{K_2 \cdot C(i) \cdot (1-\alpha(i)) \cdot T_{clk}}}{(1+\delta) \cdot t_{ox} + \sqrt{C_i \cdot (t_i' + W)}}$$

where K_1 and K_2 are technology constants, and t_i' is

$$t_i' = \frac{2K_1 \cdot \sqrt{K_2 \cdot C_i(1-\alpha_i) \cdot T_{clk}}}{(1-\beta'(i)) \cdot C(i)^2} - \frac{(1+\delta) \cdot t_{ox}}{C(i)^2}$$

where

$$\beta'(i) = \left(1 - \frac{\sqrt{K_v(i)^2 \cdot \alpha \cdot T_{clk}}}{\Delta V_{th}(t_i - W)}\right)^{2n}.$$

8.3.1.2 Piecewise HCI Model

In MOS transistors, carriers are accelerated (become hot) due to the lateral electric field near the drain junction. A small part of these hot carriers may become energetic enough to be injected into the gate oxide and cause a damage [Bern06, Obor12]. The rate of hot carrier injection is directly related to the channel length, oxide thickness, operating voltage, and transistor switching activity.

Like NBTI, HCI effect is usually modeled as V_{th} degradation (cf. section 3.3.2). The degradation model presented in equation 3.16 is simplified as in [Obor12] and is evaluated piecewise for the i^{th} window (or at time t_i where $t_i = W \times i$ and W is the window size) using the following equation:

$$\Delta V_{th}(i) = A \cdot \exp\left(\frac{-E_a}{k \cdot T(i)}\right) \cdot \sqrt{\theta(i) \cdot f \cdot (t_i' + W)} \tag{8.2}$$

where

$$t_i' = \left(\frac{\Delta V_{th}(t_i - W)}{A \cdot \exp(\frac{-E_a}{k \cdot T(i)})}\right)^2 \cdot \frac{1}{\theta(i) \cdot f}$$

Here, A is a technology constant, E_a is the activation energy, k is the Boltzmann constant, T denotes the temperature, f is the operating frequency, and θ_i is the transistor stress factor. Akin to NBTI, temperature dependence is considered in HCI evaluation. θ is an structural observable and is obtained through simulation.

8.3.2 Modeling Aging Interrelations

As discussed in section 8.3.1, temperature fluctuations on a die surface affect NBTI and HCI mechanisms. Accordingly, the temperature profile changes by the power dissipated by transistors. Fig. 8.1 shows the effect of different observables and NFPs on NBTI and HCI. As it can be seen, power affects temperature which in turn influences NBTI and HCI effects. To model these interrelations, several observables need to be estimated during the aging-aware simulation. The switching activity of the nodes are required for power estimation and HCI prediction while the signal probabilities are needed for NBTI prediction. Ambient temperature affects the die temperature. The interdependency of these parameters are modeled to accurately predict the amount of V_{th} degradation due to NBTI and HCI effects. Observables are acquired from aging-aware simulation discussed in section 8.4.

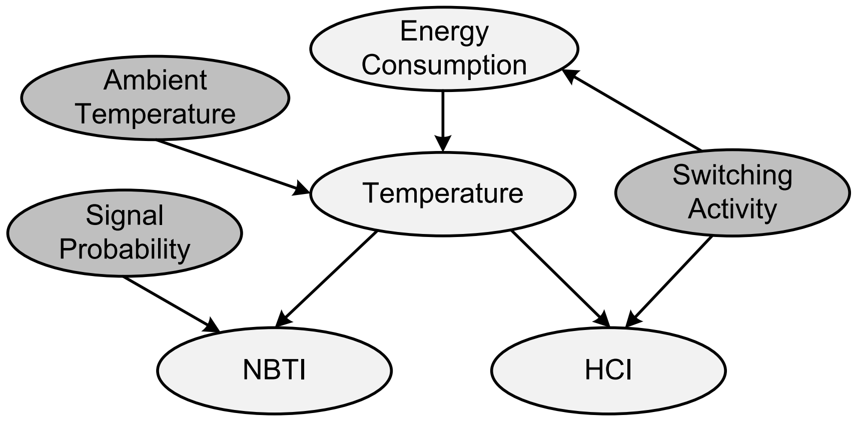


FIGURE 8.1: An example of NFP interrelation

The rest of this section discusses models for temperature and power and their interrelation with NBTI and HCI. Finally, we propose the structure of the evaluation server.

8.3.2.1 Temperature Profile

NBTI is highly susceptible to temperature variations. HCI is highly voltage dependent and is also affected by temperature. As the temperature increases, the leakage currents increase as well, thus more power is dissipated [Paci06]. Several works in literature address thermal modeling of VLSI circuits [Pedr06, Sing06, Huan04]. These models receive parameters such as dissipated energy, area, thermal resistance and material of the chip as input, and output the temperature of the chip.

A simple model to describe temperature can be formulated as [Pedr06]:

$$T_{chip} = T_a + R_\theta \times \frac{P_{tot}}{A} \tag{8.3}$$

where T_a represents the ambient temperature. R_θ is the equivalent thermal resistance of the substrate layer plus the package and the heat sink. P_{tot} is the total power consumption of the chip and A is the area. As it can be seen from equation 8.3, temperature has a linear relation with total power dissipation of the circuit which enforces accurate prediction of power.

This model forms the basis for more elaborate, block-based models, such as [Engl08, Paci06]. These models can be used to reflect the fluctuations of the temperature distribution across the die.

8.3.2.2 Power

Power dissipation in the substrate of a CMOS VLSI circuit is the sum of dynamic power dissipation, short-circuit power dissipation and leakage current [Pedr06]. For piecewise evaluation of aging models, we require the power dissipation in time units, i.e., energy. Energy is modeled recursively as in section 6.2.2.1. The parameter β in the model stands for the switching activity at the output of the gate. It is acquired by including the proper function for the switching activity in the gate model.

As the temperature distribution on the DUA cannot be considered without power dissipation of the complete SoC, architectural solutions for power—as discussed in section 8.5—should be combined with circuit level models to cover the total power dissipation of the multi-level SoC model.

8.3.2.3 Modeling interrelations

Fig. 8.2 shows the components of the aging evaluation server. It receives the current time t_i, the switching activity and the stress factor for NBTI and HCI at t_i. It also receives the degraded threshold voltages due to NBTI and HCI at time t_{i-1}, i.e., the last V_{th} estimation. The size of the window is provided by the designer. At the end of each window, the piecewise model for NBTI and HCI are evaluated based on the estimated temperature and dissipated power. The observables are evaluated using the window-based aging-aware simulation as discussed in the next section.

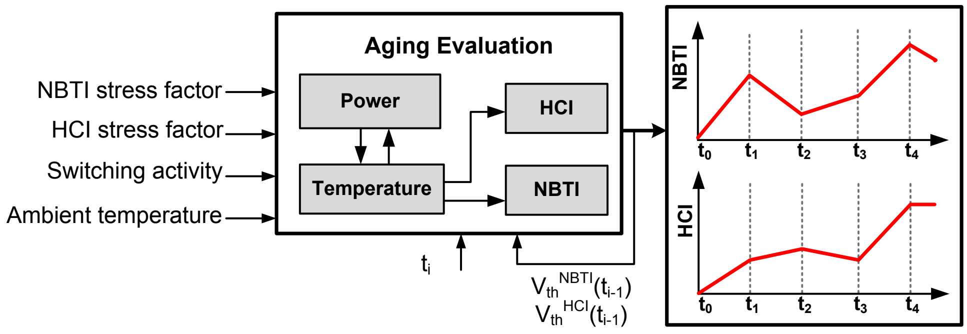


FIGURE 8.2: Aging evaluation components

8.4 Window-Based Aging-Aware Simulation

The aging-aware simulation acquires the NBTI and HCI observables. The high level simulation is carried out independent of the low level NFP-aware simulation. Thereby, the simulation speed of the low level model would not block the high level simulation procedure. To this end, gate models for the relevant observables are required. In the following, first the observable acquirement is discussed in detail. Afterward, the window-based aging-aware simulation is explained.

8.4.1 Modeling Observables

In this section, the modeling approach for the observables required for aging analysis are discussed and applied on a sample NOR gate.

8.4.1.1 NBTI Stress Factor

For a transistor c in gate g, we define the NBTI stress factor $\alpha_{[t_i,t_j]}$ as the cumulative time the transistor has been exposed to inversion in the time interval $(t_i,t_j]$. As the NBTI effect is ignorable for NMOS transistors [Stro06], $\alpha_{[t_i,t_j]}$ is defined only for PMOS transistors. c degrades when the gate terminal of the PMOS transistors are negatively biased with respect to their source or drain terminals. Two conditions must be fulfilled [Lore12]:

- logic “0” applied to c’s gate terminal
- logic “1” applied to c’s source or drain

To fulfill the conditions, each gate g is associated with a stress table S_g with 2^i integer entries, where i is the number of inputs to g. The entries in S_g correspond to the possible input patterns, e.g., $S_g(0)$ stores the number of times the pattern 00 has been observed at gate g with two inputs. Based on the mentioned conditions, the α function for each transistor is derived.

8.4.1.2 HCI Stress Factor

Both PMOS and NMOS transistors degrade due to HCI when carriers are accelerated and injected into the gate oxide. This is expressed by two stress conditions for transistor c [Lore12]:

- transition from off- to on-state at c
- conducting path from supply voltage/ground to c’s output load

For each gate g, a transition counter array (TCA) is associated to each transistor. This array counts all the transition events ($0 \rightarrow 1$, $1 \rightarrow 0$) during each run of the gate level simulation. The number of transitions for each node is monitored and added to the relevant TCA for that node.

For a transistor c, we define the switching activity $\theta_{[t_i,t_j]}$ as the cumulative time the input of the transistor switches from $0 \rightarrow 1$ or $1 \rightarrow 0$ and there is a conducting path from supply voltage/ground to c's output load in the time interval $(t_i, t_j]$.

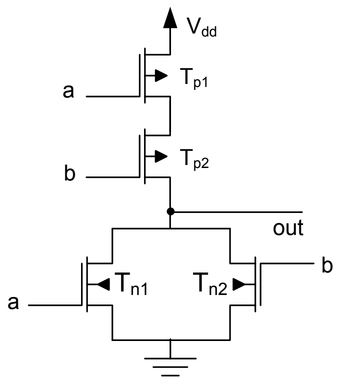
8.4.1.3 Gate Switching Activity

The switching activity $\beta_{[t_i,t_j]}$ of the gate g is defined as the fraction of time in which the output of the gate switches from $0 \rightarrow 1$ or $1 \rightarrow 0$ in the time interval $(t_i, t_j]$.

In the following, we show a two-input NOR gate model including the previously discussed observables. The same approach would apply to other gates as well.

8.4.1.4 Example: NOR Gate Model

Fig. 8.3 shows the structure of a 2-input NOR gate and the relevant stress table for NBTI degradation.



NBTI Stress Table

a	b	T_{p1}	T_{p2}
0	0	S	S
0	1	S	R
1	0	R	R
1	1	R	R

FIGURE 8.3: 2-Input NOR gate model

Based on the stress table, α_{c_i} in one cycle, c_i, is modeled as

$$\begin{aligned}\alpha_{T_{p1},c_i} &= !GATE(T_{p1},c_i) \\ \alpha_{T_{p2},c_i} &= !GATE(T_{p2},c_i) \ \&\ !GATE(T_{p1},c_i)\end{aligned} \tag{8.4}$$

where $GATE(T,c_i)$ is the input value of the transistor T at the i^{th} cycle. The value of $\alpha_{T_p,(c_1,c_k]}$, the NBTI stress factor of the transistor T_p in time interval $(c_1,c_k]$, is obtained by:

$$\alpha_{T_{p1},(c_1,c_k]} = \frac{\sum_{i=1}^{k} \alpha_{T_{p1},c_i}}{c_k - c_1}. \tag{8.5}$$

$\alpha_{T_{p2},(c_1,c_k]}$ is calculated likewise.

To calculate the stress factor for HCI, the equations should capture the switching activity on the transistor's gate:

$$\begin{aligned} GATE(T_{p1},0) = x, \quad & GATE(T_{p2},0) = x, \\ GATE(T_{n1},0) = x, \quad & GATE(T_{n2},0) = x \\ \theta_{T_{p1},c_i} &= [GATE(T_{p1},c_i) \oplus GATE(T_{p1},c_{i-1})] \ \&\ !GATE(T_{p2},c_i) \\ \theta_{T_{p2},c_i} &= GATE(T_{p2},c_i) \oplus GATE(T_{p2},c_{i-1}) \\ \theta_{T_{n1},c_i} &= GATE(T_{n1},c_i) \oplus GATE(T_{n1},c_{i-1}) \\ \theta_{T_{n2},c_i} &= GATE(T_{n2},c_i) \oplus GATE(T_{n2},c_{i-1}) \end{aligned} \tag{8.6}$$

The value of $\theta_{T_p,(c_1,c_k]}$ in time interval $(c_1,c_k]$ is obtained by

$$\theta_{T_{p1},(c_1,c_k]} = \frac{\sum_{i=1}^{k} \theta_{T_{p1},c_i}}{c_k - c_1}.$$

The accumulated value for other transistors is calculated similarly.

The switching activity to estimate the energy depends on the output of the NOR gate:

$$\begin{aligned}\beta_{nor,c_i} = {} & [(GATE(T_{p1},c_{i-1}) \oplus GATE(T_{p2},c_{i-1}))\| \\ & (GATE(T_{p1},c_{i-1}) \& GATE(T_{p2},c_{i-1}))\& \\ & (!GATE(T_{p1},c_i) \& !GATE(T_{p2},c_i))]\| \\ & [(!GATE(T_{p1},c_{i-1}) \& !GATE(T_{p2},c_{i-1}))\& \\ & (GATE(T_{p1},c_i) \oplus GATE(T_{p2},c_i))\| \\ & (GATE(T_{p1},c_i) \& GATE(T_{p2},c_i))]\end{aligned} \tag{8.7}$$

Based on equation 8.7, the value of $\beta_{T_p,(c_1,c_k]}$ in time interval $(c_1,c_k]$ is obtained by

$$\beta_{nor,(c_1,c_k]} = \frac{\sum_{i=1}^{k} \beta_{nor,c_i}}{c_k - c_1}.$$

The gate model strongly depends on the internal structure of the gate. Therefore, this approach facilitates the study of the gate structure effect on aging degradation and pattern reordering methods [Butz10] provided to mitigate aging effects.

8.4.2 Window-Based, Aging-Aware Simulation Procedure

The window-based aging-aware simulation procedure proceeds as described in section 5.5. Simulation as well as evaluation time is divided to windows. The observables are acquired by means of the proposed gate model. As the gate model integrates all the observables, they can be collected by a single simulation run. At the end of each window, the acquired observables are applied to the piecewise evaluation to predict the amount of degradation in that window.

8.5 Evaluation

The proposed method is applied to the case study of an SoC platform based on an OpenRISC-1200 processor[1]. As depicted in Fig. 8.4, the system is equipped with two display controllers

[1]OpenRISC Project, `http://opencores.org/or1k`

(VGA and LCD), an Inverse Discrete Cosine Transform accelerator (IDCT), and a Floating Point Unit (FPU). The processor communicates with the peripheral devices through Wishbone bus using single 32-bit read/write bus cycles.

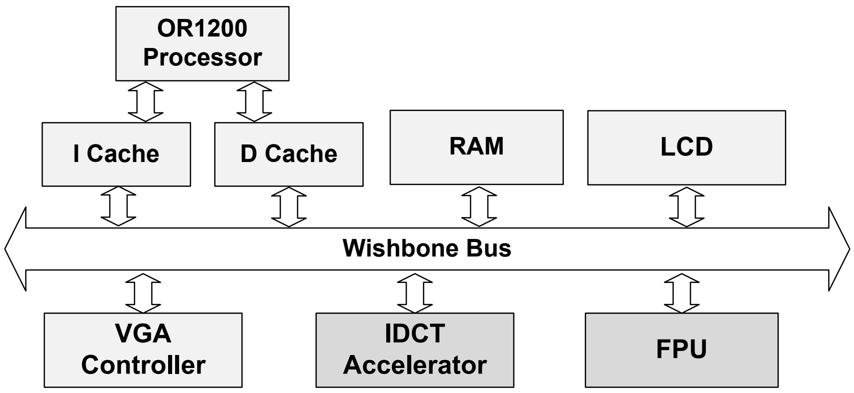


FIGURE 8.4: System architecture

The multi-level NFP-aware simulation is used to predict the NBTI and HCI stress factors with the actual system workload. The piecewise evaluation predicts circuit degradation of the IDCT accelerator and the FPU due to NBTI and HCI effects. Temperature variations of the die are also considered during the evaluations. For aging estimation, we assume the predictive technology model[2] (PTM) for the 45 nm technology:

$$\begin{aligned} V_{dd} &= 1.0\ [V] \\ V_{th} &= 0.2\ [V] \\ t_{ox} &= 1.75\ [nm] \end{aligned}$$

The goal of the experiments is to evaluate the performance and the accuracy of the multi-level simulation at the gate/transaction level. In addition, the performance and the accuracy of the piecewise evaluation approach is compared to an extensive, cycle-accurate aging analysis at gate level.

[2]`http://ptm.asu.edu`

8.5.1 Experimental Setup

The system is modeled functionally at transaction level using OSCI SystemC 2.2 and TLM 2.0 modeling libraries. Each functional unit is modeled as a set of concurrent, behavioral processes. The units are approximately-timed and communicate via TLM sockets.

As the IDCT accelerator is the most critical core in JPEG decoding, it has been chosen as the core under aging analysis. The IDCT accelerator has been obtained from OpenCores[3]. It is a sequential implementation of IDCT algorithm with integer precision. The core has been extended with two FIFO memories for the input and output data buffers, and a slave interface to Wishbone bus.

The FPU core has been extracted from MicroSparcII processor[4] and equipped with a Wishbone slave interface. It is a multi-cycle implementation of double precision.

To obtain the structural gate level models, the two DUAs were synthesized for the LSI10k generic library. The IDCT accelerator consists of 28,772 gates (113,686 transistors in CMOS technology) and 3,775 registers. The FPU requires 17,113 gates (62,362 transistors) and 630 registers.

An RTL model of the FPU and IDCT cores are generated automatically from the corresponding gate level model by *Verilator*[5]. RTL models are cycle-accurate descriptions of the DUAs in SystemC.

8.5.2 Sample Applications

For the FPU core, the following signal processing applications are considered in the case study:

- FIR: High-pass finite impulse response filter of order 14
- IIR: Low-pass Butterworth filter of order 5
- FFT: Fast Fourier transform for vectors of length 64

[3]`http://www.opencores.org`
[4]Oracle, `http://www.oracle.com/us/sun`
[5]`http://www.veripool.org/wiki/verilator`

- IFFT: Inverse FFT (vector length 64)
- JPEG: Image decompression with a floating point IDC transformation for an 8x8 pixel image

Floating point operations, such as addition, multiplication, and division are offloaded to the FPU. Random data (noise) is used as applications' inputs.

The IDCT accelerator has been evaluated in several applications with JPEG image decompression [Wall92]. The processor periodically decodes and displays a color image. The process of Huffman decoding, dequantization, and the final color conversion is performed purely in software, while the IDCT transformation is offloaded to the IDCT accelerator. The IDCT core performs the transformation on 8×8 pixel blocks. Four different image sizes are used for the experiments: 8×8, 16×16, 64×64 and 256×256 pixels. The applications are run for about 200 million cycles. The progress of the evaluated aging mechanisms is extrapolated from the results obtained for this interval.

8.5.3 Implementation

The NFP evaluation framework consists of four main components:

- the system level simulator (TLM 2.0)
- the RTL state prediction (SystemC)
- the parallel, gate level DUA simulator for observables acquisition (Java)
- the piecewise aging evaluator (Java)

The system level simulator executes the system model at transaction level. During simulation, the high level DUA stimuli in form of time-annotated transaction payloads are captured (cf. section 5.3). The high level workload is used as stimuli for DUA simulation at RTL. The RTL simulator runs in parallel to the high level system simulator. The fast-forwarding is used to improve the performance (cf. section 5.4.1). The RTL state prediction component generates the logic states of the DUA and the information about autonomous cycles (i.e., the length, the depth and the number of idle loops). The generated stimuli is passed over to the gate level DUA simulator for the acquisition of observables.

The gate level DUA simulator is implemented in Java. It contains a gate level model of the DUA and performs parallel, gate level logic simulation. An instance of the gate level simulator is a 64-slot pattern-parallel simulator. Each simulation job received from the RTL simulator is allocated to one of the free slots. If all the slots are occupied, the simulation jobs are queued, or the RTL model starts a new instance of the gate level simulator on the same or a different machine and sends the remaining transactions to a new server (cf. section 5.4.2).

During the gate level DUA simulation, the NBTI and HCI observables are monitored cycle-accurately (cf. section 8.4.1). At the end of each evaluation window, the simulator collects the observables from all the simulation slots and sends them to the aging evaluator.

Fig. 8.5 shows the structure of the aging evaluator. It comprises models for temperature distribution, power dissipation as well as NBTI and HCI aging.

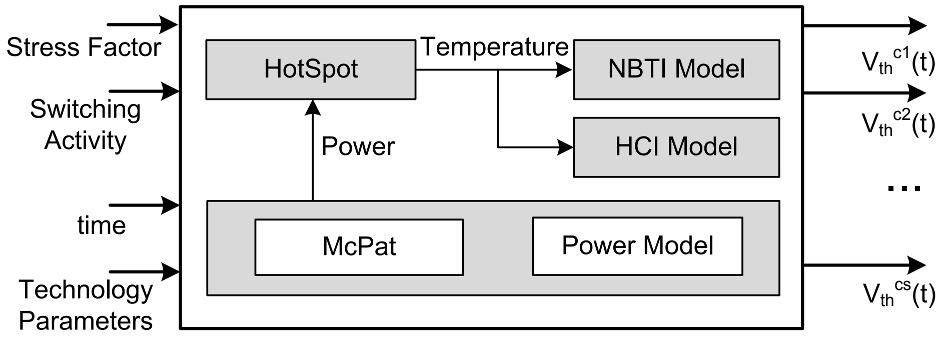


FIGURE 8.5: Block diagram of the NFP evaluator

Temperature profiles are generated with HotSpot[6] [Huan06], which is an accurate, open source tool for die temperature estimation. HotSpot is based on the model presented in 8.3.2.1. It is built on an equivalent circuit of thermal resistances and capacitances that correspond to microarchitecture blocks and essential aspects of the thermal package. It receives the dimensions of the chip and the constituent blocks and also the power dissipation of each block at the sampling point and outputs the temperature.

[6]`http://lava.cs.virginia.edu/HotSpot/index.htm`

HotSpot provides two degrees of accuracy. In the first method known as *block model*, the temperature can be calculated for each block, separately. The block model is fast, but less accurate. *Grid model* offers the choice of another more accurate, but relatively slower temperature estimation.

Using the both HotSpot models for the case study DUAs and comparing the results, we found that the maximum temperature difference in both DUAs is at most $5°C$. This temperature inaccuracy leads to total performance violation (in terms of transistor delay) of less than 1%, which is negligible. Hence, to obtain more speed up, the block model is used here for aging evaluation purposes. We assume that all transistors within a block have the same temperature and hence temperature fluctuations within the DUAs are neglected.

The power dissipated by the SoC at the time interval $(t_1,t_2]$ is considered as the sum of the energy required by the DUA and the rest of the system, mostly the microprocessor in that time interval. Power is inherently a structural property [Zhon06], so the DUA power dissipation for each window is calculated using the fine-grain model from section 6.2.2.1. The power dissipated by the microprocessor and other cores are calculated using McPat[7] [Li09]—an open source, architecture level power estimation tool.

McPat (Multicore Power, Area, and Timing) is an integrated power and area modeling framework for multithreaded, multicore, and manycore architectures. It models dynamic, short-circuit, and leakage power for the target architecture and supports comprehensive early stage design space exploration for multicore and manycore processor configurations ranging from 90nm to 22nm and beyond. Beside power, McPat also estimates the chip area which is required by HotSpot.

The NFP evaluator implements the HCI and NBTI aging models presented in Section 8.3.1. At the end of each evaluation window, the total power estimated for that window along with the area is sent to HotSpot to calculate the average temperature in that window. In the meanwhile, the average observables (NBTI and HCI stress factors) are obtained from the gate level simulator. Based on this data, for each transistor in the DUA, the aging evaluator calculates the change in the threshold voltage over the evaluation window.

[7]`http://www.hpl.hp.com/research/mcpat/`

The framework supports concurrent aging evaluation of multiple DUAs: The RT and gate level simulators for separate DUAs run in parallel and can be distributed across different machines. To accelerate the evaluation of the models, the aging evaluator can also be split into multiple jobs handling different mechanisms and/or different DUAs.

The aging Evaluator communicates with the aging analyzer using TCP/IP protocol. The mutual communication between the aging analyzer, the system level simulator and the RTL state predictor is also done using TCP/IP protocol. The NFP Evaluator works as an independent server, so, it can be run on a different machine. This feature facilitates the server-parallel simulation strategy.

8.5.4 Validation Experiments

To verify the multi-level transaction/gate level modeling accuracy, the system is modeled at RT/gate level and the simulation results are compared. The system model is simulated using a commercial tool. The bus traffic at the interface to the IDCT accelerator and the FPU is captured in form of a value change dump (VCD) file. The structural gate level model of the DUA (either the IDCT accelerator or the FPU) is then simulated in the Java-based aging simulator using the VCD file as stimulus.

The applications run in the validation experiments are equivalent to those analyzed with the proposed method (the processor runs the very same software). In the proposed method, the system is modeled at transaction level and the fast-forwarding method is used to improve the gate level simulation performance. For the validation, a cycle-accurate RTL system model is used, and the gate level simulation is run cycle-accurately without fast-forwarding. The validation experiment's output is the low level observables.

For transistor c, the reference observable, $O_c(t)$, is acquired from the validation experiment while its estimation, $\hat{O}_c(t)$, is provided by the multi-level simulation approach. The absolute estimation error is:

$$e_O^c(t) = |O_c(t) - \hat{O}_c(t)|$$

The accuracy of the proposed method is measured as a mean error over all constituent transistors $c \in \mathbb{C}$ at the end of simulation time $t = T$:

$$Error = \frac{1}{T \cdot |\mathbb{C}|} \sum_{c \in \text{DUA}} e^c_O(T)$$

To verify the aging estimation accuracy, the system is simulated for hundreds of millions cycles. We evaluate the aging-induced threshold voltage degradation for each transistor in the DUA. The accuracy and performance of the proposed method is compared to the results from the accurate, reference simulation. At the end of each evaluation window t_i, the threshold voltage degradation of each transistor $c \in \text{DUA}$ obtained with the proposed method, denoted by $\Delta\hat{V}^c_{th}(t_i)$, is compared to the value obtained in the reference simulation, denoted by $\Delta V^c_{th}(t_i)$. The absolute estimation error for transistor c is defined as:

$$e^c_{V_{th}}(t_i) = |\Delta\hat{V}^c_{th}(t_i) - \Delta V^c_{th}(t_i)|$$

The accuracy of the proposed method is measured as the mean square error (MSE) over the threshold voltage degradation for all constituent transistors in the DUA:

$$\text{MSE}(t_i) = \frac{1}{|\text{DUA}|} \sum_{c \in \text{DUA}} (e^c_{V_{th}}(t_i))^2 \tag{8.8}$$

To compare the obtained accuracy with the state-of-the-art approaches based on random stimuli, as in [Noda10], additional *comparative* experiments are conducted. In these experiments, random stimuli are applied to the DUAs and the average observables are evaluated. These approximated observables are fed to the aging models to calculate the threshold voltage degradation of each transistor in the design. The output is compared with the reference simulation which considers the actual system workload (application).

Since a thorough, cycle-accurate NFP evaluation is very time consuming and requires huge amounts of storage, we use the proposed method with the window size of 10 clock cycles instead of cycle-accurate as reference. Our experiments show that the resulting error is below 0.5×10^{-5}%/day wit respect to cycle-accurate evaluation.

8.5.5 Experimental Results

This section verifies the performance of the piecewise aging evaluation by showing the experimental results for a set of selected experiments. A complete collection of experimental results can be found in Appendix A.

Section 8.5.5.1 presents the aging prediction performance and accuracy gained by the piecewise evaluation approach. It shows that the piecewise evaluation brings considerable speed up with marginal loss of accuracy. Section 8.5.5.2 discusses the accuracy of the multi-level, window-based simulation.

8.5.5.1 Evaluation Accuracy vs. Speed Up

Table 8.1 shows the efficiency and accuracy of the piecewise evaluation method for the JPEG application running on the FPU core. Table 8.2 presents the results for the same application running on the IDCT core. The first column shows the window size while the next three columns express the required time for the RTL simulation, the gate level (GL) simulation and the piecewise aging evaluation, respectively. The RTL simulation results in a small performance overhead, but enables a huge simulation speed up due to the parallelization of the gate level simulations. The NFP evaluation time dominates the simulation time for small window sizes, but it drops significantly as the window size grows. The presented speed up considers the time for the gate level simulation and the piecewise evaluation. Columns 6 and 7 show the relative error growth for NBTI and HCI mechanisms, respectively.

TABLE 8.1: FPU-based 8x8 pixel image JPEG decoding

Win.	RTL	GL	Evaluation	speed up	Error [%/day]	
size	sim.[s]	sim.[s]	[s]	[x]	NBTI	HCI
10	160.9	11430.8	75286.1	1	≈ 0	≈ 0
10^2	46.7	4471.1	2745.7	12.0	0.001	0.0001
10^3	35.5	1810.6	318.8	40.7	0.003	0.0003
10^4	35.1	1581.0	35.5	53.6	0.004	0.0008
10^5	34.8	1476.6	5.9	58.5	0.005	0.0017
10^6	34.3	1379.1	1.6	62.8	0.007	0.0017

As the window size grows, the simulation time drops significantly while the resulting loss of accuracy is marginal. The NFP evaluation time dominates the simulation time for small window sizes, but it reduces dramatically as the window size grows.

For the IDCT-based JPEG application, as shown in table 8.2, the speed up increases remarkably with the window size. The piecewise evaluation with window size of 100 cycles is 4.7x faster while the window size of 10^6 cycles is 185.6x faster than the reference simulation. Moreover, the error grows only slightly as the window size increases.

TABLE 8.2: IDCT-based 8x8 pixel image JPEG decoding

Win.	RTL	GL	Evaluation	speed up	Error [%/day]	
size	sim.[s]	sim.[s]	[s]	[x]	NBTI	HCI
10	22.4	5545.9	14714.5	1	≈ 0	≈ 0
10^2	2.8	2711.8	1572.8	4.7	0.0003	0.0002
10^3	0.5	264.1	159.0	47.9	0.0011	0.0002
10^4	0.3	46.5	138.9	109.3	0.0011	0.0004
10^5	0.3	16.5	107.8	162.9	0.0012	0.0013
10^6	0.2	16.3	92.9	185.6	0.0065	0.0017

In the both experiments, the HCI as well as NBTI prediction error increase very slowly with the window size growth while the evaluation speed up is significant. The error analysis shows that the evaluation error does not exceed 0.007%/day for NBTI and 0.0017%/day for HCI evaluation with 200 million simulation cycles in the worst case. The IDCT based JPEG application gains more speed up compared to the FPU based JPEG application: FPU based applications use the DUA more intensively, as the majority of CPU operations are offloaded to the FPU. For this reason, the performance gain due to simulation fast forwarding is less, compared to the IDCT core.

Fig. 8.6 and 8.7 show the overall speed up vs. the NBTI evaluation error for the FPU and the IDCT core, respectively. The left *y* axis shows the speed up while the right *y* axis represents the error growth in percent per day. The error lines are distinguished from speed up by point symbols. As expected, the speed up increases as the window size grows, but it saturates for large window sizes: As the window size gets larger, all slots of the gate level simulator become occupied. For instance, for the window size of 100 cycles, the overall speed up ranges from 5x to 11x for the FPU, and from 5x to 8x for the IDCT core. For 10^4 cycles, the speed up is from 63x to 487x for the FPU and from 109x to 312x for the IDCT core, depending on the application. Although the speed up for large window sizes is significant, the error rate is only slightly increasing.

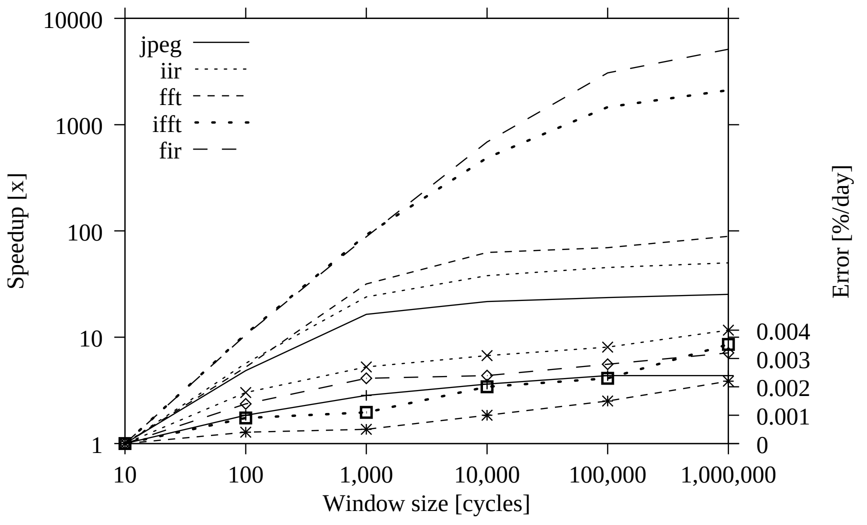


FIGURE 8.6: Overall speed up vs. window size for the FPU core

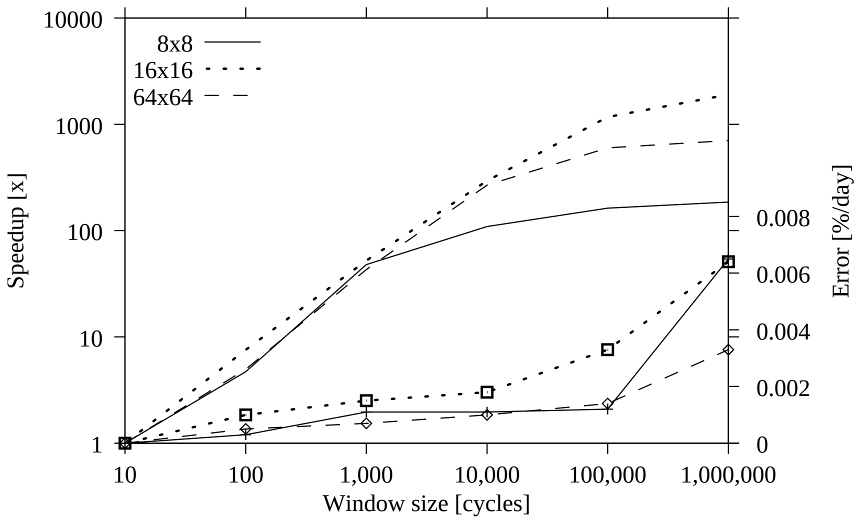


FIGURE 8.7: Overall speed up vs. window size for the IDCT core

In Fig. 8.8, the accuracy of the *comparative* experiments that neglect the application is evaluated for 8x8 JPEG application running on the FPU and IDCT cores and compared against the proposed approach. The piecewise NBTI evaluation error for the FPU core with the window size of 10^6 cycles is 0.007%/day while the comparative experiment results in 0.037%/day. The error of the proposed approach for the IDCT core ranges from 0.0003%/day for window size of 10^2 cycles to 0.0065%/day for 10^6 cycles. The comparative experiments result in 0.0435 %/day error which is significantly higher.

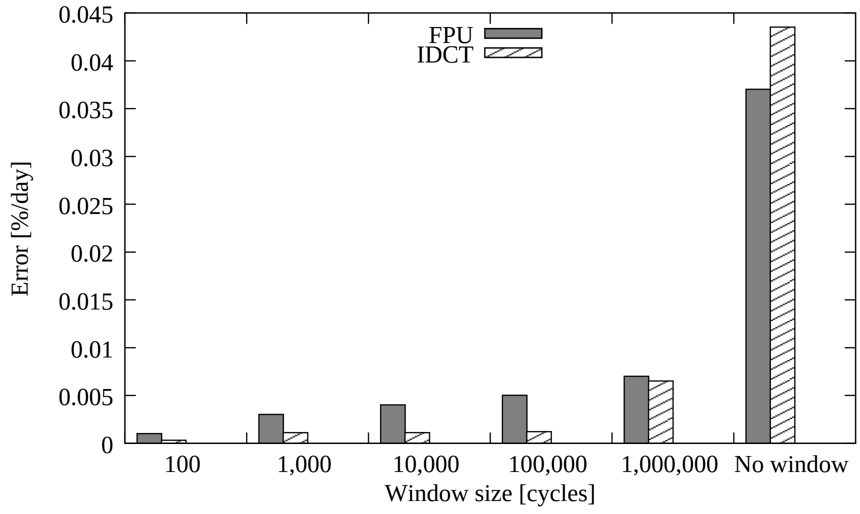


FIGURE 8.8: Relative error with random input patterns for NBTI degradation

8.5.5.2 Modeling Accuracy

This section discusses the accuracy of the window-based, multi-level modeling in detail. To this end, we run a set of experiments to verify the speed up gain vs. accuracy loss for the transaction/gate level modeling. We show that the multi-level modeling is almost as accurate as the cycle accurate models, but accelerates the simulations from 8x up to 431x.

To validate the accuracy of the multi-level modeling, two underlying assumptions are considered:

- The inputs to the DUA are stable, except when the DUA receives a transaction.
- The high level system model accurately predicts the temporal system behavior.

The first assumption holds for many point-to-point communication schemes, but is not valid in general. In a bus-centric system, the DUA bus inputs are exposed to the switching activity generated by other system components. This activity may affect the gates in the combinational transitive fan-out of bus inputs. The incurred inaccuracy is marginal if all the interface inputs are registered or gated, which is a standard design practice [Keat02].

The second assumption is justified as long as the temporal behavior of the high level DUA model matches its actual hardware operation, i.e., the timestamps of the high level DUA workloads are accurate. In general, an exact match cannot be achieved due to the abstract nature of the system model. The window-based simulation approach remedies this effect by averaging the observables in a defined time interval, i.e., the evaluation window.

Based on the following assumptions, the IDCT and FPU cores are evaluated with the aforementioned applications. Table 8.3 shows the results for the IDCT core running several JPEG decompression applications with images of size from 8×8 up to 256×256 pixels.

TABLE 8.3: JPEG decompression: Performance and accuracy

Image size	Application length[cyc.]	Prediction time [s]	Validation time		speed up [x]	Error [%]
			RTL [h]	GL [h]		
8×8	2.67 M	82	0.07	4.6	205	0.002
16×16	3.74 M	83	0.13	9.4	411	0.002
64×64	32.80 M	904	0.75	105.0	421	0.002
256×256	354.00 M	11145	12.20	1323.0	431	0.002

The application lengths in clock cycles are given in the second column. The third column, “prediction time”, gives the time required by the proposed approach to calculate the stress factors. Column 4, “validation time”, gives the time required by the validation experiments for the RTL system simulation, and the DUA gate level aging simulation. The speed up of the proposed method is evaluated with respect to the total validation time at RT and gate level. The proposed prediction method is faster than the validation experiments from 205 up to 431 times.

The mean error is stable at 0.002%. This error results mostly from the inaccuracy of temporal modeling at system level: The high abstraction level results in a slight mismatch of the total application length (and the timing of DUA workload) in the transaction and RT level simulation. However, in case of the examined applications, both the accuracy and performance gain of the proposed method improve as the application length grows.

TABLE 8.4: FPU applications: Performance and accuracy

Image size	Application length[cyc.]	Prediction time [s]	Validation time		speed up [x]	Error [%]
			RTL [h]	GL [h]		
IIR	3.7 M	0.37	0.17	3.4	9.7	0.003
FIR	18.5 M	1.00	0.55	7.7	8.2	0.001
FFT	24.4 M	0.72	0.58	7.6	11.3	0.005
IFFT	217.0 M	5.59	4.59	103.0	19.3	0.006
JPEG	121.0 M	10.63	3.46	77.4	7.6	0.002

The FPU core is evaluated in several signal processing applications, as shown in table 8.4. As discussed before, in presence of the FPU, several CPU operations are turned over to it. Therefore, compared with the previously discussed JPEG applications, the DUA is used more intensively. For this reason, the performance gain due to the simulation fast forwarding is less, compared to the previous results. Nevertheless, the proposed method considerably reduces the effort of the gate level simulation, and achieves a minimal speed up of 7.6x with an average error below 0.006%.

Fig. 8.9 shows a histogram of NBTI stress factor estimation error evaluated for the FPU applications. In all the scenarios, more than 70% of the stress factors are estimated with an error below 1%. The maximum estimation error is below 3%. This error is observed for the transistors that constitute the gates of the transitive combinational fan-in of the Wishbone bus. It results from the assumption that the primary DUA inputs are stable during the autonomous simulation, as discussed before.

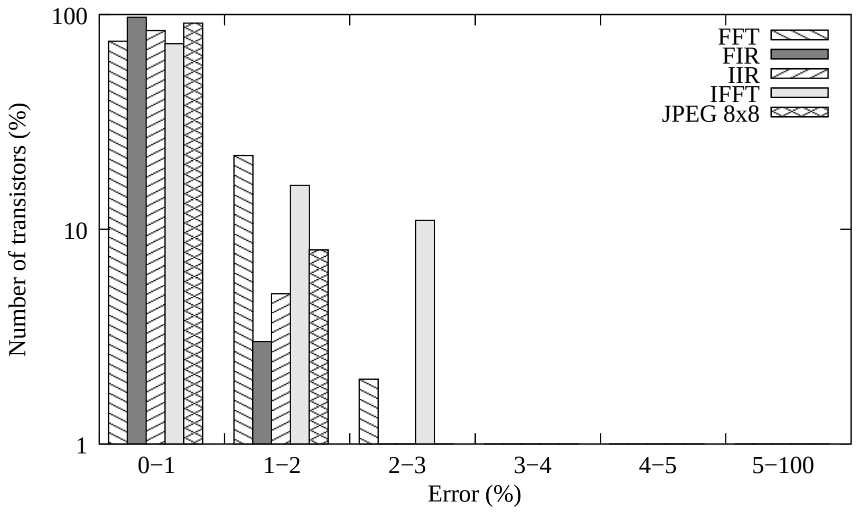


FIGURE 8.9: FPU applications: Error histogram

8.6 Summary

This section applied the piecewise evaluation approach to estimate aging degradation due to NBTI and HCI mechanisms for a typical SoC. The simulation time is split into evaluation windows, during which the average system operating conditions are observed. NFP models are partially linearized with respect to their parameters and evaluated once per window. The accuracy of the NFP evaluation can be adjusted by changing the window size. Experimental results for NBTI and HCI aging prediction show that with a window size of 1 million clock cycles, the achieved speed up ranges from 25x to 35893x, while the average estimation error grows from 0.0022%/day to 0.065%/day for NBTI and 0.0013%/day to 0.0018%/day for HCI aging mechanism.

Part IV

Final Notes

Chapter 9

Concluding Remarks

The main challenge of every semiconductor company is to introduce a new generation of hardware chips in a better quality and a larger quantity in a short time. To compensate the high investment necessary to produce more complex hardware components in newer technologies, a company must be able to develop and produce new products with the fastest rate and the minimum cost [Doer07]. However, product quality and yield start at the design stage and are not simply a manufacturing responsibility. Any potential problem should be discovered and eliminated as close to the beginning of the product cycle as possible [Chia07].

Beside the exponential increase of complexity which poses problems to functional behavior of silicon chips, smaller feature sizes introduce new challenges to non-functional properties: Checking NFPs such as on-chip temperature as well as the impact of the environment on the system, that were ignored or only considered in rare application areas in the past, are becoming more and more focused on during system design because of nanoelectronic effects [Vieh09]; new nano-scale transistors are more prone to variability which manifest as violation of NFPs such as components reliability and performance [Bork05]. On the other hand, rapid move to new technologies leave older technologies immature and prevent the development of novel approaches to deal with the emerging challenges [De99]. As the dimensions of NFPs increase, it becomes more important to understand their impact on the most sensitive design parameters at early design phases. This results in optimized performance and yield which is the final goal of the production.

This work focuses on non-functional properties as an important factor in today's semiconductors. It proposes a fast evaluation approach with adjustable accuracy at early design phases.

An NFP is usually expressed with a model which describes the dependencies and effective parameters. Several parameters change the NFP during the lifetime of the system. The effect of these parameters should be considered during short- and long-term NFP evaluations for accurate NFP prediction. The proposed multi-level simulation methodology is able to predict NFPs at more than one abstraction level, thus providing fast, yet accurate evaluations. The proposed "piecewise evaluation" approach for long-term NFP analysis allows dynamic changes on NFP parameters during the evaluation and at discrete timesteps. Therefore, the effect of dynamic variations on NFPs is reflected in the evaluations.

The accuracy heavily depends on the accuracy of the provided parameters. "Window-based, multi-level simulation" methodology provides an efficient approach to estimate operational parameters accurately considering system workload.

The presented methodologies can be used at early design phases to study the effect of dynamic variations on deciding non-functional properties for different design choices. In addition, they can be used to verify architectural level solutions to mechanisms such as aging.

The authors in [Chan11] show that just using switch level models to verify architecture level aging mitigation techniques result in inaccurate estimations. The multi-level simulation and the piecewise evaluation approaches can be used to verify architectural techniques such as power gating [Cali09, Cali12], dynamic instruction scheduling [Sidd10, Abel07] and lifetime awareness [Srin04b, Srin05] at early design phases using multi-level models.

9.1 Future Directions

The results of this work may provide the foundation for a couple of further research directions.

The piecewise evaluation evaluates NFP models at certain timesteps, called windows. The size of the window is decided considering the required accuracy for the evaluations. However, as the window size is decided, it would remain constant for the length of the simulation. The accuracy of the evaluations can be improved by using adaptable window sizes for the evaluations. After certain timesteps, the accuracy of the evaluations can be verified, and the size of the window can be adjusted to the new value.

The evaluation speed can be improved using GP-GPUs (General Purpose Graphics Processing Units)—specialized manycore processors that provide considerable processing power for delivering high-performance graphics. The piecewise evaluation approach can well be moved to GPU architectures to provide more evaluation speed.

Several approaches in literature aim at proposing on-chip sensors to mitigate aging effects, specially NBTI [Kean10] and HCI [Kim10]. The piecewise evaluation approach can be used in conjunction to these approaches to place the monitor. As the gate model maps transistor characteristics to gate level components, the piecewise evaluation can well find the transistors which may fail due to aging.

Other research directions may include using the piecewise evaluation for aging-aware static timing analysis to recognize critical paths. As the proposed approach is not limited to a particular NFP, it can be used to explore special architectures, such as low-power or high performance architectures.

Bibliography

[Abad03] W. Abadeer and W. Ellis, "Behavior of NBTI Under AC Dynamic Circuit Conditions," in Proc. International Reliability Physics Symposium, pp. 17–22, 2003.

[Abel07] J. Abella, X. Vera, and A. Gonzalez, "Penelope: The NBTI-Aware Processor," in Proc. International Symposium on Microarchitecture (MICRO), pp. 85–96, 2007.

[Abra64] M. Abramowitz and I. Stegun, "Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables,". Applied mathematics series, National Bureau of Standards, 1964.

[Agar08] M. Agarwal, V. Balakrishnan, A. Bhuyan, K. Kim, B. Paul, W. Wang, B. Yang, Y. Cao, and S. Mitra, "Optimized Circuit Failure Prediction for Aging: Practicality and Promise," in Proc. International Test Conference (ITC), pp. 1–10, 2008.

[Agt11] H. Agt, G. Bauhoff, R.-D. Kutsche, and N. Milanovic, "Modeling and Analyzing Non-Functional Properties to Support Software Integration," in Advanced Information Systems Engineering Workshops (C. Salinesi and O. Pastor, editors), vol. 83 of *Lecture Notes in Business Information Processing*, pp. 149–163, Springer Berlin Heidelberg, 2011.

[Alam03] M. Alam, "A Critical Examination of the Mechanics of Dynamic NBTI for PMOSFETs," in Proc. International Electron Devices Meeting, pp. 14.4.1–14.4.4, 2003.

[Alam05] M. Alam and S. Mahapatra, "A Comprehensive Model of PMOS NBTI Degradation," Microelectronics Reliability, vol. 45, pp. 71–81, August 2005.

[Alam07] M. Alam, H. Kufluoglu, D. Varghese, and S. Mahapatra, “A Comprehensive Model for PMOS NBTI Degradation: Recent Progress,” Microelectronics Reliability, vol. 47, pp. 853–862, June 2007.

[Alam11] M. Alam, K. Roy, and C. Augustine, “Reliability- and Process-Variation Aware Design of Integrated Circuits - A Broader Perspective,” in IEEE International Reliability Physics Symposium (IRPS), pp. 4A.1.1–4A.1.11, 2011.

[Anti02] H. Antia, “Numerical Methods for Scientists and Engineers,”. Birkhaeuser Basel, 2002.

[Atki08] K. Atkinson, “An Introduction to Numerical Analysis,”. Wiley India Pvt. Limited, 2nd edition, 2008.

[Ayns08] J. Aynsley, “OSCI TLM-2.0 User Manual,”. Open SystemC Initiative (OSCI), June 2008.

[Baga08] V. Bagad and S. Kawachale, “VLSI Design,”. Technical Publications, 2008.

[Bagr95] R. Bagrodia, Y. an Chen, V. Jha, and N. Sonpar, “Parallel Gate-Level Circuit Simulation on Shared Memory Architectures,” in Proc. Workshop on Parallel and Distributed Simulation (PADS), pp. 170–174, 1995.

[Baha07] R. I. Bahar, D. Hammerstrom, J. Harlow, W. H. J. Jr., C. Lau, D. Marculescu, A. Orailoglu, and M. Pedram, “Architectures for Silicon Nanoelectronics and Beyond,” Computer, vol. 40, pp. 25–33, January 2007.

[Bara11] R. Baranowski, S. Di Carlo, N. Hatami, M. Imhof, M. Kochte, P. Prinetto, H. Wunderlich, and C. Zoellin, “Efficient Multi-Level Fault Simulation of HW/SW Systems for Structural Faults,” SCIENCE CHINA Information Sciences, vol. 54, pp. 1784–1796, September 2011.

[Bark12] M. Barke, M. Kargel, W. Lu, F. Salfelder, L. Hedrich, M. Olbrich, M. Radetzki, and U. Schlichtmann, “Robustness Validation of Integrated Circuits and Systems,” in Proc. Asia Symposium on Quality Electronic Design (ASQED), pp. 145–154, 2012.

[Baue98] J. Bauer, M. Bershteyn, I. Kaplan, and P. Vyedin, “A Reconfigurable Logic Machine for Fast Event-Driven Simulation,” in Proc. Design Automation Conference (DAC), pp. 668–671, 1998.

[Baue10] E. Bauer, “Design for Reliability: Information and Computer-Based Systems,”. Wiley-IEEE Press, 2010.

[Belt09] G. Beltrame, C. Bolchini, and A. Miele, “Multi-Level Fault Modeling for Transaction-Level Specifications,” in Proc. Great Lakes Symposium on VLSI (GLSVLSI), pp. 87–92, 2009.

[Berg09] B. Bergman, J. de Mare, T. Svensson, and S. Loren, editors, “Robust Design Methodology for Reliability: Exploring the Effects of Variation and Uncertainty,”. Wiley, 2009.

[Bern06] J. B. Bernstein, M. Gurfinkel, X. Li, J. Walters, Y. Shapira, and M. Talmor, “Electronic Circuit Reliability Modeling,” Microelectronics Reliability, vol. 46, pp. 1957–1979, December 2006.

[Bhar06] S. Bhardwaj, W. Wang, R. Vattikonda, Y. Cao, and S. Vrudhula, “Predictive Modeling of the NBTI Effect for Reliable Design,” in Proc. Custom Integrated Circuits Conference (CICC), pp. 189–192, 2006.

[Blac67] J. R. Black, “Mass Transport of Aluminum by Momentum Exchange with Conducting Electrons,” in Proc. International Reliability Physics Symposium, pp. 148–159, 1967.

[Blac10] D. Black, J. Donovan, B. Bunton, and A. Keist, “SystemC: From the Ground Up, Second Edition,”. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2010.

[Boli97] A. Boliolo, L. Benini, G. De Micheli, and B. Ricco, “Gate-Level Power and Current Simulation of CMOS Integrated Circuits,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 5, pp. 473–488, December 1997.

[Bona04] A. Bona, V. Zaccaria, and R. Zafalon, “System Level Power Modeling and Simulation of High-End Industrial Network-on-Chip,” in Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings, vol. 3, pp. 318–323, 2004.

[Bork04] S. Borkar, T. Karnik, and V. De, “Design and Reliability Challenges in Nanometer Technologies,” in Proc. Design Automation Conference (DAC), pp. 75–75, 2004.

[Bork05] S. Borkar, “Designing Reliable Systems from Unreliable Components: the Challenges of Transistor Variability and Degradation,” IEEE MICRO, vol. 25, pp. 10–16, November-December 2005.

[Broo00] D. Brooks, V. Tiwari, and M. Martonosi, “Wattch: A Framework for Architectural-Level Power Analysis and Optimizations,” SIGARCH Comput. Archit. News, vol. 28, pp. 83–94, May 2000.

[Brya10] A. Bryant, N. Parker-Allotey, I. R. Swan, D. P. Hamilton, P. Mawby, T. Ueta, T. Nisijima, and K. Hamada, “Validation of a Fast Loss and Temperature Simulation Method for Power Converters,” in Proc. International Conference on Integrated Power Electronics Systems (CIPS), pp. 1–6, 2010.

[Burc93] R. Burch, F. Najm, B. Ping Yang, and T. N. Trick, “A Monte Carlo Approach for Power Estimation,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 1, no. 1, pp. 63–71, 1993.

[Butz10] P. F. Butzen, V. D. Bem, A. I. Reis, and R. P. Ribas, “Transistor Network Restructuring Against NBTI Degradation,” Microelectronics Reliability, vol. 50, pp. 1298–1303, September–November 2010.

[Cai03] L. Cai and D. Gajski, “Transaction Level Modeling: an Overview,” in Proc. International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), pp. 19–24, 2003.

[Cali09] A. Calimera, E. Macii, and M. Poncino, “NBTI-Aware Power Gating for Concurrent Leakage and Aging Optimization,” in Proc. International Symposium on Low Power Electronics and Design, pp. 127–132, 2009.

[Cali12] A. Calimera, E. Macii, and M. Poncino, “Design Techniques for NBTI-Tolerant Power-Gating Architectures,” IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 59, pp. 249–253, March 2012.

[Cano06] J. Cano and D. Rios, “Reliability Forecasting in Complex Hardware/Software Systems,” in Proc. International Conference on Availability, Reliability and Security (ARES), pp. 300–304, 2006.

[Cao11] Y. Cao, “Predictive Technology Model for Robust Nanoelectronic Design,”. Springer US, 2011.

[Chak04] S. Chakravarthi, A. Krishnan, V. Reddy, C. Machala, and S. Krishnan, “A Comprehensive Framework for Predictive Modeling of Negative Bias Temperature Instability,” in Proc. International Reliability Physics Symposium Proceedings, pp. 273–282, 2004.

[Cham95] R. D. Chamberlain, “Parallel Logic Simulation of VLSI Systems,” in Proc. Design Automation Conference (DAC), pp. 139–143, 1995.

[Chan81] K. M. Chandy and J. Misra, “Asynchronous Distributed Simulation via a Sequence of Parallel Computations,” Communications of the ACM - Special issue on simulation modeling and statistical computing, vol. 24, pp. 198–206, April 1981.

[Chan11] T.-B. Chan, J. Sartori, P. Gupta, and R. Kumar, “On the Efficacy of NBTI Mitigation Techniques,” in Proc. Design, Automation and Test in Europe Conference (DATE), pp. 1–6, 2011.

[Chee06] M. Cheema and O. Hammami, “Introducing Energy and Area Estimation in HW/SW Design Flow Based on Transaction Level Modeling,” in Proc. International Conference on Microelectronics (ICM), pp. 182–185, 2006.

[Chen03] S. Cheng, “Partial Difference Equations,”. Advances in Discrete Mathematics and Applications, 3, Taylor & Francis, 2003.

[Chen12] J. Chen, S. Wang, and M. Tehranipoor, “Efficient Selection and Analysis of Critical-Reliability Paths and Gates,” in Proc. the Great Lakes Symposium on VLSI (GLSVLSI), pp. 45–50, ACM, 2012.

[Chia07] C. C. Chiang and J. Kawa, “Design for Manufacturability and Yield for Nano-Scale CMOS,”. Integrated Circuits and Systems, Springer Netherlands, 2007. 10.1007/978-1-4020-5188-3.

[Chu06] P. Chu, “RTL Hardware Design Using VHDL: Coding for Efficiency, Portability, and Scalability,”. Wiley, 2006.

[Clau09] A. Clauset, C. R. Shalizi, and M. E. J. Newman, “Power-Law Distributions in Empirical Data,” SIAM Rev., vol. 51, pp. 661–703, November 2009.

[Davi93] A. Davis, “Software Requirements: Objects, Functions and States (Revised Edition),”. Prentice Hall, 1993.

[De99] V. De and S. Borkar, “Technology and Design Challenges for Low Power and High Performance [Microprocessors],” in In Proc. International Symposium on Low Power Electronics and Design, pp. 163–168, 1999.

[DeBo09] M. DeBole, R. Krishnan, V. Balakrishnan, W. Wang, H. Luo, Y. Wang, Y. Xie, Y. Cao, and N. Vijaykrishnan, “New-Age: A Negative Bias Temperature Instability-Estimation Framework for Microarchitectural Components,” International Journal of Parallel Programming, vol. 37, pp. 417–431, August 2009.

[Dewe90] A. Dewey and S. Director, “Principles of VLSI System Planning:: A Framework for Conceptual Design,”. The Kluwer international series in engineering and computer science: VLSI, computer architecture and digital signal processing, Springer, 1990.

[Dige97] G. Digele, S. Lindenkreuz, and E. Kasper, “Fully Coupled Dynamic Electro-Thermal Simulation,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 5, no. 3, pp. 250–257, 1997.

[DN10] G. Di Natale, M. Doulcier, M.-L. Flottes, and B. Rouzeyre, “Self-Test Techniques for Crypto-Devices,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 18, no. 2, pp. 329–333, 2010.

[Doer07] R. Doering and Y. Nishi, editors, “Handbook of Semiconductor Manufacturing Technology, Second Edition,”. CRC, 2 edition, 2007.

[Donl04] A. Donlin, “Transaction Level Modeling: Flows and Use Models,” in Proc. International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), pp. 75–80, 2004.

[dSF09] A. da Silva Farina and S. Prieto, "On the Use of Dynamic Binary Instrumentation to Perform Faults Injection in Transaction Level Models," in Proc. International Conference on Dependability of Computer Systems, pp. 237–244, 2009.

[Engl08] T. English, K. Man, E. Popovici, and M. Schellekens, "HotSpot : Visualizing Dynamic Power Consumption in RTL Designs," in Proc. East-West Design and Test Symposium (EWDTS), pp. 45–48, 2008.

[Fang98] P. Fang, J. Tao, J. Chen, and C. Hu, "Design in Hot-Carrier Reliability for High Performance Logic Applications," in Proc. Custom Integrated Circuits Conference, pp. 525–531, 1998.

[Fuji90] R. M. Fujimoto, "Parallel Discrete Event Simulation," Communications of the ACM, vol. 33, pp. 30–53, October 1990.

[Fuji99a] R. M. Fujimoto, "Parallel and Distributed Simulation Systems,". New York, NY, USA: John Wiley & Sons, Inc., 1st edition, 1999.

[Fuji99b] R. Fujimoto, "Parallel and Distributed Simulation," in Proc. Simulation Conference, pp. 122–131, 1999.

[Gafn88] A. Gafni, "Rollback Mechanisms for Optimistic Distributed Simulation Systems," in Proc. SCS Multiconference on Distributed Simulation, pp. 61–67, 1988.

[Gai88] S. Gai, P. L. Montessoro, and F. Somenzi, "MOZART: A Concurrent Multilevel Simulator," IEEE Trans. on CAD of Integrated Circuits and Systems, vol. 7, no. 9, pp. 1005–1016, 1988.

[Gajs09] A. S. G. A. S. G. Gajski, D., "Embedded System Design: Modeling, Synthesis, Verification,". Springer, 2009.

[Gers09] A. Gerstlauer, C. Haubelt, A. Pimentel, T. Stefanov, D. Gajski, and J. Teich, "Electronic System-Level Synthesis Methodologies," IEEE Transaction on CAD of Integrated Circuits and Systems, vol. 28, pp. 1517–1530, oct. 2009.

[Ghen05] F. Ghenassia, editor, "Transaction-Level Modeling with SystemC - TLM Concepts and Applications for Embedded Systems,". Springer, 2005.

[Ghos86] S. Ghosh, “On the Concept of Dynamic Multi-Level Simulation,” in Proc. Symposium on Simulation, pp. 201–205, 1986.

[Ghos87] S. Ghosh, “Dynamic Multi-Level Simulation of Digital Hardware Designs,” Simulation, vol. 48, pp. 247–252, June 1987.

[Ghos92] A. Ghosh, S. Devadas, K. Keutzer, and J. White, “Estimation of Average Switching Activity in Combinational and Sequential Circuits,” in Proc. Design Automation Conference (DAC), pp. 253–259, 1992.

[Giel08] G. Gielen, P. De Wit, E. Maricau, J. Loeckx, J. Martin-Martinez, B. Kaczer, G. Groeseneken, R. Rodriguez, and M. Nafria, “Emerging Yield and Reliability Challenges in Nanometer CMOS Technologies,” in Proc. Design, Automation and Test in Europe Conference (DATE), pp. 1322–1327, 2008.

[Glin07] M. Glinz, “On Non-Functional Requirements,” in Proc. International Requirements Engineering Conference (RE), pp. 21–26, 2007.

[Goel89] A. Goel and Y. T. Au-Yeung, “Electromigration in the VLSI Interconnect Metallizations,” in Proc. Midwest Symposium on Circuits and Systems, pp. 821–824, 1989.

[Han11] S. Han, J. Choung, B.-S. Kim, B. H. Lee, H. Choi, and J. Kim, “Statistical Aging Analysis with Process Variation Consideration,” in Proc. International Conference on Computer-Aided Design (ICCAD), pp. 412–419, 2011.

[Hata12] N. Hatami, R. Baranowski, P. Prinetto, and H. Wunderlich, “Efficient System-Level Aging Prediction,” in Proc. European Test Symposium (ETS), pp. 1–6, 2012.

[Heid90] P. Heidelberger and H. S. Stone, “Parallel Trace-Driven Cache Simulation by Time Partitioning,” in Proc. Simulation Conference, pp. 734–737, 1990.

[Hill04] M. Hiller, A. Jhumka, and N. Suri, “EPIC: Profiling the propagation and effect of data errors in software,” IEEE Trans. Computers, vol. 53, no. 5, pp. 512–530, 2004.

[Hsia95] M. S. Hsiao and J. H. Patel, “A New Architectural-Level Fault Simulation Using Propagation Prediction of Grouped Fault-Effects,” in Proc. International Conference on Computer Design (ICCD), pp. 628–635, 1995.

[Huan04] W. Huang, M. R. Stan, K. Skadron, K. Sankaranarayanan, S. Ghosh, and S. Velusam, "Compact Thermal Modeling for Temperature-Aware Design," in Proc. Design Automation Conference (DAC), pp. 878–883, 2004.

[Huan06] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. Stan, "HotSpot: a Compact Thermal Modeling Methodology for Early-Stage VLSI Design," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 14, pp. 501–513, May 2006.

[Huiz90] C. M. Huizer, "Power Dissipation Analysis of CMOS VLSI Circuits by means of Switch-Level Simulation," in Proc. European Solid-State Circuits Conference (ESSCIRC), pp. 61–64, 1990.

[Hwan08] Y. Hwang, S. Abdi, and D. Gajski, "Cycle-Approximate Retargetable Performance Estimation at the Transaction Level," in Proc. Design, Automation and Test in Europe Conference (DATE), pp. 3–8, 2008.

[ITR12] "ITRS. International technology roadmap for semiconductors 2012," 09 2012.

[Jaco99] I. Jacobson, G. Booch, and J. Rumbaugh, "The Unified Software Development Process,". Object technology series, Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[Jeff85] D. R. Jefferson, "Virtual Time," ACM Transactions on Programming Languages and Systems, vol. 7, pp. 404–425, July 1985.

[Jhum05] A. Jhumka, S. Klaus, and S. A. Huss, "A Dependability-Driven System-Level Design Approach for Embedded Systems," in Proc. Design, Automation and Test in Europe Conference (DATE), pp. 372–377, 2005.

[Jian98] Y.-M. Jiang, S.-Y. Huang, K.-T. Cheng, D. Wang, and C.-Y. Ho, "A Hybrid Power Model for RTL Power Estimation," in Proc. Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 551–556, 1998.

[Kang86] S. Kang, "Accurate Simulation of Power Dissipation in VLSI Circuits," IEEE Journal of Solid-State Circuits, vol. 21, no. 5, pp. 889–891, 1986.

[Kawa06] J. Kawa, C. Chiang, and R. Camposano, "EDA Challenges in Nano-scale Technology," in Proc. Custom Integrated Circuits Conference (CICC), pp. 845–851, 2006.

[Kean10] J. Keane, T.-H. Kim, and C. Kim, "An On-Chip NBTI Sensor for Measuring pMOS Threshold Voltage Degradation," IEEE Transaction on VLSI Systems, vol. 18, pp. 947–956, October 2010.

[Keat02] M. Keating and P. Bricaud, "Reuse methodology manual for system-on-a-chip designs,". Springer, 2002.

[Kell08] I. Keller, K. H. Tarn, and V. Kariat, "Challenges in Gate Level Modeling for Delay and SI at 65nm and Below," in Proc. Design Automation Conference (DAC), pp. 468–473, 2008.

[Kies04] T. Kiesling and S. Pohl, "Time-Parallel Simulation with Approximative State Matching," in Proc. Workshop on Parallel and Distributed Simulation (PADS), pp. 195–202, 2004.

[Kim08] T. Kim, R. Persaud, and C. Kim, "Silicon Odometer: An On-Chip Reliability Monitor for Measuring Frequency Degradation of Digital Circuits," IEEE Journal of Solid State Circuits, vol. 43, pp. 874–880, April 2008.

[Kim10] K. K. Kim, W. Wang, and K. Choi, "On-Chip Aging Sensor Circuits for Reliable Nanometer MOSFET Digital Circuits," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 57, no. 10, pp. 798–802, 2010.

[Kim11] D. Kim, M. Ciesielski, K. Shim, and S. Yang, "Temporal Parallel Simulation: A Fast Gate-Level HDL Simulation Using Higher Level Models," in Proc. Design, Automation and Test in Europe Conference (DATE), pp. 1–6, 2011.

[Koch09] M. Kochte, C. Zoellin, M. Imhof, R. Khaligh, M. Radetzki, H.-J. Wunderlich, S. Di Carlo, and P. Prinetto, "Test Exploration and Validation Using Transaction Level Models," in Proc. Design, Automation and Test in Europe Conference (DATE), pp. 1250–1253, 2009.

[Koch10] M. A. Kochte, M. Schaal, H.-J. Wunderlich, and C. G. Zoellin, "Efficient Fault Simulation on Many-Core Processors," in Proc. Design Automation Conference (DAC), pp. 380–385, 2010.

[Koto98] G. Kotonya and I. Sommerville, "Requirements Engineering: Processes and Techniques,". Worldwide Series in Computer Science, John Wiley & Sons, September 1998.

[Kris05] A. Krishnan, C. Chancellor, S. Chakravarthi, P. Nicollian, V. Reddy, A. Varghese, R. Khamankar, and S. Krishnan, "Material Dependence of Hydrogen Diffusion: Implications for NBTI Degradation," in Proc. International Electron Devices Meeting (IEDM) Technical Digest, pp. 687–691, 2005.

[Kwon99] A.-C. Kwong, B. Cockburn, and D. Elliott, "Dynamic Combined Pattern-Parallel and Fault-Parallel Fault Simulation on Computational RAM," in Proc. Canadian Conference on Electrical and Computer Engineering, pp. 438–445, 1999.

[Laid84] K. J. Laidler, "The Development of the Arrhenius Equation," Journal of Chemical Education, vol. 61, no. 6, p. 494, 1984.

[Lala01] P. K. Lala, "Self-Checking and Fault-Tolerant Digital Design,". Morgan Kaufmann, 2001.

[Law07] A. Law, "Simulation Modeling and Analysis,". McGraw-Hill Series in Industrial Engineering and Management Science, McGraw-Hill Education, 2007.

[Lee92] H. K. Lee and D. S. Ha, "HOPE: An Efficient Parallel Fault Simulator for Synchronous Sequential Circuits," in Proc. Design Automation Conference (DAC), pp. 336–340, 1992.

[Lent99] K. Lentz, J. Heller, and P. Montessoro, "System Verification Using Multilevel Concurrent Simulation," IEEE Micro, vol. 19, pp. 60–67, jan/feb 1999.

[Lent00] K. P. Lentz and J. B. Homer, "Handling Behavioral Components in Multi-Level Concurrent Fault Simulation," in Proc. Annual Simulation Symposium (SS), pp. 149–156, 2000.

[Leve03] R. Leveugle and K. Hadjiat, “Multi-Level Fault Injections in VHDL Descriptions: Alternative Approaches and Experiments,” Journal of Electronic Testing, vol. 19, no. 5, pp. 559–575, 2003.

[Leve04] R. Leveugle, D. Cimonnet, and A. Ammari, “System-Level Dependability Analysis with RT-Level Fault Injection Accuracy,” in Proc. International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT), pp. 451–458, 2004.

[LG08] A. Leon-Garcia, “Probability, Statistics, and Random Processes for Electrical Engineering,”. Pearson Education, Limited, 2008.

[Li09] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, “McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures,” in Proc. International Symposium on Microarchitecture (MICRO), pp. 469–480, 2009.

[Liao03] W. Liao, F. Li, and L. He, “Microarchitecture Level Power and Thermal Simulation Considering Temperature Dependent Leakage Model,” in Proc. International symposium on Low power electronics and design, pp. 211–216, 2003.

[Liao05] W. Liao, L. He, and K. Lepak, “Temperature and Supply Voltage Aware Performance and Power Modeling at Microarchitecture Level,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 24, no. 7, pp. 1042–1053, 2005.

[Lin91] Y.-B. Lin and E. D. Lazowska, “A Time-Division Algorithm for Parallel Simulation,” ACM Transactions on Modeling and Computer Simulation (TOMACS), vol. 1, pp. 73–83, January 1991.

[Lore09] D. Lorenz, G. Georgakos, and U. Schlichtmann, “Aging Analysis of Circuit Timing Considering NBTI and HCI,” in Proc. International On-Line Testing Symposium (IOLTS), pp. 3–8, 2009.

[Lore10] D. Lorenz, M. Barke, and U. Schlichtmann, “Aging Analysis at Gate and Macro Cell Level,” in Proc. International Conference on Computer-Aided Design (ICCAD), pp. 77–84, 2010.

[Lore12] D. Lorenz, M. Barke, and U. Schlichtmann, “Efficiently Analyzing the Impact of Aging Effects on Large Integrated Circuits,” Microelectronics Reliability, vol. 52, pp. 1546–1552, August 2012.

[Maci04] E. Macii, “RTL power Estimation and Optimization,” in Proc. Integrated Circuits and Systems Design (SBCCI), p. 1, 2004.

[Makr98] Y. Makris and A. Orailoglu, “DFT Guidance Through RTL Test Justification and Propagation Analysis,” in Proc. International Test Conference (ITC), pp. 668–677, 1998.

[Manz11] M. Manzano, E. Calle, and D. Harle, “Quantitative and Qualitative Network Robustness Analysis under Different Multiple Failure Scenarios,” in Proc. International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), pp. 1–7, 2011.

[Mart07] G. Martin, B. Bailey, and A. Piziali, “ESL Design and Verification: A Prescription for Electronic System Level Methodology,”. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007.

[Mass04] J. Massey, I. Microeletronics, and V. Essex Junction, “NBTI: What We Know and What We Need to Know-A Tutorial Addressing the Current Understanding and Challenges for the Future,” in IEEE International Integrated Reliability Workshop Final Report, pp. 199–211, 2004.

[McPh06] J. McPherson, “Reliability Challenges for 45nm and Beyond,” in ACM/IEEE Design Automation Conference (DAC), pp. 176–181, 2006.

[Merr11] M. Merrett, P. Asenov, Y. Wang, M. Zwolinski, D. Reid, C. Millar, S. Roy, Z. Liu, S. Furber, and A. Asenov, “Modelling Circuit Performance Variations Due to Statistical Variability: Monte Carlo Static Timing Analysis,” in Proc. Design, Automation and Test in Europe Conference (DATE), pp. 1–4, 2011.

[Meye93] W. Meyer and R. Camposano, “Fast Hierarchical Multi-Level Fault Simulation of Sequential Circuits with Switch-Level Accuracy,” in Proc. Design Automation Conference (DAC), pp. 515–519, 1993.

[Meye95] W. Meyer and R. Camposano, “Active Timing Multilevel Fault-Simulation with Switch-Level Accuracy,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 14, pp. 1241–1256, August 1995.

[Mise08] S. Misera, H. T. Vierhaus, and A. Sieber, “Simulated Fault Injections and their Acceleration in SystemC,” Microprocessors and Microsystems - Embedded Hardware Design, vol. 32, no. 5-6, pp. 270–278, 2008.

[Misr86] J. Misra, “Distributed Discrete-Event Simulation,” ACM Comput. Surv., vol. 18, pp. 39–65, March 1986.

[Mons01] F. Monsieur, E. Vincent, D. Roy, S. Bruyre, G. Pananakakis, and G. Ghibaudo, “Time to Breakdown and Voltage to Breakdown Modeling for Ultra-Thin Oxides (Tox<32 Å),” in Proc. International Integrated Reliability Workshop Final Report, pp. 20–25, 2001.

[Moor69] G. Moore, “Trends in Silicon Device Technology,” IEEE Transactions on Electron Devices, vol. 16, no. 2, p. 234, 1969.

[Mukh03] S. Mukherjee, C. Weaver, J. Emer, S. Reinhardt, and T. Austin, “A Systematic Methodology to Compute the Architectural Vulnerability Factors for a High-Performance Microprocessor,” in Proc. International Symposium on Microarchitecture (MICRO), pp. 29–40, 2003.

[Mukh05] S. S. Mukherjee, J. S. Emer, and S. K. Reinhardt, “The Soft Error Problem: An Architectural Perspective,” in Proc. International Conference on High-Performance Computer Architecture (HPCA), pp. 243–247, 2005.

[Mukh11] S. Mukherjee, “Architecture Design for Soft Errors,”. Elsevier Science, 2011.

[Na13] J. Na, “A Novel Simulation Fault Injection using Electronic Systems Level Simulation Models,” IEEE Design and Test of Computers, vol. PP, March 2013.

[Najm94] F. Najm, “A Survey of Power Estimation Techniques in VLSI Circuits,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 2, pp. 446–455, December 1994.

[Nass04] S. Nassif, “The Impact of Variability on Power,” in Proc. International Symposium on Low Power Electronics and Design (ISLPED), p. 350, 2004.

[Nava04] Z. Navabi, S. Mirkhani, M. Lavasani, and F. Lombardi, “Using RT Level Component Descriptions for Single Stuck-at Hierarchical Fault Simulation,” Journal of Electronic Testing, vol. 20, no. 6, pp. 575–589, 2004.

[Nico94] D. Nicol and R. Fujimoto, “Parallel Simulation Today,” Annals of Operations Research, vol. 53, pp. 249–285, 1994. 10.1007/BF02136831.

[Nico96] D. M. Nicol, “Principles of Conservative Parallel Simulation,” in Proc. Conference on Winter Simulation, pp. 128–135, 1996.

[Nico12] G. Nicolescu, I. O’Connor, and C. Piguet, editors, “Design Technology for Heterogeneous Embedded Systems,”. Springer Netherlands, 2012.

[Noda10] M. Noda, S. Kajihara, Y. Sato, K. Miyase, X. Wen, and Y. Miura, “On Estimation of NBTI-Induced Delay Degradation,” in Proc. European Test Symposium (ETS), pp. 107–111, 2010.

[Obor12] F. Oboril and M. B. Tahoori, “ExtraTime: Modeling and Analysis of Wearout Due to Transistor Aging at Microarchitecture-Level,” in Proc. International Conference on Dependable Systems and Networks (DSN), pp. 1–12, 2012.

[Open08] Open SystemC Initiative (OSCI) TLM Working Group, “Transaction Level Modeling Standard 2 (OSCI TLM 2),” June 2008. www.systemc.org.

[Orsh02] M. Orshansky, L. Milor, P. Chen, K. Keutzer, and C. Hu, “Impact of Spatial Intrachip Gate Length Variability on the Performance of High-Speed Digital Circuits,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 21, no. 5, pp. 544–553, 2002.

[Orsh10] M. Orshansky, S. Nassif, and D. Boning, “Design for Manufacturability and Statistical Design: A Constructive Approach,”. Springer Publishing Company, Incorporated, 1st edition, 2010.

[Paci06] G. Paci, P. Marchal, F. Poletti, and L. Benini, "Exploring Temperature-Aware Design in Low-Power MPSoCs," in Proc. Design, Automation and Test in Europe Conference (DATE), pp. 1–6, 2006.

[Paul02] P. Paulin, C. Pilkington, and E. Bensoudane, "StepNP: A System-Level Exploration Platform for Network Processors," IEEE Trans. Design & Test of Computers, vol. 19, pp. 17–26, December 2002.

[Paul05] B. Paul, K. Kang, H. Kufluoglu, M. Alam, and K. Roy, "Impact of NBTI on the Temporal Performance Degradation of Digital Circuits," IEEE Electron Device Letters, vol. 26, no. 8, pp. 560 – 562, 2005.

[Pedr06] M. Pedram and S. Nazarian, "Thermal Modeling, Analysis, and Management in VLSI Circuits: Principles and Methods," Proceedings of the IEEE, vol. 94, pp. 1487–1501, August 2006.

[Poly03] A. Polyanin and V. Zaitsev, "Handbook of Nonlinear Partial Differential Equations,". Handbooks of mathematical equations, Taylor & Francis, second edition, 2003.

[Prad96] D. Pradhan, "Fault-Tolerant Computer System Design,". Fault-Tolerant Computer Systems, Prentice Hall, 1996.

[Rade08] M. Radetzki and R. S. Khaligh, "Accuracy-Adaptive Simulation of Transaction Level Models," in Proc. Design, Automation and Test in Europe Conference (DATE), pp. 788–791, 2008.

[Rajs00] R. Rajsuman, "System-on-a-Chip: Design and Test,". Norwood, MA, USA: Artech House, Inc., 1st edition, 2000.

[Rang03] S. Rangan, N. Mielke, and E. Yeh, "Universal Recovery Behavior of Negative Bias Temperature Instability [PMOSFETs]," in Proc. International Electron Devices Meeting (IEDM), pp. 14.3.1–14.3.4, 2003.

[Ravi03] S. Ravi, A. Raghunathan, and S. Chakradhar, "Efficient RTL Power Estimation for Large Designs," in Proc. International Conference on VLSI Design, pp. 431–439, 2003.

[Reth11] S. Rethinagiri, R. Atitallah, and J. Dekeyser, “A System Level Power Consumption Estimation for MPSoC,” in Proc. International Symposium on System on Chip (SoC), pp. 56–61, 2011.

[Rigo11] S. Rigo, R. Azevedo, and L. Santos, editors, “Electronic System Level Design,”. Springer, 2011.

[Robe06] S. Robertson and J. Robertson, “Mastering the Requirements Process,”. Pearson Education, 2006.

[Roth04] K. Rothbart, U. Neffe, C. Steger, R. Weiss, E. Rieger, and A. Mühlberger, “High Level Fault Injection for Attack Simulation in Smart Cards,” in Proc. Asian Test Symposium (ATS), pp. 118–121, 2004.

[Roy06] K. Roy, T. Mak, and K. Cheng, “Test consideration for nanometer-scale CMOS circuits,” IEEE Design & Test of Computers, vol. 23, no. 2, pp. 128–136, 2006.

[Russ89] G. Russell and I. Sayers, “Advanced Simulation and Test Methodologies for VLSI Design,”. Springer, 1989.

[Saab90] D. G. Saab, R. B. Mueller-Thuns, D. Blaauw, J. T. Rahmeh, and J. A. Abraham, “Hierarchical Multi-Level Fault Simulation of Large Systems,” Journal of Electronic Testing, vol. 1, pp. 139–149, 1990. 10.1007/BF00137390.

[Saad12] M. Saadatmand and M. Sjödin, “Towards Accurate Monitoring of Extra-Functional Properties in Real-Time Embedded Systems,” in Proc. Asia-Pacific Software Engineering Conference (APSEC), pp. 338–342, 2012.

[Sadr12] M. Sadri, A. Barolini, and L. Benini, “MiMAPT: Adaptive Multi-Resolution Thermal Analysis at RT and Gate Level,” in Proc. International Workshop on Thermal Investigations of ICs and Systems (THERMINIC), pp. 1–6, 2012.

[Salu08] K. Saluja, S. Vijayakumar, W. Sootkaneung, and X. Yang, “NBTI Degradation: A Problem or a Scare?,” in Proc. International Conference on VLSI Design (VLSID), pp. 137–142, 2008.

[Sang12] J. Sanguinetti, “Abstraction and Standardization in Hardware Design,” IEEE Design Test of Computers, vol. 29, pp. 8–13, April 2012.

[Sant99] M. B. Santos and J. P. Teixeira, “Defect-Oriented Mixed-Level Fault Simulation of Digital Systems-on-a-Chip Using HDL,” in Proc. Design, Automation and Test in Europe Conference (DATE), pp. 549–553, 1999.

[Sant03] M. Santos, J. Fernandes, I. Teixeira, and J. Teixeira, “RTL Test Pattern Generation for High Quality Loosely Deterministic BIST,” in Proc. Design, Automation and Test in Europe Conference (DATE), pp. 994–999, 2003.

[Schr03] D. Schroder and J. Babcock, “Negative bias temperature instability: Road to cross in deep submicron silicon semiconductor manufacturing,” Journal of Applied Physics, vol. 94, p. 1, 2003.

[Segu04] J. Segura and F. C. Hawkins, “CMOS Electronics:How It Works, How It Fails,”. Wiley-IEEE Press, 2004.

[Shan02] G. Shankaran, “A Multi Level Modeling Approach with Boundary Condition Import for System, Sub-System or Board Level Thermal Characterization,” in Proc. Symposium Semiconductor Thermal Measurement and Management, pp. 35–41, 2002.

[Shan06] L. Shang and R. Dick, “Thermal Crisis: Challenges and Potential Solutions,” IEEE Potentials,, vol. 25, pp. 31 –35, sept.-oct. 2006.

[Shen12] R. Shen, S. X.-D. Tan, and H. Yu, “Statistical Performance Analysis and Modeling Techniques for Nanometer VLSI Designs,”. Springer US, 2012.

[Shim99] K. Shim, I. K. Oh, S. M. Hong, B. S. Ryu, K. Y. Lee, and T. W. Cho, “A Multi-Level Approach to Low Power MAC Design,” in Proc. Workshop on Signal Processing Systems (SiPS), pp. 723–731, 1999.

[Sidd10] T. Siddiqua and S. Gurumurthi, “A Multi-Level Approach to Reduce the Impact of NBTI on Processor Functional Units,” in Proc. ACM Great Lakes Symposium on VLSI (GLSVLSI), pp. 67–72, 2010.

[Sidd11] T. Siddiqua, S. Gurumurthi, and M. Stan, “Modeling and Analyzing NBTI in the Presence of Process Variation,” in Proc. International Symposium on Quality Electronic Design (ISQED), pp. 1–8, 2011.

[Sieg11] N. Siegmund, M. Rosenmuller, C. Kastner, P. Giarrusso, S. Apel, and S. Kolesnikov, “Scalable Prediction of Non-functional Properties in Software Product Lines,” in Proc. International Software Product Line Conference (SPLC), pp. 160–169, 2011.

[Sieg12] N. Siegmund, “Measuring and Predicting Non-Functional Properties of Customizable Programs,”. PhD thesis, Fakultät für Informatik der Otto-von-Guericke-Universität Magdeburg, 2012.

[Sina01] O. Sinanoglu and A. Orailoglu, “RT-level Fault Simulation Based on Symbolic Propagation,” in Proc. VLSI Test Symposium (VTS), pp. 240–245, 2001.

[Sing06] S. Singh, A. Bansal, M. Meterelliyoz, J. Choi, K. Roy, and J. Murthy, “Compact Thermal Models for Thermally Aware Design of VLSI Circuits,” in Proc. The Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronics Systems (ITHERM), pp. 671–677, 2006.

[Srin04a] J. Srinivasan, S. Adve, P. Bose, and J. Rivers, “The Impact of Technology Scaling on Lifetime Reliability,” in Proc. International Conference on Dependable Systems and Networks (DSN), pp. 177–186, 2004.

[Srin04b] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, “The Case for Lifetime Reliability-Aware Microprocessors,” ACM SIGARCH Computer Architecture News, vol. 32, pp. 276–288, March 2004.

[Srin05] J. Srinivasan, S. Adve, P. Bose, and J. Rivers, “Lifetime Reliability: Toward an Architectural Solution,” IEEE Micro, vol. 25, pp. 70–80, ¡may-June 2005.

[Sriv05] A. Srivastava, D. Sylvester, and D. Blaauw, “Statistical Analysis and Optimization for VLSI: Timing and Power,”. Springer US, April 2005.

[Stef08] D. Stefanovic and M. Kayal, “Structured Analog CMOS Design,”. Springer Publishing Company, 1st edition, 2008.

[Stro06] A. W. Strong, E. Y. Wu, R.-P. Vollertsen, J. Sune, G. L. Rosa, and T. D. Sullivan, “Reliability Wearout Mechanisms in Advanced CMOS Technologies,”. John Wiley & Sons, 2006.

[Suta12] K. Sutaria, J. Velamala, and Y. Cao, “Multi-Level Reliability Simulation for IC Design,” in Proc. International Conference on Solid-State and Integrated Circuit Technology (ICSICT), pp. 1–4, 2012.

[Szek97] V. Szekely, A. Poppe, A. Pahi, A. Csendes, G. Hajas, and M. Rencz, “Electro-Thermal and Logi-Thermal Simulation of VLSI Designs,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 5, no. 3, pp. 258–269, 1997.

[Tayl09] R. N. Taylor, N. Medvidovic, and E. M. Dashofy, “Software Architecture: Foundations, Theory, and Practice,”. Wiley, 1 edition, 2009.

[Tham84] K. Tham, R. Willoner, and D. Wimp, “Functional Design Verification by Multi-Level Simulation,” in Proc. Design Automation Conference (DAC), pp. 473–478, 1984.

[Thom08] D. Thomas and P. Moorby, “The Verilog® Hardware Description Language,”. Springer London, Limited, 2008.

[Tu93] R. Tu, E. Rosenbaum, W. Chan, C. Li, E. Minami, K. Quader, P. Ko, and C. Hu, “Berkeley Reliability Tools-BERT,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 12, pp. 1524–1534, October 1993.

[Turi52] A. M. Turing, “The Chemical Basis of Morphogenesis.,” Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences, vol. 237, no. 641, pp. 37–72, 1952.

[Ulri73] E. Ulrich and T. Baker, “The Concurrent Simulation of Nearly Identical Digital Networks,” in Proc. Workshop on Design Automation, pp. 145–150, 1973.

[Vatt06] R. Vattikonda, W. Wang, and Y. Cao, “Modeling and Minimization of PMOS NBTI Effect for Robust Nanometer Design,” in Proc. Design Automation Conference (DAC), pp. 1047–1052, 2006.

[Velu05] S. Velusamy, W. Huang, J. Lach, M. Stan, and K. Skadron, “Monitoring Temperature in FPGA Based SoCs,” in Proc. International Conference on Computer Design: VLSI in Computers and Processors (ICCD), pp. 634–637, 2005.

[Vieh09] A. Viehl, B. Sander, O. Bringmann, and W. Rosenstiel, “Analysis of Non-functional Properties of MPSoC Designs,” in Languages for Embedded Systems and their Applications (M. Radetzki, editor), Springer, April 2009.

[vL01] A. van Lamsweerde, “Goal-Oriented Requirements Engineering: A Guided Tour,” in Proc. International Symposium on Requirements Engineering, pp. 249–262, 2001.

[vM06] A. von Meier, “Electric Power Systems: A Conceptual Introduction,”. Wiley, 2006.

[VW86] G. Van Wylen and R. Sonntag, “Fundamentals of Classical Thermodynamics,”. Wiley, 1986.

[Wall92] G. Wallace, “The JPEG Still Picture Compression Standard,” IEEE Transactions on Consumer Electronics, vol. 38, pp. 18 –34, February 1992.

[Wang03] Q. Wang and S. Roy, “RTL Power Optimization with Gate-Level Accuracy,” in Proc. International Conference on Computer Aided Design (ICCAD), pp. 39–45, 2003.

[Wang06] L. Wang, C. Wu, and X. Wen, “VLSI Test Principles and Architectures: Design for Testability,”. Systems on Silicon, Elsevier Science, 2006.

[Wang07a] L. Wang, C. Stroud, and N. Touba, “System-on-Chip Test Architectures: Nanometer Design for Testability,”. Morgan Kaufmann, 2007.

[Wang07b] W. Wang, V. Reddy, A. Krishnan, R. Vattikonda, S. Krishnan, and C. Y, “Compact Modeling and Simulation of Circuit Reliability for 65-nm CMOS Technology,” IEEE Transactions on Device and Materials Reliability, vol. 7, pp. 509–517, December 2007.

[Wang07c] W. Wang, Z. Wei, S. Yang, and Y. Cao, “An Efficient Method to Identify Critical Gates Under Circuit Aging,” in Proc. International Conference on Computer-Aided Design (ICCAD), pp. 735–740, 2007.

[Wang07d] W. Wang, S. Yang, S. Bhardwaj, R. Vattikonda, S. Vrudhula, F. Liu, and Y. Cao, “The Impact of NBTI on the Performance of Combinational and Sequential Circuits,” in Proc. Design Automation Conference (DAC), pp. 364–369, 2007.

[Wang07e] Y. Wang, H. Luo, K. He, R. Luo, H. Yang, and Y. Xie, "Temperature-Aware NBTI Modeling and the Impact of Input Vector Control on Performance Degradation," in Proc. Design, Automation and Test in Europe Conference (DATE), pp. 546–551, 2007.

[Wang09] L. Wang, Y. Chang, and K. Cheng, "Electronic Design Automation: Synthesis, Verification, and Test,". The Morgan Kaufmann Series in Systems on Silicon, Elsevier Science, 2009.

[Wang10] W. Wang, S. Yang, S. Bhardwaj, S. Vrudhula, F. Liu, and Y. Cao, "The Impact of NBTI Effect on Combinational Circuit: Modeling, Simulation, and Analysis," IEEE Transaction on VLSI Systems, vol. 18, pp. 173–183, January 2010.

[Watt04] N. Wattanapongsakorn and S. P. Levitan, "Reliability Optimization Models for Embedded Systems with Multiple Applications," IEEE Transactions on Reliability, vol. 53, no. 3, pp. 406–416, 2004.

[West85] N. H. E. Weste and K. Eshraghian, "Principles of CMOS VLSI Design: A Systems Perspective,". Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1985.

[Wieg09] K. Wiegers, "Software Requirements,". Microsoft Press, 2009.

[Wu00] L. Wu, J. Fang, H. Yonezawa, Y. Kawakami, N. Iwanishi, H. Yan, P. Chen, A. I-Hsien Chen, N. Koike, Y. Okamoto, C.-S. Yeh, and Z. Liu, "GLACIER: A Hot Carrier Gate Level Circuit Characterization and Simulation System for VLSI Design," in Proc. International Symposium on Quality Electronic Design (ISQED), pp. 73–79, 2000.

[Wund09] H. Wunderlich and S. Holst, "Generalized Fault Modeling for Logic Diagnosis," in Proc. Models in Hardware Testing, pp. 133–155, 2009.

[Xueh09] W. Xuehui, Z. Lei, X. Nong, and T. Yuhua, "Time Management in Parallel Discrete Event Simulation," in Proc. International Forum on Information Technology and Applications (IFITA), pp. 209–212, 2009.

[Yaco89] G. Yacoub and W. Ku, "An Accurate Simulation Technique for Short-Circuit Power Dissipation Based on Current Component Isolation," in Proc. International Symposium on Circuits and Systems, pp. 1157–1161, 1989.

[Yin01] Z. Yin, Y. Min, and X. Li, "An Approach to RTL Fault Extraction and Test Generation," in Proc. Asian Test Symposium (ATS), pp. 219–224, 2001.

[Zhan06] S. Zhang, "Integrating Non-Functional Properties to Architecture Specification and Analysis," in Proc. International Conference on Information Technology: New Generations (ITNG), pp. 112–117, 2006.

[Zhan09] Y. Zhang and G. Zhang, "Fast Gate-Level Simulation and Power Analysis for High Performance Microprocessor," in Proc. International Conference on Computer Science Education, pp. 1155–1158, 2009.

[Zhon06] L. Zhong, S. Ravi, A. Raghunathan, and N. Jha, "RTL-Aware Cycle-Accurate Functional Power Estimation," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 25, pp. 2103 –2117, oct. 2006.

[Zhu05] L. Zhu, G. Chen, B. Szymanski, C. Tropper, and T. Zhang, "Parallel Logic Simulation of Million-Gate VLSI Circuits," in Proc. International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 521–524, 2005.

[Zurc68] F. W. Zurcher and B. Randell, "Iterative Multi-Level Modeling - A Methodology for Computer System Design," in Proc. IFIP Congress, pp. 138–142, 1968.

Appendix A

Additional Result Tables

This appendix presents additional results for several applications running on the FPU and IDCT cores.

TABLE A.1: FPU-based IIR application

Win.	RTL	GL	Eval.	speed-up [x]			Error [%/day]	
size	sim.[s]	sim.[s]	[s]	GL	Eval.	Overall	NBTI	HCI
10	208.3	7984.9	165793.6	1	1	1	≈ 0	≈ 0
10^2	58.5	2541.5	42926.2	3.1	3.9	3.8	0.0018	0.0017
10^3	10.4	834.0	6152.5	9.6	27.0	24.9	0.0027	0.0026
10^4	9.9	638.0	969.9	12.5	171.0	108.1	0.0031	0.0030
10^5	9.2	609.1	106.5	13.1	1557.2	242.9	0.0032	0.0030
10^6	8.6	577.7	16.9	13.8	9790.0	292.3	0.0390	0.0031

TABLE A.2: FPU-based FIR application

Win.	RTL	GL	Eval.	speed-up [x]			Error [%/day]	
size	sim.[s]	sim.[s]	[s]	GL	Eval.	Overall	NBTI	HCI
10	55.1	1922.6	9072.3	1	1	1	≈ 0	≈ 0
10^2	16.2	331.3	1113.0	5.8	8.1	7.6	0.0005	0.0004
10^3	14.0	185.0	70.8	10.4	128.1	43.0	0.0010	0.0006
10^4	13.5	124.0	12.3	15.5	737.1	80.7	0.0011	0.0008
10^5	12.0	110.5	7.4	17.4	1232.1	93.3	0.0046	0.0008
10^6	11.5	108.1	4.5	18.0	1994.8	97.5	0.0089	0.0009

TABLE A.3: FPU-based IFFT application

Win.	RTL	GL	Evaluation	speed-up [x]			Error [%/day]	
size	sim.[s]	sim.[s]	[s]	GL	Eval.	Overall	NBTI	HCI
10	746.9	6998.0	15003.9	1	1	1	≈ 0	≈ 0
10^2	91.8	813.0	1770.5	7.5	8.5	8.2	0.0014	0.0009
10^3	21.5	487.2	192.5	12.5	78.0	31.0	0.0015	0.0012
10^4	10.8	393.1	24.4	15.5	614.3	50.5	0.0045	0.0013
10^5	8.3	320.7	5.2	19.0	2868.8	64.7	0.0063	0.0013
10^6	5.0	290.2	2.4	21.0	6262.1	72.1	0.0078	0.0014

TABLE A.4: FPU-based FFT application

Win.	RTL	GL	Evaluation	speed-up [x]			Error [%/day]	
size	sim.[s]	sim.[s]	[s]	GL	Eval.	Overall	NBTI	HCI
10	204.4	6861.1	12375.1	1	1	1	≈ 0	≈ 0
10^2	56.3	1940.0	1728.8	3.5	7.1	5.2	0.0004	0.0008
10^3	26.9	445.7	160.2	15.4	77.2	31.7	0.0005	0.0012
10^4	18.9	284.9	21.5	24.1	575.2	62.8	0.0010	0.0012
10^5	19.5	272.8	3.7	25.1	3381.2	69.6	0.0030	0.0014
10^6	19.3	215.3	1.1	31.9	11301.5	88.9	0.0035	0.0015

TABLE A.5: IDCT-based 16x16 pixel image JPEG decoding

Win.	RTL	GL	Evaluation	speed-up [x]			Error [%/day]	
size	sim.[s]	sim.[s]	[s]	GL	Eval.	Overall	NBTI	HCI
10	44.0	5234.7	27634.4	1	1	1	≈ 0	≈ 0
10^2	5.7	790.6	3551.0	6.6	7.8	7.6	0.0007	0.0009
10^3	1.0	193.4	435.8	27.0	63.4	52.2	0.0015	0.0012
10^4	0.5	54.0	57.8	96.9	477.7	293.9	0.0022	0.0012
10^5	0.5	18.4	9.7	284.7	2858.6	1171.5	0.0026	0.0013
10^6	0.4	13.8	3.4	379.5	7991.4	1905.4	0.0061	0.0017

TABLE A.6: FPU-based 64x64 pixel image JPEG decoding

Win.	RTL	GL	Evaluation	speed-up [x]			Error [%/day]	
size	sim.[s]	sim.[s]	[s]	GL	Eval.	Overall	NBTI	HCI
10	568.6	48450.6	136906.0	1	1	1	≈ 0	≈ 0
10^2	67.5	6530.7	39669.0	7.4	3.4	4.0	0.0010	0.0013
10^3	11.4	948.1	7093.6	51.1	19.3	23.0	0.0023	0.0016
10^4	7.1	282.1	386.7	171.8	354.0	277.1	0.0025	0.0021
10^5	6.6	196.5	49.5	246.5	2764.1	753.2	0.0051	0.0038
10^6	6.4	182.6	5.9	265.3	23338.9	983.5	0.0089	0.0064

Appendix B

Curriculum Vitae of the Author

Nadereh Hatami received her Master in Computer Engineering/-Computer Architecture from the University of Tehran, Iran in 2008 before she joined Politecnico di Torino, department of system and computer engineering in Italy to conduct her PhD under the supervision of Prof. Paolo Prinetto in 2009. In December 2010, she joint the University of Stuttgart, Institute of Computer Architecture and Computer Engineering of Prof. Dr. rer. nat. habil. H.-J. Wunderlich to continue her PhD.

During her stay in Stuttgart, she was involved in the research projects "OASIS: Online Failure Prediction for Microelectronic Circuits Using Aging Signatures" and "ROCK: Robust On–Chip Communication" supported by the German Research Foundation (DFG). She supported several courses in Italy as well as Stuttgart including "Hardware Lab", "Advanced Programming", "Digital Logic Design", "Computer Architecture" and "Hardware-Based Fault Tolerance". Her research interests include—but are not limited to—reliability and fault tolerance, high-level hardware simulation and synthesis, and electronic design automation.

Appendix C

Publications of the Author

Hatami, N.; Baranowski, R.; Prinetto, P.; Wunderlich, H., ”Efficient system-level aging prediction,” IEEE European Test Symposium (ETS), 2012, pp.1,6, 28-31 May 2012.

Baranowski, R.; Di Carlo, S.; Hatami, N.; Imhof, M. E.; Kochte, M. A.; Prinetto, P.; Wunderlich, H.-J.; Zoellin, C. G., Efficient Multi-level Fault Simulation of HW/SW Systems for Structural Faults, SCIENCE CHINA Information Sciences, Volume 54, Number 9, pp. 1784-1796.

Kochte, M.A.; Zoellin, C.G.; Baranowski, R.; Imhof, M.E.; Wunderlich, H.; Hatami, N.; Carlo, S.D.; Prinetto, P.; , ”Efficient Simulation of Structural Faults for the Reliability Evaluation at System-Level,” IEEE Asian Test Symposium (ATS), pp.3-8, 1-4 Dec. 2010.

Kochte, M.A.; Zoellin, C.G.; Baranowski, R.; Imhof, M.E.; Wunderlich, H.-J.; Hatami, N.; Di Carlo, S.; Prinetto, P.; , ”System reliability evaluation using concurrent multi-level simulation of structural faults,” IEEE International Test Conference (ITC), pp.1, 2-4 Nov. 2010.

Kochte, M.A.; Zoellin, C.G.; Baranowski, R.; Imhof, M.E.; Wunderlich, H.-J; Hatami, N.; Di Carlo, S.; Prinetto, P.; , ”Effiziente Simulation von strukturellen Fehlern für die Zuverlässigkeitsanalyse auf Systemebene,” 4. GMM/GI/ITG-Fachtagung Zuverlaessigkeit und Entwurf (ZuE’10), Wildbad Kreuth, Germany, Sept. 13-15, 2010, pp.25-32.

Hatami, N.; Indaco, M.; Prinetto, P.; Tiotto, G.; , ”Communication interface synthesis from TLM 2.0 to RTL,” IEEE Design & Test Symposium (EWDTS), pp.222-226, 17-20 Sept. 2010.

Hatami, N.; Prinetto, P.; Trapanese, A.; , ”Hardware design methodology to synthesize communication interfaces from TLM to RTL,” IEEE Automation Quality and Testing Robotics (AQTR), pp.1-5, 28-30 May 2010.

Di Carlo, S.; Hatami, N.; Prinetto, P.; , ”Test infrastructures evaluation at transaction level,” IEEE International Test Conference (ITC), pp.1, 1-6 Nov. 2009.

Di Carlo, S.; Hatami, N.; Prinetto, P.; Savino, A.; , ”System Level Testing via TLM 2.0 Debug Transport Interface,” IEEE Defect and Fault Tolerance in VLSI Systems (DFT), pp.286-294, 7-9 Oct. 2009.

Hatami, N.; Ghofrani, A.; Prinetto, P.; Navabi, Z.; , ”TLM 2.0 simple sockets synthesis to RTL,” Design & Technology of Integrated Systems in Nanoscal Era (DTIS), pp.3-8, 6-9 April 2009.

Hatami, N.; Navabi, Z.; , ”An advanced method for synthesizing TLM2-based interfaces,” Design & Test Symposium (EWDTS), pp.104-108, 9-12 Oct. 2008.

Index