# COMPREHENSIVE VARIATION-AWARE AGING SIMULATOR FOR LOGIC TIMING AND SRAM STABILITY

A Dissertation
Presented to
The Academic Faculty

by

Taizhi Liu

In Partial Fulfillment
of the Requirements for the Degree
Ph.D. in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
May, 2017

# COMPREHENSIVE VARIATION-AWARE AGING SIMULATOR FOR LOGIC TIMING AND SRAM STABILITY

*Dedicated to my loving parents, who guided me to where I am today.*

*And to my wife and my daughter, who made me a better person.*

# ACKNOWLEDGEMENTS

I would like to express my special thanks to my advisor, Professor Linda S. Milor, for her guidance and advice during my Ph.D. study. I would also like to thank Professor Azad Naeemi and Professor Sung Kyu Lim for their helpful suggestions and for their time being on my reading committee. I am also very grateful to Professor Abhijit Chatterjee and Professor Jye-Chyi Lu, both of whom have agreed to serve on my dissertation committee.

I would like to express my thanks to all the lab members, Dr. Fahad Ahmed, Dr. Chang-Chih Chen, Dr. Woongrae Kim Dr. Soonyoung Cha, Kexin Yang, Rui Zhang, Daehyun kim for their collaboration, and valuable comments and feedback.

I would like to extend special thanks to my wife and my best friend, Jing Lu, for her unconditional love and support. I would also like to thank my newly born daughter, Natalie Liu, who made me a dad and a better person. I am also very thankful to my parents, Xinglai Liu and Jinfeng Zhu, who have always been on my side, for their love and encouragement throughout my life.

Last but not least, I would like to thank all the professors, teachers, families, and friends who guided me to become the person that I am today.

# TABLE OF CONTENTS

## LIST OF TABLES

# LIST OF FIGURES

# LIST OF SYMBOLS AND ABBREVIATIONS

| | |
|---|---|
| BTI | Bias Temperature Instability |
| CAD | Computer-Aided Design |
| CRC | Cyclic Redundancy Check |
| ECC | Error Correcting Codes |
| FEOL | Front-End-Of-Line |
| FPGA | Field-Programmable Gate Array |
| GCV | Generalized Cross Validation |
| GOBD | Gate Oxide Breakdown |
| HCI | Hot Carrier Injection |
| L1 | First-Level |
| MARSP | Multivariate Adaptive Regression Splines |
| PDK | Process Design Kit |
| PVT | Process-Voltage-Temperature |
| RISC | Reduced Instruction Set Computing |
| RSM | Response Surface Methodology |
| RV | Random Variable |
| SPICE | Simulation Program with Integrated Circuit Emphasis |
| SRAM | Static Random Access Memories |
| STA | Static Timing Analysis |
| StTA | Statistical Timing Analysis |
| TDDB | Time-Dependent Dielectric Breakdown |

# SUMMARY

The purpose of this research is to develop a framework which analyzes circuit-level reliability and evaluates the lifetimes of complex systems like state-of-art microprocessors. The novelty of the proposed work lies on its statistical timing analyzer and the ability to handle the combined effect of a variety of front-end-of-line (FEOL) wearout mechanisms, while including both the manufacturing process variability and the real-time uncertainties in workload and ambient conditions like operating temperature and IR drops. Overall, the proposed framework presents the correlation between circuit performance (speed) and circuit lifetime, which enables circuit designers to avoid excessive guard-banding, by using a better understood reliability budget to achieve higher performance.

Historically, research work on aging effect is active mainly within the communities of device and reliability physics. The relative lack of design knowledge and aging-aware CAD tools further creates a barrier for managing the impact of device degradation on circuit performance. This work not only bridges the gap between device-level wearout mechanisms and circuit-level performance degradations, but also takes into account the impact of software usage on hardware reliability 0–[14]. The timing analyzer in this work is the first attempt in the literature to achieve a comprehensive process-voltage-temperature-aging-aware statistical timing analysis (StTA) while considering the effect of several FEOL wearout mechanisms (bias temperature instability (BTI), hot carrier injection (HCI), gate oxide breakdown (GOBD)) simultaneously. Because of this contribution, one paper of this research received the Best Paper Award at ESREF 2015 (European Symposium on Reliability of Electron Devices, Failure Physics, and Analysis).

Front-end-of-line (FEOL) wearout mechanisms (BTI, HCI, GOBD) degrade transistor characteristics as a function of stress probability. A FPGA-based emulation platform has been developed to determine the activity and stress-state profiles while the emulated systems are running benchmarks. The activity and stress-state profiles are then used to determine the IR-drop profiles and the thermal profiles of a system. Taking into account the detailed voltage, thermal and electrical stress profiles, the degradations of transistor characteristics for each device within the system are calculated according to the device-level wearout models. More specifically, BTI and HCI are modeled as degradations of transistor threshold voltage while GOBD is modeled as degradations of gate-to-source and gate-to-drain resistance.

Then, a unified gate-delay model is proposed to link device-level degradations and the gate delays. The gate-delay model includes the following parameters: channel length, threshold voltage, GOBD breakdown resistances, supply voltage, temperature, input slew and Pi-model parameters (for the RC load). Among them, the threshold voltage of each transistor combines the effect of process variation, BTI, and HCI, while the GOBD breakdown resistances of each transistor represent the GOBD effect. A method, called multivariate adaptive regression splines (MARSP), is employed to characterize the gate delay as a function of these parameters. MARSP is well-suited for capturing essential nonlinearities and interactions in a high-dimension parameter space.

Based on the unified gate-delay models, a statistical timing engine is developed to estimate the variability of circuit-performance degradation due to the aging effect when PVT variations are present. The proposed timing engine consists of two parts: a block-based analyzer and a path-based analyzer. The block-based analyzer performs PVT-aging-aware critical path extraction, and the path-based analyzer performs accurate circuit-delay estimation of the extracted paths.

Using the statistical timing engine, a framework of circuit-level aging assessment has been constructed. The proposed framework determines the detailed electrical stress profiles, thermal profiles and IR-drop profiles of each device within the system under study. Combining these profiles and device wearout models, the statistical timing engine is applied to characterize microprocessor performance degradation and assess system lifetime. Moreover, the lifetime estimates take into account realistic use scenarios which include active, standby, and sleep modes.

Overall, this work presents cross-layer solutions that enable aging analysis of large complex microprocessors, which run realistic workloads. Additionally, this work addresses challenges that arise to attain accurate aging-aware gate-delay models especially when the dimension of the involved parameters is high. It's the first attempt to have a comprehensive aging simulator which handles the effect of BTI, HCI and GOBD simultaneously while also taking into account the workloads, ambient conditions and manufacturing variability 0–[5]. The framework presented here manages to assess aging of the entire design under realistic usages, and enables designers to tighten excessive guard-banding while still meeting the reliability requirements 0–[13].

# 1. INTRODUCTION

The aggressive scaling of CMOS technology not only brings benefits of speed and power, but also poses challenges to circuit designers because of the ever-increasing manufacturing variability and reliability issues. The challenge is further compounded by the real-time uncertainties in workload and ambient conditions, which dynamically influence the circuit degradation rate. To improve design predictability and guarantee system lifetime, it's essential to have accurate aging simulation tools for reliability.

It is very challenging to accurately characterize circuit performance degradation due to device reliability in the presence of Process-Voltage-Temperature (PVT) variations for a complex system. A common way to deal with this in industry is to add an extra guard-band for aging on top of the worst PVT corners. The guard-band is set by assuming that all the transistors receive the worst-case stress conditions. However, adding excessive guard-banding sacrifices the performance of microprocessors and creates headaches in timing closure.

The purpose of this research is to present a solution to assess the lifetimes of complex systems like state-of-art microprocessors, while taking into account the effect of PVT variations and a variety of Front-end-of-line (FEOL) device wearout mechanisms. The proposed aging simulator achieves accurate statistical timing analysis to study the combined impact of aging effect, manufacturing variability, ambient conditions and realistic workloads 0–[13]. It presents the relationship between circuit performance (speed) and circuit reliability, and gives insights for designers to achieve optimal tradeoff between performance and reliability. Moreover, circuit designers can benefit from the proposed work to avoid excessive guard-banding to achieve higher performance while still maintaining the required reliability.

The device wearout mechanisms studied in this work are Bias Temperature Instability (BTI), Hot Carrier Injection (HCI), and gate oxide breakdown (GOBD). BTI is a wearout mechanism which causes the threshold voltage, Vth, of CMOS transistors to increase over time under voltage stress, resulting in a temporally-dependent degradation of digital circuit delay. HCI also degrades the threshold voltage of the transistors under stress. From the perspective of circuit operation, HCI and BTI stress have different time windows. HCI stresses devices only during the dynamic switching period when current flows through the device, whereas BTI stresses devices as a function of logic state.

GOBD, also time-dependent dielectric breakdown (TDDB), is another reliability concern for CMOS devices. When the gate dielectric layer abruptly loses its insulating properties, it’s called hard breakdown (HBD), and HBD can be detected as a large jump in the current versus time curve. Prior to hard breakdown, GOBD involves soft breakdown (SBD) where the leakage current in the gate dielectric slightly increases with time, while the gate dielectric still retains its insulation property. It’s well understood that GOBD causes gate delay degradation as the transistors gradually weaken before hard breakdown happens. Eventually, the degraded circuit will fail to work when the delay exceeds the clock period.

To take into account the susceptibility of circuit performance to reliability and variations, statistical timing analysis (StTA) is needed to be PVT and aging aware to assure the design meets timing specifications and the reliability requirements before committing a design to manufacture. The objective of StTA is to determine the distribution of the circuit delay with acceptable accuracy and reasonable runtime. In the circuit, each transistor undergoes different workload, different temperature and different voltages, which means that each transistor ages differently. In order to achieve accurate timing analysis, the gate-delay model has to be properly characterized to take into account the effect of different PVT parameters and different aging conditions of transistors in the gate. As circuit integration increases dramatically, the complexity of large circuits make it very challenging to perform accurate PVT-aging-aware timing analysis.

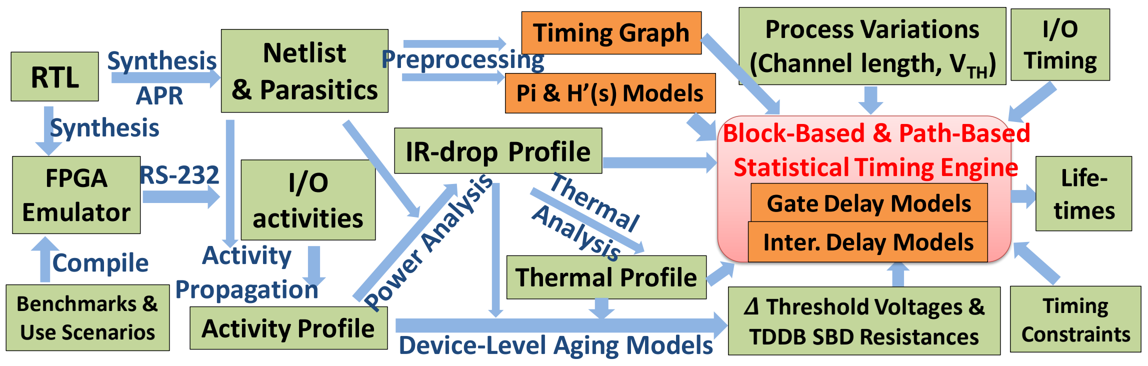


**Figure 1 – The framework of the proposed PVT-aware aging simulator.**

The big picture of the proposed aging simulation framework is shown in Figure 1. Because the wearout mechanisms being studied are activity, supply voltage (VDD) and temperature dependent, the research has utilized FPGA emulation to efficiently acquire electrical, thermal and IR-drop profiles for large systems like microprocessors. Simulating a large system on FPGA emulation platform takes only a few minutes to complete, while running Register-Transfer Level (RTL) simulations to extract the activity profiles of each net might take a few months to simulate a single benchmark.

The RTL is firstly synthesized and loaded to the FPGA with the Xilinx ISE (Integrated Software Environment). Once the FPGA is programmed, the activity can be collected by placing counters at the I/O ports to track the state probabilities and the toggle rates of the ports during application runtime. Since the I/O ports for each unit can be found on the top of each module, the counters are attached to the ports automatically with a scripting language. The activity transportation unit is inserted into the RTL automatically as well. The RTL is also synthesized for layout generation. Using the RC information from the layout, the net activities, and the computed power profile (via a power simulator), the thermal profile throughout the microprocessor is computed using a thermal simulator. The net activities and layout are also used to determine the IR-drop profile throughout the microprocessor.

For process variations, the proposed work includes inter-die, intra-die, and intra-gate variations supporting any distribution and any correlation profile between different parameters. The process variations due to channel length ($\Delta L$) and threshold voltage ($\Delta Vth$) for each transistor are considered. Because BTI and HCI also impact the threshold voltage, the threshold voltage for each transistor is a combination of variation due to the process, BTI and HCI, while $\Delta L$ is only due to process variations.

The degradations of each transistor in the circuit are calculated according to the device-level wearout models and the extracted electrical stress, thermal, IR-drop, and process profiles. As BTI and HCI both cause threshold voltage shifts, it’s straightforward to combine their effect. However, it’s not straightforward to further include GOBD as there is a clear gap between circuit timing and GOBD. This research bridges this gap by using a unified gate-delay model based on Multivariate Adaptive Regression Splines (MARSP) method 0–[3]. MARSP is well suited for capturing essential nonlinearities and interactions in a high-dimension space. The proposed gate-delay model links device-level wearout models and the gate delays. The gate-delay models include many parameters, including PVT and wearout parameters. Among them, the threshold voltage of each transistor combines the effect of process variation, BTI, and HCI, while the GOBD breakdown resistances of each transistor represent the GOBD effect.

The electrical stress, thermal, IR-drop, and process profiles. and RC parasitics, and the degradation profiles, generate all the parameters for each gate which are needed to calculate the gate delays. Based on the unified gate-delay models, a statistical timing engine is developed to estimate the variability of circuit-performance degradation due to the aging effect when PVT variations are present. The proposed timing engine consists of two parts: a block-based timing analyzer and a path-based timing analyzer. The block-based timing analyzer performs PVT aging-aware critical path extraction, and the path-based timing analyzer performs accurate circuit-delay estimation of the extracted paths.

The statistical timing engine proposed in this work is achieved based on the Monte Carlo (MC) method. Monte Carlo StTA sufficiently samples the random parameter domain based on the Metropolis sampling algorithm. For each sample, the circuit delay is computed using traditional static timing analysis (STA) based on gate-delay models. If a sufficient number of samples are drawn, the circuit-delay distribution is obtained. The Monte Carlo approach has the advantage of being completely general, having the ability of handling any kind of parameter distribution, and any correlations between different parameters. Monte Carlo StTA also has the advantage of accuracy, and it is universally used as a reference to validate the accuracy of probabilistic StTA implementations. The disadvantage of MC

StTA is the high runtime cost due to the required sample size.  There have been some existing efforts to reduce the large sample size of MC StTA using reduced sampling techniques, such as quasi-Monte Carlo [15], Latin hypercube [16], importance sampling [17], and the Karhunen-Loeve expansion model [18]. Moreover, the Monte Carlo method intrinsically facilitates parallel computation, like multi-threading. As nowadays a great deal of computational resources is available, many computation-intensive approaches become practical. As it's shown in this thesis, the runtime of the proposed aging-aware StTA framework is very acceptable.

The proposed PVT-aging-aware statistical timing engine performs the analysis of circuit timing degradation. The timing engine generates the delay degradation at a variety of stress times. Lifetimes can then be calculated according to the specified operating frequency. When the circuit delay of the aged circuit exceeds the clock period, the amount of stress time is marked as the chip lifetime.

Overall, this research work has proposed a framework of aging simulation to estimate the lifetimes of complex systems like state-of-art microprocessors while considering the combined effect of process variations, workloads, ambient conditions, and a variety of FEOL wearout mechanisms. The results of this aging simulators can present some insights to circuit designers for them to get the optimal trade-offs between performance and reliability.  This thesis also applies the proposed aging simulator for finding the optimum operating voltage which can achieve the best lifetime for reliability-critical applications

The rest of the thesis is organized as follows. Chapter 2 gives some background knowledge followed by a survey of the related work. Chapter 3 presents the device-level wearout models that are used in this research. Chapter 4 gives the flow of standard cell characterization and how the MARSP method is used in the gate-delay modeling. Chapter 5 utilizes the MARSP-based delay models to construct a Monte-Carlo based Statistical Timing Analyzer which combines the advantages of both block-based method and path-based method. Chapter 6 presents the framework of the proposed aging simulator, and presents the methodology of how to efficiently evaluate performance degradation of a microprocessor due to BTI, HCI and GOBD, simultaneously. In Chapter 7, the proposed aging simulator is extended to study the relationship between performance and reliability of the cache memory for different cache configurations. Chapter 8 concludes this thesis.

# 2. RELATED WORK

This chapter surveys prior related work in three aspects: gate-delay characterizations for standard cells, statistical timing analysis, and aging analysis due to BTI, HCI and GOBD.

## 2.1 Gate-Delay Modeling

There are many existing works on standard cell characterization using both the traditional method [19]–[23] and the emerging current source-based method [24]–[28]. Except for [20],[21], all methods evaluate only the accuracy in characterizing a single cell. And [20],[21] consider only a single benchmark circuit, without determining the accuracy of their method with respect to SPICE.

This research uses a method of standard cell characterization consisting of three models: an input capacitance model for standard cells, a sensitivity model for variational resistive-capacitive loads, gate delay and interconnect delay models via multivariate adaptive regression splines (MARSP). This is the first time that MARSP has been applied to standard cell characterization.

MARSP is an adaptive procedure for multivariate nonparametric regression. That is, it doesn't take a predetermined form, but it constructs the model structure according to the information derived from the data. The MARSP technique is well suited for high-dimensional problems while capturing essential nonlinearities and interactions. Due to its adaptive nature, MARSP can 'filter out' the negligible parameters without manual intervention. For a complex cell containing over 40 devices, the categorization (or clustering) of switching/non-switching devices and on-transition/off-transition/non-transition devices used in [19],[22] is cumbersome. MARSP can reduce this effort by automatically capturing essential parameters and removing negligible parameters through its intelligent process.

## 2.2 Statistical Timing Analysis

Statistical timing Analysis can be categorized into two board classes: Monte Carlo (MC) method and Probabilistic method. Monte Carlo method is based on sample-space enumeration while Probabilistic method is based on statistical operations between random variables (RVs).

Probabilistic StTA models gate delay and arrival times as RVs and propagates arrival time through the circuit via statistical sum and maximum operations. Probabilistic StTA has drawn much attention and research effort due to its runtime advantages. However, it has proven to be challenging to efficiently model skewness in the arrival-time distribution which results from nonlinearity of gate delays and the statistical maximum operation. Much effort [29]–[34] has tried to address these issues.

Monte Carlo StTA sufficiently samples the probability regions based on the Metropolis sampling algorithm [35]. For each sample, the circuit delay is computed using gate-delay models in which canonical first-order or quadratic model is usually used. Although for each sample the computed circuit delay can be either an overestimate or an underestimate, the error in estimating the circuit-delay distribution is acceptable if a sufficient number of samples are drawn. Monte Carlo StTA has the advantage of being completely general. [36] has shown that Monte Carlo StTA is accurate even in scenarios with high dimensionality and non-Normal distributions in the process variation space, where Probabilistic StTA has difficulties.

Monte Carlo based StTA is universally used to validate the accuracy of all practical probabilistic StTA implementations, but never used as a practical StTA method itself. The accuracy of Monte Carlo StTA relies on its large sample size, because the root-mean-square error in the estimate of circuit delay decreases as $O(n^{-0.5})$ where n is the sample size [18]. The high runtime cost due to its required sample size has been the main hurdle preventing Monte Carlo StTA from being practical.

There have been some existing efforts to reduce the large sample size of Monte Carlo StTA using reduced sampling techniques, such as quasi-Monte Carlo [15], Latin hypercube [16], importance sampling [17], and the Karhunen-Loeve expansion model [18]. In this work, it proposes to reduce the sample size and the overall runtime cost by estimating the circuit delay of each sample with significant accuracy.  In other words, the standard deviation of the error for each estimate is a function of the standard deviation of the error of each component, and if we reduce the standard deviation of the errors of the components, it becomes possible to reduce the sample size while achieving the same standard deviation of the error for the system.

The difficulties of extending gate-level characterization to circuit-delay approximation are from two sources. First, the output transition time of each gate propagates to be used as the input transition time of the subsequent gate. The transition-time error is accumulated and magnified stage by stage, which further undermines the accuracy of circuit delay. Second, input capacitances of standard cells are part of the loads of previous stage cells. With process variations present, these input capacitances are variational, which causes the loads of previous stage gates to also be variational. If this variability of loads is neglected, this will cause transition time error and the error will be magnified stage by stage. This work proposes a framework consisting of sophisticated gate-level models to solve these difficulties. The proposed framework has significantly smaller error in estimating circuit delay, with accuracy verified with SPICE. Because of the accuracy of our framework for each sample, only limited samples are needed to get the circuit delay distribution and thus the runtime efficiency is improved significantly.

## 2.3 Aging Analysis of Circuit Timing Due To BTI, HCI and GOBD

Many prior works have studied the impact of BTI on circuit timing [37]–[42]. [37] considers the effect of voltage, temperature and node switching activity, without considering process variations. [38] and [39] propose a framework to study the BTI effect using an iterative scheme to deal with the interdependence between temperature and power profiles, together with BTI degradation. [40] provides a probabilistic method to study the BTI effect using both the Reaction-Diffusion (R-D) and the Trapping-Detrapping (T-D) models. [41] and [42] analyze the impact of Negative Bias Temperature Instability (NBTI) on circuit timing using a Monte Carlo analysis technique and first order models of variation.

HCI stresses devices only during the dynamic switching period when current flows through the device, whereas BTI stresses devices as a function of logic state. The impact of HCI on circuit timing has been studied in several research efforts. [43] studies the impact of the HCI effect on the soft-error rate of small-scale digital circuits. [44] proposes a scalable approach for HCI degradation analysis using the Gaussian model for process variations and a first-order model for timing degradation.

Gate Oxide Breakdown (GOBD) involves soft breakdown (SBD) where the leakage current in the gate dielectric slightly increases with time, while the gate dielectric still retains its insulation property. It's well understood that GOBD causes gate delay degradation as the transistors gradually weaken before hard breakdown happens. GOBD has been widely studied at the transistor-level. However, less study has been focused on the gate level. In [45], an analytical model is presented to predict the delay of logic gates subject to GOBD. [46] analyzes the impact of GOBD on a 41-stage ring oscillator. Unfortunately, only small circuits, like ring oscillators, are studied.

As BTI and HCI both cause threshold voltage shifts, it's straightforward to combine their effect. [47] studied the combined effect of BTI and HCI on circuit timing. However, it's not straightforward to further include GOBD as there is a clear gap between circuit timing and GOBD. This work bridges this gap by using a unified MARSP-based gate-delay model. Therefore, the proposed framework can study the combined aging effect of BTI, HCI and GOBD.

# 3. DEVICE-LEVEL WEAROUT MECHANISMS

## 3.1 Bias Temperature Instability (BTI)

Bias Temperature Instability (BTI) includes Negative Bias Temperature Instability (NBTI) and Positive Bias Temperature Instability (PBTI). NBTI is the degradation of a pMOS device under negative gate stress, and PBTI is the degradation of an nMOS device under positive gate stress. BTI results in shifts in device parameters, such as threshold voltage, transconductance, device mobility, etc., but is generally associated with shifts in the threshold voltage.

The initial distribution of threshold voltages is generally assumed to be Normal. Recent experimental work has shown that the threshold voltage shift ($\Delta V_{th}$) as a function of time under DC stress ($t_{DC}$) is best modeled with trapping/de-trapping theory [48]:

$$\Delta V_{tp/tn}(DC) = \phi(T, E_F)(A + \mathrm{Bln}(t_{DC})) \tag{1}$$

where,$A, B,$ and $\phi$ are constants. $\phi$ is proportional to the number of available traps and is a function of temperature, $T$, and the Fermi level, $E_F$. The temperature dependence is incorporated in $\phi(T, E_F)$. The duty cycle, $\alpha$, impacts the shift and is incorporated as an effective Fermi level [49], where $E_{F,eff} = \alpha E_{F,on} + (1 - \alpha)E_{F,off}$, and $E_{F,on}$ and $E_{F,off}$ are Fermi levels when the device is on and off, respectively. The duty cycle accounts for the percentage of the time that the transistor is under stress, i.e., when the gate terminal of the NMOS device is at a HIGH voltage or the gate terminal of the PMOS device is at a LOW voltage. More specifically, if the duty cycle, $\alpha$, for a transistor is given, then the time under stress, $t_{stress}$, and the recovery time, $t_{rec}$, can be obtained as $t_{stress} = \alpha \cdot t_{total}$,

$t_{rec} = (1-\alpha) \cdot t_{total}$, respectively, where $t_{total}$ is the total time that the circuit is aging. Hence, overall,

$$\Delta V_{tp/tn} = \phi_0 e^{-E_F/kT} g(t_{stress}/(t_{stress}+t_{rec})) \cdot (A + Bln(t_{stress}+t_{rec})) \quad (2)$$

where $\phi_0$ is a constant. The constants were obtained from the experimental results in [50].

## 3.2 Hot Carrier Injection (HCI)

HCI describes the phenomenon by which carriers at a MOSFET's drain gain sufficient energy to be injected into the gate oxide and cause degradation of some device parameters. This occurs as carriers shoot out from the source of a MOSFET, accelerate in the channel, and experience impact ionization near the drain end of the device. Damage can occur at the interface, within the oxide and/or within the sidewall spacer. Interface-state generation and charge trapping induced by this mechanism result in degradation of some MOSFET parameters, such as the threshold voltage, transconductance, channel mobility, and drain saturation current.

Historically, HCI was only a major concern for nMOS devices, with pMOS devices showing comparatively negligible degradation because (a) holes have a smaller impact ionization rate and (b) holes face a higher $Si - SiO_2$ barrier than electrons. However, subsequent reports have revealed that HCI effects on pMOS devices have also been observed [51]. Since hot electrons are generated during logic transitions, the impact of HCI is directly proportional to the switching frequency. In this research, predictive HCI lifetime models under dynamic stress are used for long term performance-degradation simulations. The threshold voltage degradation due to HCI during stress time is modeled as [52]:

$$\Delta V_{tp/tn} = A_{HCI}(r_{trans}t_{stress}t_{trans})^n \tag{3}$$

where $r_{trans}$ is the frequency-dependent transition rate, $t_{stress}$ is the stress time, $t_{trans}$ is the transition time, and $A_{HCI}$ is a technology dependent constant that depends on the inversion charge, the trap generation energy, the hot electron mean free path, and other process-dependent factors. The data used in our study of HCI comes from the experimental data in [53],[54].

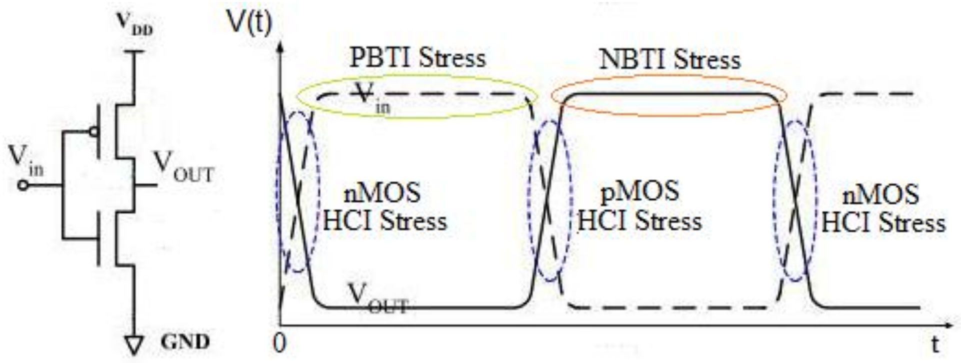


**Figure 2 – Stress-time windows of NBTI, PBTI and HCI for an inverter.**

From the perspective of circuit operation, HCI and BTI stress have different time windows. HCI stresses devices only during the dynamic switching period when current flows through the device, whereas BTI stresses devices as a function of logic state. The stress time windows of BTI and HCI for an inverter circuit are illustrated in Figure 2 as an example.

## 3.3 Gate Oxide Breakdown (GOBD)

GOBD is one of the key reliability issues for CMOS devices. Stress induced leakage current (SILC) is induced by trap-assisted tunneling mechanisms where electrons pass from the cathode to the anode via defect sites (neutral traps) in the gate dielectric by the electrical field [55]–[58]. When the gate dielectric experiences partial breakdown, it is known as soft breakdown (SBD) [55],[59].

Experimental observations indicate that the mean time to failure is a function of the total gate oxide surface area, temperature, and gate voltage due to the weakest-link character of gate dielectric breakdown [60]. However, when abstracting this relationship to the system level, it is important to take into account details of circuit operation, not just the surface area. Moreover, circuits have been known to operate during breakdown [61]. In order to model circuit performance degradation under breakdown, time-dependent resistance models [62],[63] and time-dependent leakage current models [64] have been proposed for SPICE simulation. This research work uses time-dependent resistance models.

Using emulation, described in the next section, the devices are partitioned into groups that experience equivalent stress and temperature. More specifically, for an nMOS device, the time under stress is the time that the gate has the supply voltage applied. This time depends on the input patterns and the propagation of these patterns to each MOSFET. For each group of devices, the next step is to determine the number of devices experiencing different numbers of soft breakdown paths. This is done using the percolation model.

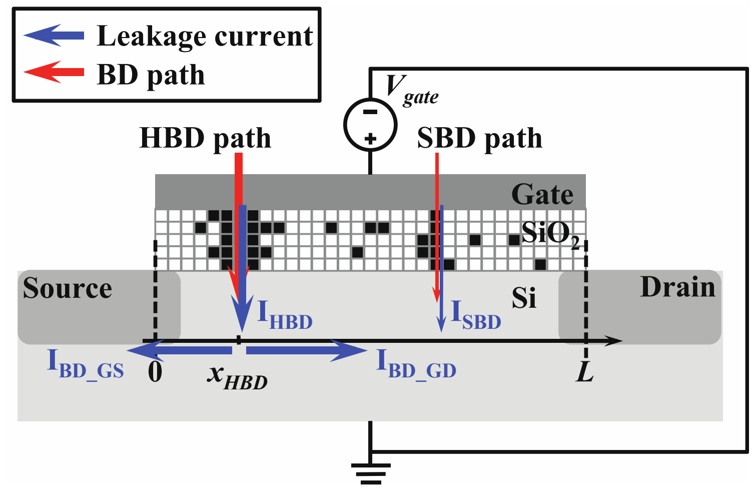


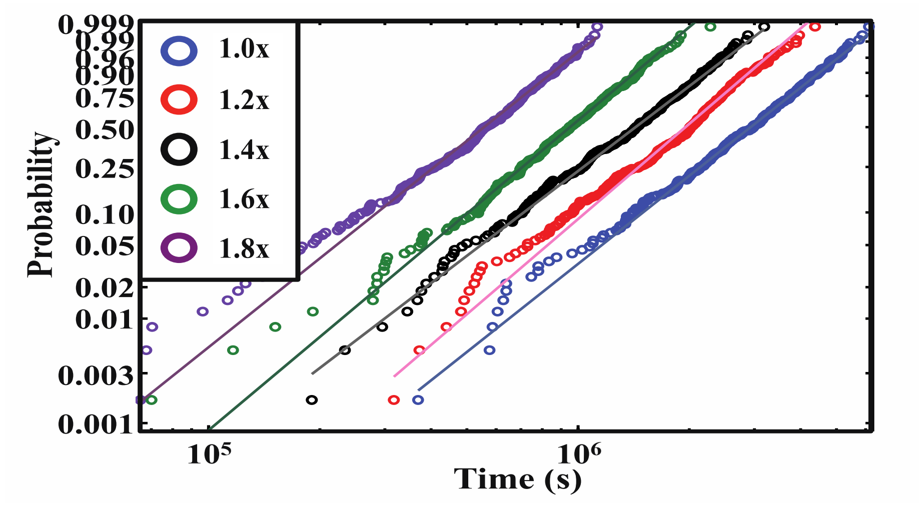
**Figure 3 – Defect generation in the gate dielectric layer based on a 2D percolation model for SBD and HBD paths.**

The percolation model (PM) concept involves placing neutral traps randomly within the gate dieelctric and analyzing the number of resistive conduction paths in a 2D matrix representing the gate dielectric layer [65], as shown in Figure 3. The 2D model have been expanded to 3D to count an accurate number of conduction paths in this research.

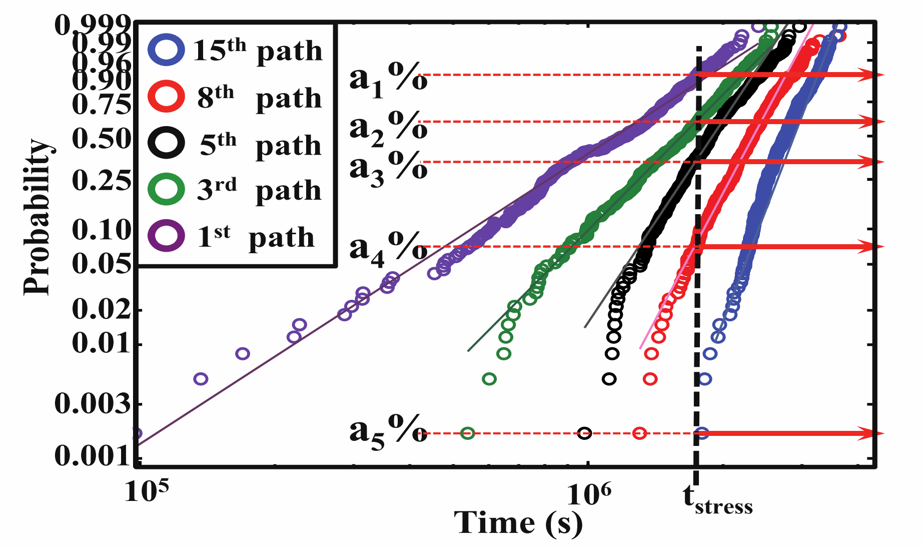
In the percolation model, the defect generation rate depends only on the gate voltage ($V_G$) and temperature ($T$). Therefore, during electrical stress, the trap density in the gate dielectric increases with stress time $t$ as a power law in the anode hole injection model. Stress is converted to a number of traps [66],[67]:

$$N_{trap}(t, V_G) = A\,exp(BV_G)\,t^{\beta} \cdot \tau_{ox} WL \cdot \exp(-\theta T) \tag{4}$$

where $A, B,$ and $\beta$ are fitting constants, and $\tau_{ox}, W,$ and $L$ are oxide thickness, gate width, and length, respectively, and $\theta \approx 0.01℃^{-1}$ [67].



(a)



(b)

**Figure 4 – Time distribution of defect generation in SiO2. (a) The probability distribution of the time of occurrence of the kth SBD path for different gate sizes. (b) The probability distribution of the number of SBD paths for a fixed gate size as a function of time.**

Figure 4 shows the PM simulation results and the probability distribution of the time of occurrence of conduction paths in the oxide layer as a function of gate size (Figure

4(a)). It also shows the number of SBD paths as a function of time (Figure 4(b)). Then, as it's seen in Figure 4(b), if we know the stress duration of an applied gate voltage to a MOSFET, the probability of a fixed number of conduction paths can be estimated.

The results in Figure 4 indicate the probability that at least $n$ paths are observed in the oxide. To find the probability that there are exactly $n$ paths, it is necessary to subtract the $n-1^{st}$ curve from the $n^{th}$ curve. A result is illustrated in Figure 5.

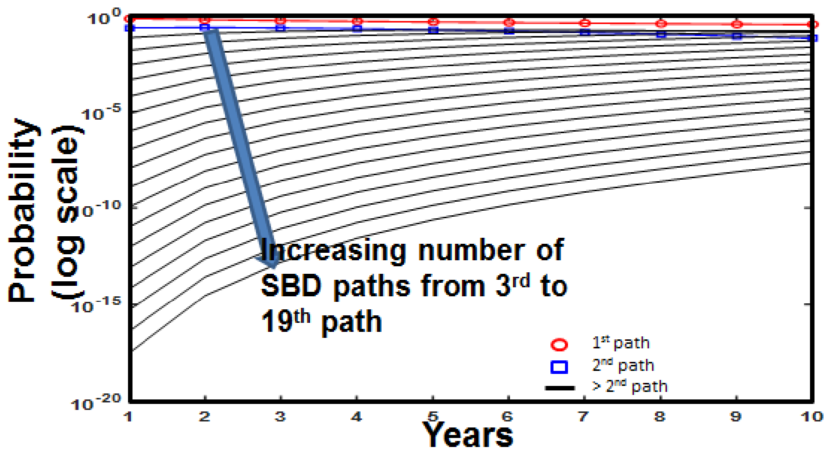


**Figure 5 – Probability of the kth SBD path for a fixed gate size and as a function of time.**

Note that the number of breakdown paths in the gate depends on the operating conditions. Hence, given a time under operation, the emulator determines the time under stress for each group of devices. This is used to look up probabilities of different numbers of SBD paths. The number of devices in the group multiplies the probabilities to estimate the expected number of devices with each number of breakdown events in the group.

The time under stress for the $i^{th}$ device is a function of bias. Let $\alpha_i$, where $0 \leq \alpha_i \leq 1$, be the fraction of time under stress for the $i^{th}$ device. Then, $t_{stress,i} = t\alpha_i$, where

$t$ is the time under operation. Let $p_i = f_{SBD(n)}(t_{stress,i})$ be a probability of $n$ SBD paths for the $i^{th}$ device. Therefore, $p_i = f_{SBD(n)}(t\alpha_i)$. If the group of devices has *N* devices, then $Np_i$ devices are randomly selected to have $n$ breakdown paths at time *t*. Each sample randomly selects the devices experiencing SBD.

Next, for each breakdown path, the SBD leakage resistance is calculated with the QPC model [68],

$$R_{SBD} \cong V_G / \left[ \frac{4e}{h\alpha} N \cdot exp(-\alpha\Phi) \cdot sinh\left( \frac{\alpha e(V_G - V_0)}{2} \right) \right] \quad (5)$$

where $\Phi = 3 \sim 4eV,\ V_0 = 0 \sim 0.5V,\ \alpha = 2{\sim}3eV^{-1},\ h$ is Plank's constant, $e$ is the electron charge, and *N* is number of SBD conduction paths [68]. The location of SBD, gate-to-drain vs. gate-to-source, is randomly selected. The resistance as a function of the number of SBD paths is illustrated in Figure 6.

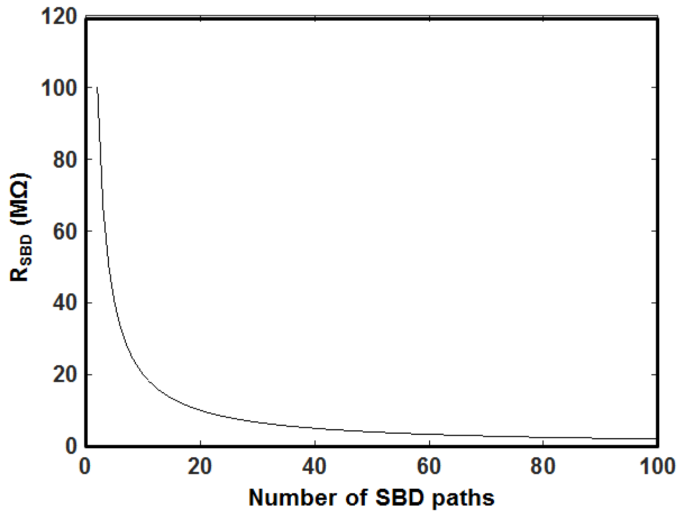


**Figure 6 – SBD resistance as a function of the number of SBD paths.**

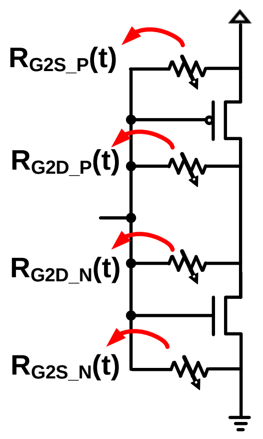


**Figure 7 – Device-level GOBD soft breakdown model used in this work.**

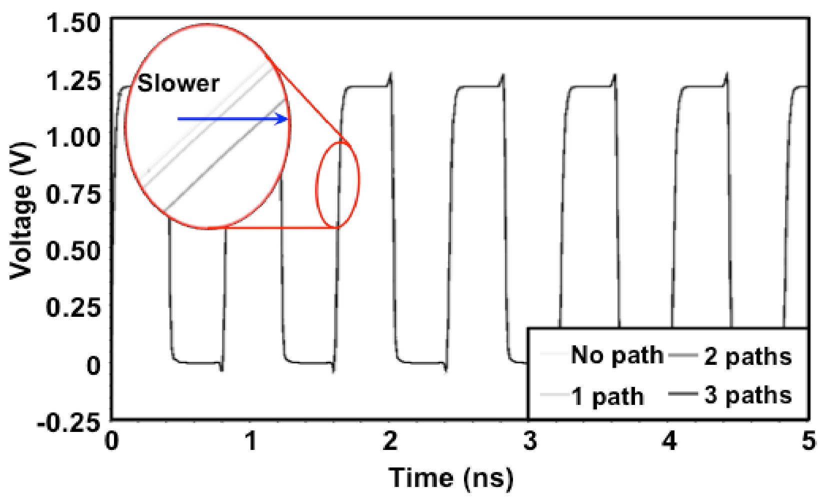


**Figure 8 – The impact of SBD on ring oscillator performance.**

In summary, the device-level GOBD model used in this work introduces two time-dependent resistances for each transistor, shown in Figure 7. The impact of SBD on a ring oscillator is illustrated in Figure 8. It shows the waveform comparison of the case with no SBD path, one SBD path, two SBD paths and three SBD paths. It can be seen that more SBD paths result in larger delays, while not degrading signal swing.

# 4. STANDARD CELL CHARACTERIZATION AND RC INTERCONNECT CHARACTERIZATION

This chapter introduces the method for standard cell characterization and interconnect characterization. The proposed method incorporates three sophisticated gate-level models: an input capacitances model, a sensitivity model of variational resistive-capacitive loads, and gate and interconnect delay models via the multivariate adaptive regression splines (MARSP) method.

## 4.1 Standard Cell Characterization

### *4.1.1 Variation Modelling*

**Table 1 – Variations and corners of PVT parameters**

| **Var.** | **Random Variations** | **Corners** | **Var.** | **Random Variations** | **Corners** |
|---|---|---|---|---|---|
| $\Delta L_p$ | Gaussian, 3σ=20% | [-20%, 20%] | $\Delta L_n$ | Gaussian, 3σ=20% | [-20%, 20%] |
| $\Delta Vth_p$ | Gaussian, 3σ=20% | [-20%, 20%] | $\Delta Vth_n$ | Gaussian, 3σ=20% | [-20%, 20%] |
| ΔVdd | Uniform, (-10%, 10%) | [-10%, 10%] | ΔT (°C) | Uniform, (-50, 50) | [-50,50] |
| Slope | Uniform, (10ps, 3ns) | [10ps, 3ns] | | | |

Table 1 presents the PVT parameters, where $\Delta L$ denotes channel length variation, $\Delta Vth$ denotes threshold voltage variation, $\Delta Vdd$ denotes supply voltage variation, $\Delta T$ is temperature variation, $Slope$ is the input transition time, and the subscripts $p$ and $n$

correspond to PMOS and NMOS devices, respectively. The percentages given in Table 1 are percentages of nominal values of the corresponding parameters. Corner information is given to facilitate later explanations.

The process parameters, namely $\Delta L_p, \Delta L_n, \Delta Vth_p, \Delta Vth_n$ in Table 1, correspond to each transistor. The value can be a combination of inter-die, intra-die, and intra-gate variation. The examples in this thesis consider inter-die variation for channel length and intra-gate variation for threshold voltage because lithography/etch has the strongest impact on channel length and random dopant fluctuations have the strongest impact on the threshold voltage. However, any process model with any between and within-die variation model can be implemented, including measured distributions and correlation profiles.

The voltage, temperature and slope parameters, namely $\Delta Vdd, \Delta T, Slope$, are applied to each cell. As will be shown in later sections, our method supports a temperature profile from a thermal simulator and a voltage profile from a power grid simulator which takes into account the IR-drop effects. Again, the temperature profile and voltage profile don’t have to be a specific distribution. They could be any form of distribution with a correlation structure.

### *4.1.2 Input Capacitances of Standard Cells*

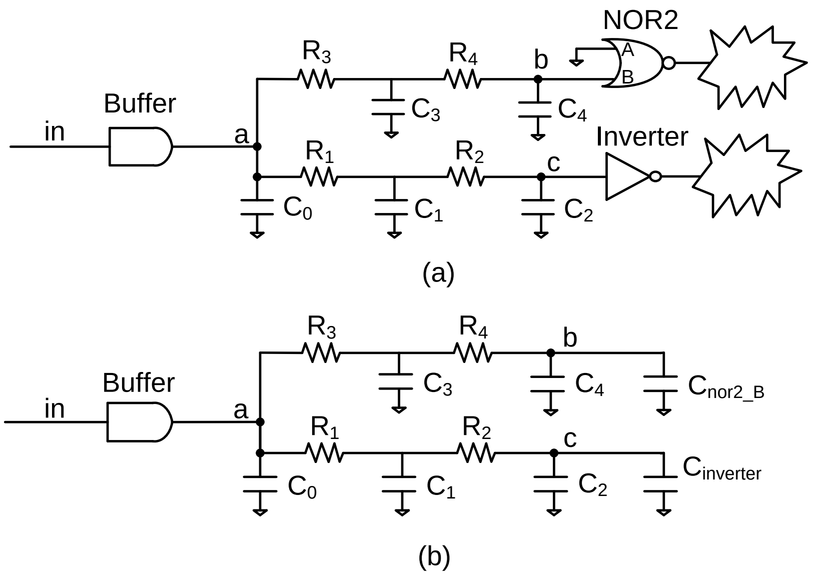


**Figure 9 – Input capacitances of standard cells, as well as the interconnect network, construct the load of previous gate. (a) A buffer and its interconnect load. (b) The buffer with the variational model of the input capacitance of the next stage incorporated into its load.**

Figure 9(a) illustrates a small patch of a gate-level circuit. The buffer gate in this example has two fanout gates, an inverter gate and a NOR2 gate. The load seen by the buffer is the interconnect network together with the input capacitances of its fanout gates, as shown in Figure 9(b). With fanout gates modeled as corresponding input capacitances, circuit-level timing analysis can be done stage by stage, in the way that each stage contains a standard cell and its connecting load as Figure 9(b) shows.

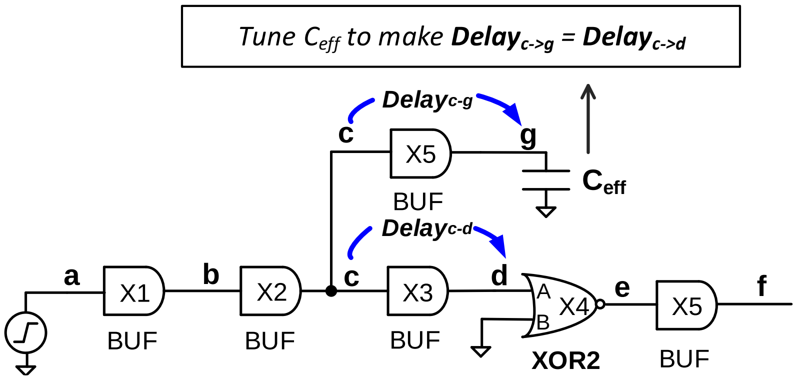

Tune $C_{eff}$ to make $\mathbf{Delay}_{c\text{->}g}$ = $\mathbf{Delay}_{c\text{->}d}$
Delayc-g
c
X5
g
BUF
Ceff
Delayc-d
a
X1
BUF
b
X2
BUF
c
X3
BUF
d
A
B
X4
XOR2
e
X5
BUF
f

**Figure 10 – Circuit used to characterize input capacitances of standard cells.**

With PVT variations in consideration, it is not satisfactory to model input capacitances as fixed values [4]. The variations of standard cell input capacitances must be taken into account. This work presents a modeling method to get variation-aware equations of input capacitances of standard cells. Firstly, a test circuit is used to characterize the actual input capacitance. Figure 10 shows the case of input A of XOR2 gate, where $C_{eff}$ is tuned until the delay from node *c* to node *g* is equal to the delay from node *c* to node *d*. The acquired value of $C_{eff}$ is the input capacitance of input A of gate *X4*. *X1* and *X2* are used to produce a reasonable input slope at node c, and *X5* is used as a load to prevent node e from switching excessively fast.

When PVT variations of gate *X4* are present, the acquired values of $C_{eff}$ vary. The PVT parameters in Table 1 are considered. That is, seven parameters are considered for an inverter gate, while eleven parameters are considered for a two-input gate. Central Composite Design [69], which uses the parameter corners in Table 1, is employed to design the experiments.

After the input capacitance of input *A* of gate *X4* has been characterized with each experiment, a first-order linear regression equation can be found as follows:

$$C_{input\_cap} = k_0 + \sum_{i=1}^{n} k_i X_i \tag{6}$$

where *n* is the number of considered PVT parameters, $k_0$ is the constant term, $X_i$, $i=1,2,\ldots,n$ denote considered PVT parameters, and $k_i$, $i=1,2,\ldots,n$ are the first-order sensitivity coefficients.

The input-capacitance model is tested in the context of Figure 9(b). Random variations of PVT parameters in Table 1 are applied to the three gates in Figure 9(a). *Slope* is only applied to the Buffer. Two methods are implemented to get the input capacitance used in Figure 9(b): one uses a fixed value for the pin capacitance from the standard cell library and the other uses the variational model in (6). The simulations were done using hSPICE [70], and the errors were obtained by comparing to the results from the circuit in Figure 9(a). Table 2 lists the average error of these two methods.

**Table 2 – Accuracy comparison using fixed input capacitance vs. variational input capacitance.**

| | **Gate Delay Error (node in to node a)** | **Interconnect Delay Error (node a to node b)** |
|---|---|---|
| Fixed Input Capacitance model | 5.88% | 6.91% |
| Proposed Input Capacitance model | 0.32% | -0.07% |

Please note that every input of a gate has its input capacitance. For example, a NOR2 gate has two input-capacitance models, for input *A* and input *B*, respectively. And for each input capacitance, scenarios for the rising-edge and the falling-edge are considered separately.

### *4.1.3 Approximations of Interconnect RC Networks Using Moments Matching*

An RC interconnect is a linear system with one input and one or multiple outputs. The interconnect input could be a primary input or a gate output, while the interconnect outputs could be a primary output or inputs of its loading gates. Consider the interconnect network in Figure 9(a), node *a* is the interconnect input and node *b* and *c* are the outputs.

For an interconnect network, we denote its input admittance function as *Y(s)* and its transfer functions as $H_1(s)$, $H_2(s)$ …, for each output. They can be expanded at $s=0$ using a Taylor series as follows (only showing one transfer function):

$$\mathrm{H(s)} = m_0 + m_1 s + m_2 s^2 + m_3 s^3 + \cdots \tag{7}$$

$$\mathrm{Y(s)} = y_0 + y_1 s + y_2 s^2 + y_3 s^3 + \cdots. \tag{8}$$

Here, $m_0$, $m_1$ … are called the moments of *H(s)*, while $y_0$, $y_1$ … are the moments of *Y(s)*. Please note that $y_0$ is zero and $m_0$ is one for RC trees [70]. The first, second and third moments of *Y(s)* are $y_1$, $y_2$ and $y_3$ respectively, and $m_1$, $m_2$ and $m_3$ are those for *H(s)* similarly. We use the modified nodal analysis (MNA) [70] method to generate moments in (7) and (8) via Matlab.

Reduced-order models are routinely used to replace the original large-order models. We will introduce the reduced-order model of *Y(s)* first, and then the reduced-order model of *H(s)*.

### 4.1.3.1 Pi-model as an Approximation of Y(s)

The Pi-model is the most popular reduced-order model to estimate the input admittance of RC interconnects. Figure 11 gives the structure of the Pi-model, where *Y(s)* denotes the input admittance of the original network and *Y'(s)* denotes the input admittance of the Pi-model.

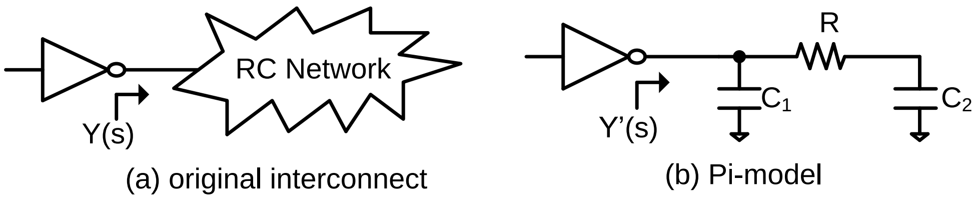


**Figure 11 – Y'(s) in Pi-model as an approximation of original input admittance function Y(s).**

The values of $C_1$, *R*, and $C_2$ are obtained by equating the first, second, and third moments of *Y(s)* to the first, second and third moments of *Y'(s),* respectively. The equations for the Pi-model are as follows:

$$C_2 = {y_2}^2/y_3 \tag{9}$$

$$R = -y_2/C_2^2 \tag{10}$$

$$C_1 = y_1 - C_2 \tag{11}$$

where *$y_1$, $y_2$* and *$y_3$* are the first three moments of *Y(s)* in (8).

Thus, the sample space of the input admittance function of interconnect is reduced down to a three-parameter space. These three parameters *$C_1$*, *R* and *$C_2$*, plus PVT parameters in Table 1, construct the parameter space of standard cell characterization which is introduced later in Section 4.1.4. Specifically, ten parameters are included for the characterization of an inverter, while fourteen parameters are needed for the characterizations of BUF, NOR2 and NAND2.

#### 4.1.3.2 H'(s)-model as an Approximation of H(s)

This work employs a stable two-pole (S2P) approximation [72] to get the reduced-order model of *H(s)*. S2P preserves the first three moments of *H(s)* and more importantly guarantees the generated model is stable. The obtained model, *H'(s)*, is a second-order model with two stable poles, where

$$H'(s) = \frac{k_1}{s+p_1} + \frac{k_2}{s+p_2} \tag{12}$$

and where

$$p_1 = -\frac{m_2}{m_3} \tag{13}$$

$$p_2 = p_1 \left| \frac{\frac{1}{m_1} - \frac{m_1}{m_2}}{\frac{m_1}{m_2} - \frac{m_2}{m_3}} \right| \tag{14}$$

$$k_1 = \frac{1 + m_1 p_2}{p_1 - p_2} p_1^2 \tag{15}$$

$$k_2 = -\frac{1 + m_1 p_1}{p_1 - p_2} p_2^2. \tag{16}$$

It can be easily deduced from (13) and (16) that

$$k_1 p_2 + k_2 p_1 = p_1 p_2. \tag{17}$$

*H'(s)* in (12) cannot be directly included in a netlist for circuit-level simulation. Therefore, this work implements it as a two-port network with the same transfer function, as shown in Figure 12.

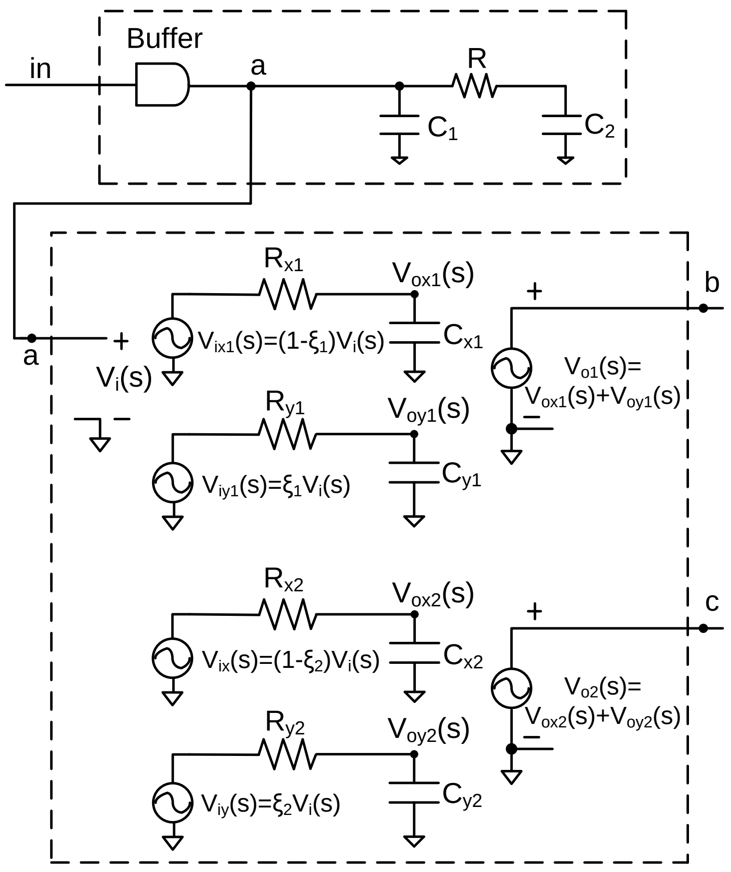


**Figure 12 – An illustration of single-stage timing analysis with a Pi-model for gate analysis and an H'(s)-model for interconnect analysis.**

In Figure 12, $V_i(s)$ is the input of this network while $V_o(s)$ is the output. $R_x$ and $C_x$ constitute a low-pass RC filter fed by a $V_i(s)$-controlled voltage source $V_{ix}(s)$ with gain $(1-\xi)$, while $R_y$ and $C_y$ form the other low-pass RC filter fed by a $V_i(s)$-controlled voltage source $V_{iy}(s)$ with gain $\xi$. The output voltages of the two filters, $V_{ox}(s)$ and $V_{oy}(s)$, are added to form $V_o(s)$.

Now, it's time to show how the transfer function in Figure 12 is exactly the same as in (12). The transfer function in Figure 12 is as follows:

$$H'(s) = \frac{V_o(s)}{V_i(s)} = \frac{V_{ox}(s) + V_{oy}(s)}{V_i(s)}$$

$$= \frac{\frac{\frac{1}{C_x s}}{\left(R_x + \frac{1}{C_x s}\right)}(1-\xi)V_i(s) + \frac{\frac{1}{C_y s}}{R_y + \frac{1}{C_y s}}\xi V_i(s)}{V_i(s)}$$

$$= \frac{\frac{1-\xi}{R_x C_x}}{s + \frac{1}{R_x C_x}} + \frac{\frac{\xi}{R_y C_y}}{s + \frac{1}{R_y C_y}} . \qquad (18)$$

By mapping (18) to (12), we get

$$\frac{1}{R_x C_x} = p_1 \qquad (19)$$

$$\frac{1}{R_y C_y} = p_2 \qquad (20)$$

$$\frac{1-\xi}{R_x C_x} = k_1 \qquad (21)$$

$$\frac{\xi}{R_y C_y} = k_2 . \qquad (22)$$

It is easily seen that (17) still holds for (13)–(16), so that *H'(s)* in (18) is exactly the same as in (12).

As can be seen from (19) and (20), $R_xC_x$ is determined by $1/p_1$ while $R_yC_y$ is similarly determined by $1/p_2$. In our work, we set $C_x$ and $C_y$ to a fixed value, $10^{-15}$F, leaving $R_x$ and

$R_y$ to be calculated according to (19) and (20), respectively. Therefore, we actually have three parameters in the *H'(s)*-model, $R_x$, $R_y$ and $\xi$, which means *H(s)* is simplified to a three-parameter space.

### *4.1.4 Sensitivity of the Pi-Model and the H'(s)-Model to Variations of Input Capacitances in the Fanout Gates*

As input capacitances of fanout gates are included as part of the interconnect, the variations of these capacitances cause the Pi-model and the *H'(s)*-model to vary. Here we denote the variational input capacitances of fanout gates of one interconnect network as $\overrightarrow{C_{fanout}} = [C_{fanout\_1}, C_{fanout\_2}, \dots, C_{fanout\_N}]$ where $N$ is the number of fanout gates at the output of this interconnect network. It is impractical to run the whole process, i.e., moment generation and calculations of $C_1$, $R$, $C_2$, $R_x$, $R_y$, and $\xi$, for every variational sample of $\overrightarrow{C_{fanout}}$. Thus, we expand a first-order Taylor-series at nominal values of $\overrightarrow{C_{fanout}}$. Let's take $C_1$ of the Pi-model as an example, since the other parameters ($R$, $C_2$, $R_x$, $R_y$ and $\xi$) are similar.

$$C_1 = C_{1\_nominal} + \sum_{i=1}^{N} \alpha_i (C_{fanout\_i} - C_{fanout\_i\_nominal}) \tag{23}$$

where $C_{fanout\_i\_nominal}$ is the nominal value of $C_{fanout\_i}$, $\alpha_i$ is the first derivative of $C_1$ with respect to $C_{fanout\_i}$, $i \epsilon \{1,2, \dots, N\}$, and $C_{1\_nominal}$ is the nominal value of $C_1$.

The errors in using (23) to calculate the Pi-model and the *H'(s)*-model parameters are all less than 0.1% in our test RC interconnects. Please note that (23) is unique for each interconnect in a circuit, but needs to be characterized only once and the characterization time is negligible because the interconnect network size in our experiments throughout this thesis is not large (less than 100 RC segments).

### *4.1.5 Gate-Delay Characterization using Multivariate Adaptive Regression Splines (MARSP)*

As mentioned in Section 4.1.2, with fanout gates being modeled as corresponding input capacitances, a circuit can be divided into gate-level stages for timing analysis as shown in Figure 9(b). The timing analysis of each stage incorporates two parts: gate delay/output-slew modeling and interconnect delay/output-slew modeling. The output of the buffer in Figure 9(b), node *a*, is also the input of the interconnect network. The delay of the buffer is the delay from node *in* to node *a*, while the delay of the interconnect network is the delay from node *a* to node *b* or from node *a* to node *c*. The output slew of the buffer is used as the input slew of the interconnect. The total delay of each stage is the sum of the gate delay and the interconnect delay.

Using the Pi-model and *H'(s)*-model, the circuit in Figure 9(b) is transformed into Figure 12. The upper box in Figure 12 represents gate timing analysis, while the lower box represents interconnect timing analysis. In this section, MARSP is used to characterize the delay and slew models of standard cells.

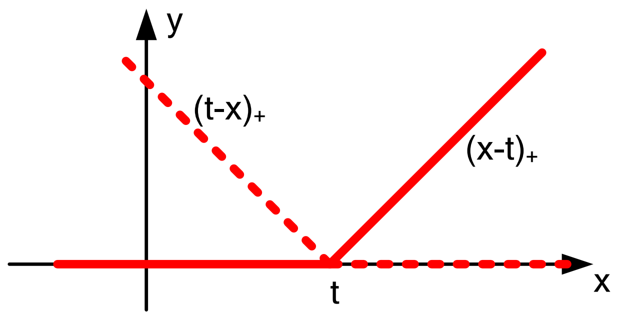


**Figure 13 – The solid line denotes the form of the hinge function (x-t)+ while the dashed line denotes the hinge function (t-x)+.**

This work employs MARSP [73] to characterize a fitted function between response variables (gate delay or slew time) and the explanatory parameters (PVT parameters, aging parameters, and RC loads). MARSP is an adaptive procedure that uses piecewise linear segments and is well suited for high-dimensional problems while capturing essential nonlinearities and interactions.

The piecewise nature of MARSP allows it to split the whole high-dimension parameter space into multiple subspaces, and each subspace has a unique regression model. MARSP inherently integrates all the regression models of different subspaces into one general expression using piecewise hinge functions [73].

A hinge function has the form of $(x-t)_+$ or $(t-x)_+$ which are shown in Figure 13. They are defined as:

$$(x-t)_+ = \begin{cases} x-t, & if\ x>t, \\ 0, & otherwise, \end{cases} \tag{24}$$

$$(t-x)_+ = \begin{cases} t-x, & if\ x<t, \\ 0, & otherwise, \end{cases} \tag{25}$$

where $t$ is a constant, called the knot. MARSP forms a collection of hinge-function pairs for each explanatory parameter $X_j$ with knots at $x_{j1}, x_{j2}, \ldots, x_{jM}$, where $M$ is the number of experiments.

MARSP models have the following form:

$$f(\vec{X}) = \beta_0 + \sum_{t=1}^{T} \beta_t \, h_t(\vec{X}) \tag{26}$$

where $h_t(\vec{X})$ is called a basis function. (Constant 1 is the basis function of the intercept term.) MARSP builds a model in two phases: the forward stepwise addition and the backward stepwise deletion.

In the forward phase, MARSP starts with a model which consists of an intercept term. Then it repeatedly adds basis functions in pairs to the model step by step. It finds the pair of basis functions that gives the maximum reduction in the sum-of-squares residue error. The two basis functions in the pair are identical except that a different side of a mirrored hinge function is used for each function. Each new basis function consists of a term already in the model (which could be a constant 1) multiplied by a new hinge function. The process of forward stepwise addition continues until the change in residual error is smaller than a threshold or until the maximum number of terms is reached.

After the forward stepwise addition, we have a large model which typically overfits the data. An overfit model has a good fit to the data used to build the model but will not generalize to new data. To build a model with a better generalization ability, backward stepwise deletion prunes the model. The backward phase uses Generalized Cross

Validation (GCV) to choose the best model subset. The GCV formula trades off goodness-of-fit against model complexity. The backward stepwise deletion removes model terms one by one, deleting the least important term (according GCV) at each step until the model again has only the intercept term. At the end of the backward phase, from among the "best" models of each size, the one with the lowest GCV value is selected and outputted as the final one.

MARSP is a form of nonparametric regression which doesn't take a predetermined form, but constructs the model structure according to the information derived from the data. It 'filters out' the negligible parameters without manual intervention which eliminates the need for categorization (or clustering) of switching/non-switching transistors and on-transition/off-transition/non-transition transistors, as in [19],[22].

Why is MARSP better than the traditional regression technique? The traditional regression technique or the Response Surface Methodology (RSM) technique suffers from the disadvantage of using the same model to cover the entire parameter space. One model is insufficient to accurately estimate the gate delay (or slew time) over the whole parameter space, especially when the dimension of the parameter space is large, e.g., in the case where intra-gate variability is considered. Therefore, the timing behavior must be characterized separately for different PVT subspaces (corners) which results in cumbersome parameter space splitting and large characterization efforts.

The piecewise nature of MARSP allows it to split the whole high-dimension parameter space into multiple subspaces, and each subspace has a unique regression model. MARSP inherently integrates all the regression models of different subspaces into one general expression using piecewise hinge functions. Therefore, MARSP saves characterization effort by characterizing standard cells only once over the whole PVT space, including the corners.

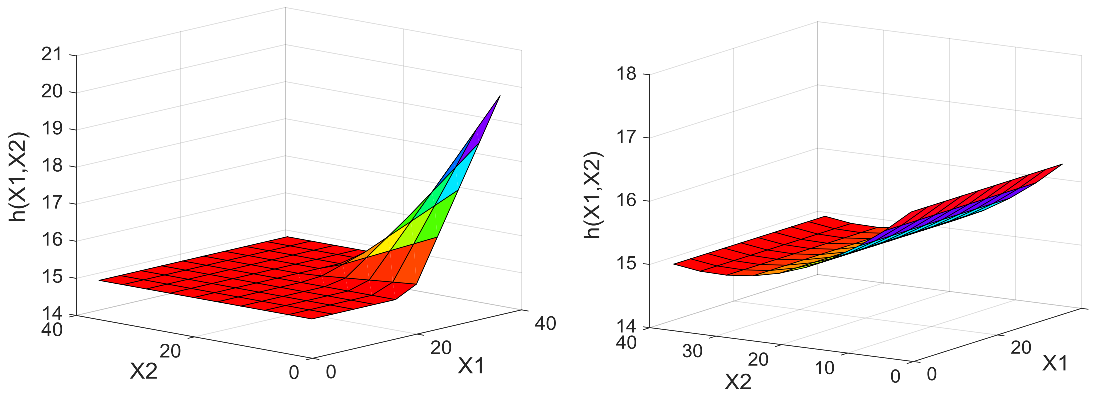


**Figure 14 – The efficiency and accuracy of MARSP model is shown for a non-linear case. (a) a nonlinear function $\boldsymbol{h(X_1, X_2)}$ which changes only when X1 is high and X2 is low. (b) a quadratic model regressed from $\boldsymbol{h(X_1, X_2)}$.**

Figure 14(a) shows a situation where the response variable *h(*$X_1$*,*$X_2$*)* only changes at the corner where $X_1$ is large and $X_2$ is low. The MARSP model can easily handle this nonlinear relationship by using hinge functions as follows:

$$h(X_1, X_2) = 15 + 0.015 * (20 - x)_+ * (y - 20)_+ \tag{27}$$

Figure 14(b) shows the quadratic regression model which has poor accuracy for this case.

### *4.1.6 Experimental Results*

**Table 3 – Variations and Corners of Pi-model and H’(s)-model**

| **Var.** | **Random Variation** | **Corners** | **Var.** | **Random Variation** | **Corners** |
|---|---|---|---|---|---|
| $C_2$ (fF) | Uniform, (1,100) | [1,100] | $R_x$ (Ohm) | Uniform, (0.5, 1000) | [0.5, 1000] |
| $R$ (Ohm) | Uniform, (1,500) | [1, 500] | $R_y$ (Ohm) | Uniform, (0.5, 1000) | [0.5, 1000] |
| $C_1$ (fF) | Uniform, (0.01,1) | [0.01,1] | $\xi$ | Uniform, (-0.99,0.99) | [-0.99,0.99] |

Central Composite Design [69] is employed to design the characterization experiments, and then hSPICE [70] is used to run the simulations. The distributions and corners of PVT parameters are as shown in Table 1, while the distributions and corners of $C_1$, $R$ and $C_2$ are as shown in Table 3. The implementation of MARSP is from a Matlab toolbox called ARESLab [74]. Note that rising-edge and falling-edge scenarios are characterized separately. Besides, for a two-input cell, for example, NOR2, each input is characterized separately with the other input being at a non-controlling value.

**Table 4 – Comparison of MARSP and quadratic models on representative standard cells.**

| Cell | Dimen. | Char. Time (s) | | Number of Operations (worst case in MARSP) | | Number of Operations (RSM) | | Char. Error | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | MARSP | | RSM | |
| | | MARSP | Quadr. | Mul. | Add. | Mul. | Add. | mean | SD | mean | SD |
| INVX1 | 10 | 480 | 1 | 189 | 108 | 120 | 65 | 0.28% | 2.13% | -1.68% | 5.01% |
| NOR2X1 | 14 | 683 | 1 | 182 | 104 | 224 | 119 | 0.01% | 2.35% | -2.90% | 7.48% |
| CLKBUF3 | 70 | 3577 | 79 | 227 | 126 | 280 | 148 | 0.02% | 2.51% | -4.64% | 6.45% |
| XOR2X1 | 30 | 1503 | 2 | 203 | 116 | 274 | 145 | 0.06% | 2.17% | -3.82% | 7.28% |
| DFFPOSX1 | 50 | 2489 | 42 | 223 | 124 | 278 | 146 | 0.08% | 2.36% | -4.03% | 6.79% |
| Inter. | 5 | 22 | 1 | 44 | 28 | 26 | 14 | 0.01% | 0.12% | -1.59% | 3.53% |

Table 4 presents the results of some representative cells using NCSU 45nm FreePDK [75]. It can be seen that MARSP is substantially more accurate than the quadratic method. This accuracy is important, because these errors accumulate when cells are cascaded together. This is because, and error in a previous stage is amplified in the next stage. Hence, a small error in cell characterization can become a large error for the circuit. This accuracy, however does come at a cost in characterization time, as noted in Table 4.

#### 4.1.6.1 Validation Using Test Paths

The characterized gate-delay models are also validated using some test paths from ISCAS85 benchmarks [76]. ISCAS85 benchmark circuits were synthesized and the obtained netlists go to Cadence Encounter [77] to do place and route, and then parasitic RC interconnects were extracted using QRC [78]. The longest path of each circuit is identified using a script, and in total we considered ten test paths. The ten paths are not necessarily the critical paths when considering process variations. They are selected simply to determine the accuracy of timing analysis for paths. We implemented our framework using C++ and Perl. The experiments were run on a Linux platform with a 2.27GHz CPU and 4GB memory.

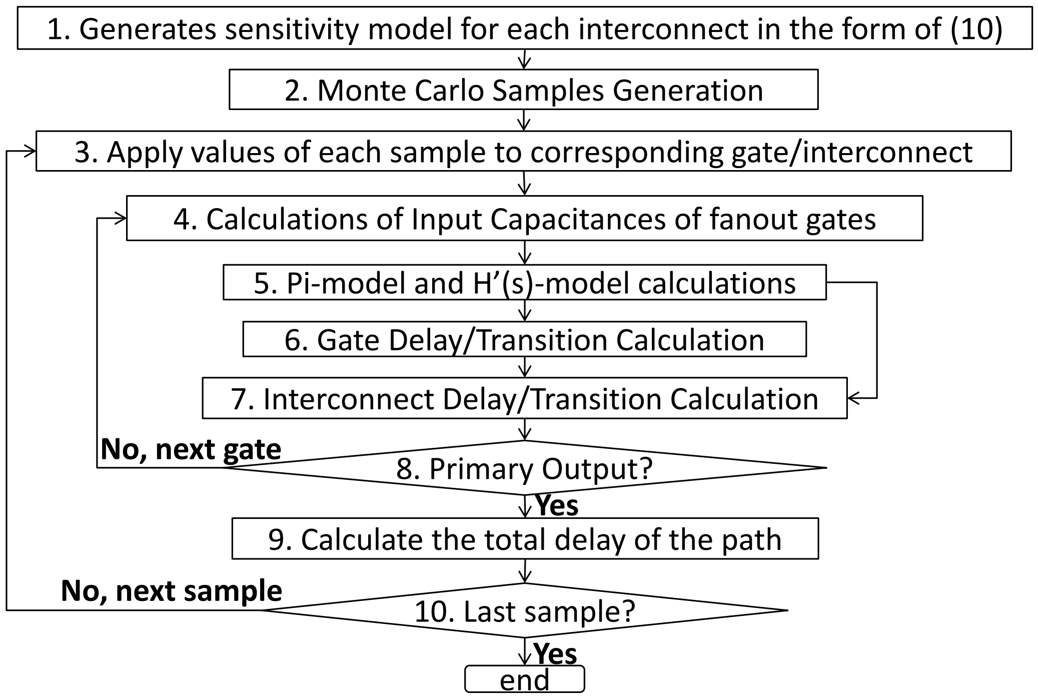
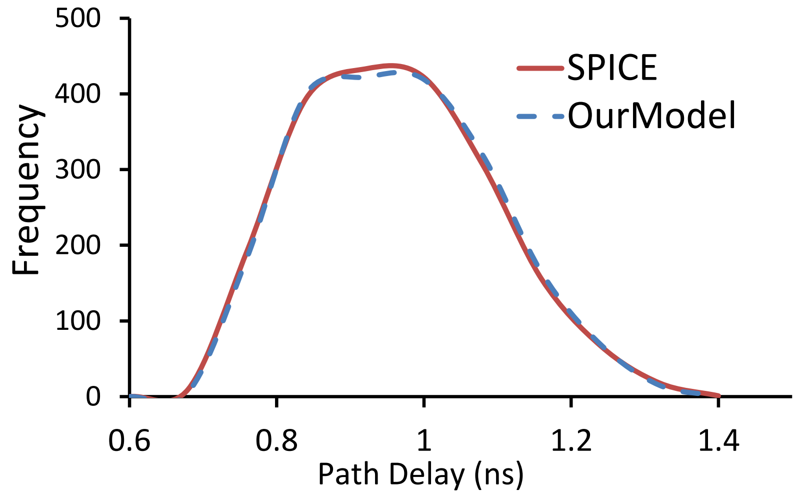
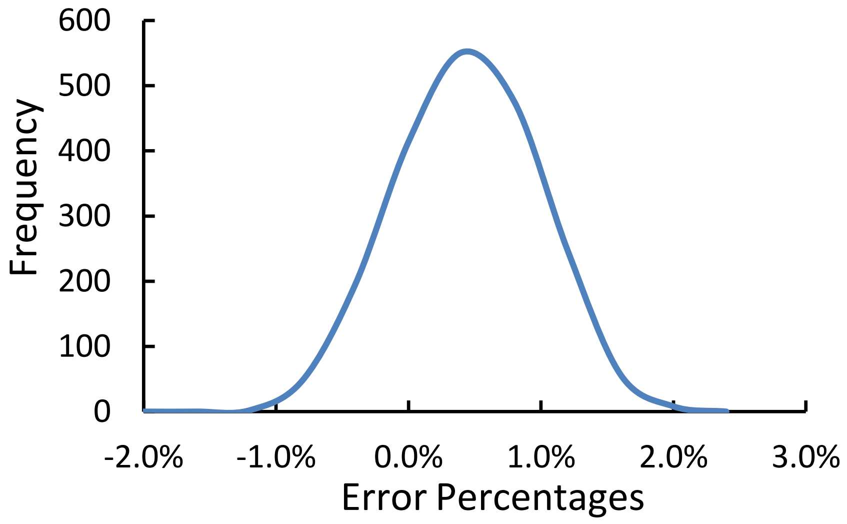

1. Generates sensitivity model for each interconnect in the form of (10)
2. Monte Carlo Samples Generation
3. Apply values of each sample to corresponding gate/interconnect
4. Calculations of Input Capacitances of fanout gates
5. Pi-model and H'(s)-model calculations
6. Gate Delay/Transition Calculation
7. Interconnect Delay/Transition Calculation
No, next gate
8. Primary Output?
Yes
9. Calculate the total delay of the path
No, next sample
10. Last sample?
Yes
end

**Figure 15 – Flow graph of the implementation which applies gate-level models to path-delay analysis.**

Figure 15 shows the implementation flow for extending our gate models to estimating path-delay distributions. Table 5 presents the results in comparison to hSPICE [70] using ten test paths. Figure 16 gives a histogram comparison for one of the paths between hSPICE and our model. The average mean and standard deviation (SD) values of errors per sample are 0.71% and 1.25%, respectively.



(a)



(b)

**Figure 16 – Experimental results of validating the proposed gate-delay models on a 43-stage test path. (a): Monte Carlo histogram comparison between our model and SPICE for test path (N85 to N724) in circuit c499. (2000 samples were run.) (b): error percentage histogram for this test case.**

**Table 5 – Experimental results of validating the proposed gate-delay models on ISCAS85 benchmarks.**

| Path | Circuit Name | PI to PO | Num. of stages | Num. of samples | Running Time | | | Path-delay error per sample (MARSP) | | Path-delay error per sample (quadratic) | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | SPICE (s) | MARSP (s) | Quadr. (s) | Mean | SD | Mean | SD |
| 1 | c432 | N102 to N421 | 60 | 2000 | 4211 | 119 | 192 | 1.21% | 1.21% | -10.99% | 24.9% |
| 2 | c499 | N85 to N724 | 43 | 2000 | 2417 | 84 | 138 | 0.25% | 0.54% | -13.95% | 26.6% |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | c880 | N1 to N878 | 57 | 2000 | 4148 | 113 | 185 | -0.31% | 0.53% | -14.10% | 25.4% |
| 4 | c1355 | G11 to G1352 | 42 | 2000 | 2412 | 86 | 144 | 1.54% | 1.49% | -10.68% | 26.7% |
| 5 | c1908 | N19 to N2890 | 72 | 2000 | 5878 | 155 | 250 | 2.12% | 1.92% | -10.76% | 26.9% |
| 6 | c2670 | N227 to N3851 | 54 | 2000 | 4001 | 110 | 180 | 0.90% | 1.16% | -15.81% | 25.1% |
| 7 | c3540 | N33 to N5360 | 77 | 2000 | 6719 | 166 | 270 | 0.75% | 1.10% | -5.94% | 27.5% |
| 8 | c5315 | N335 to N8128 | 71 | 2000 | 6454 | 153 | 251 | -1.53% | 1.55% | -11.72% | 26.3% |
| 9 | c6288 | N290 to N6287 | 221 | 2000 | 19898 | 422 | 698 | 1.29% | 1.50% | -14.43% | 23.1% |
| 10 | c7552 | N18 to N11334 | 116 | 2000 | 9948 | 228 | 378 | 0.87% | 1.48% | -10.23% | 25.8% |
| avg. | - | - | 81 | 2000 | 6608 | 164 | 268 | 0.71% | 1.25% | -11.86% | 25.8% |

## 4.2 Interconnect Characterization

For the timing analysis of interconnect, we consider $R_x$, $R_y$ and $\xi$ of the *H'(s)*-model and *VDD* and *slope* in Table 1, which are the five explanatory parameters. The characterization variables are interconnect delay and interconnect transition time which are defined similarly to standard cell characterization.

For the design of experiments, 4000 Monte Carlo simulations in hSpice [70] were used, where the distributions of $R_x$, $R_y$ and $\xi$ are as in Table 3 and distributions of *VDD* and *slope* are as in Table 1. Again, MARSP methods are employed to characterize the fitted functions. Then another 1000 Monte Carlo samples are used to test the generated models.

The characterization results for interconnect are also shown in the last row of Table 4. This research work doesn't consider interconnect variability (spacing, width). Please note that a higher-order *H'(s)*-model which matches more moments of the original *H(s)* can be easily put into our methodology at the expense of adding more parameters to the MARSP models.

# 5. MONTE CARLO BASED FRAMEWORK FOR CIRCUIT-LEVEL STATISTICAL TIMING ANLAYSIS

This chapter proposes a framework of statistical timing analysis (StTA) which utilizes the MARSP-based gate-delay models shown in Chapter 4. The proposed statistical timing analyzer contains two procedures: block-based StTA and path-based StTA. Block-based StTA performs statistical critical-path extraction using static timing analysis. Then the extracted critical paths are fed into path-based StTA to perform input-vector-dependent dynamic timing analysis to generate accurate circuit-delay distributions.

The proposed timing analyzer is implemented with C++ and Perl, and the experiments were run on a Linux platform with a 2.27GHz CPU and 1GB memory without using multi-threading. To verify the performance of our framework, ISCAS85 benchmarks [76] were implemented with NCSU 45nm technology [75], and test six industrial designs (two from a sponsor and four from the IWLS2005 [79] benchmarks) were implemented with a commercial 90nm technology. All the experiments are based on the following settings: $\Delta Vth$ is subject to intra-gate Gaussian variations and $\Delta L$ is due to inter-die Gaussian variations with a three-sigma value equal to their corners in Table 1. All other parameters are uniformly distributed between their corners in Table 1 and Table 3.

## 5.1 Block-Based Statistical Timing Analyzer

### *5.1.1 Implementation*

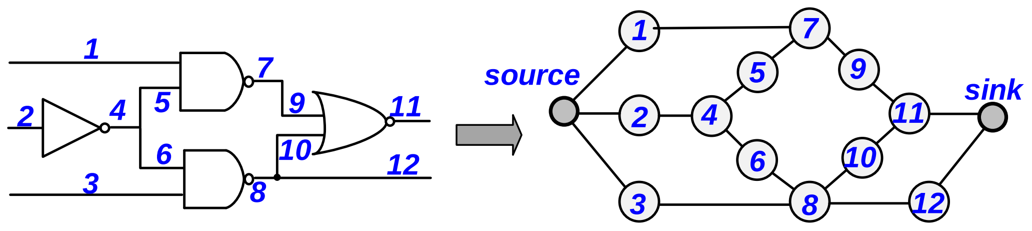


**Figure 17 – Abstraction of a timing graph from a combinational circuit.**

While probabilistic block-based StTA performs statistical sum and max operations between random variables, Monte Carlo block-based StTA requires sum and max operations between numerical values. The timing graph which is used in traditional static timing analysis (STA) is abstracted from a combinational circuit, as shown in Figure 17. The nodes of the timing graph represent primary inputs/outputs of the circuit and gate input/output pins. The edges of the timing graph represent gate input-pin-output-pin delay and interconnect delay. Each delay (gate delay or interconnect delay) comes from the maximum value of all the switching scenarios. After a forward traversal of the timing graph, the circuit delay is obtained from the maximum arrival time of all primary outputs. And by a back traversal, the path which results in the maximum circuit delay is identified as the most critical path. For each Monte Carlo sample, the process of the forward traversal and the back traversal is repeated to produce a circuit delay and a critical path. With a number of samples, the distribution of circuit delay and a set of critical paths are obtained in the presence of process variations.

### *5.1.2 Experimental Results*

Theoretically, block-based StTA produces an overestimate of circuit delay in comparison with SPICE results, because SPICE calculates the gate delay according to the real switching propagation (so called dynamic timing analysis) while block-based StTA propagates arrival time without considering the switching directions (so called static timing analysis). Experimental results confirm this theory, showing our block-based implementation has an average 7.42% overestimate of SPICE results on ten ISCAS85 benchmark circuits [76].

**Table 6 – Comparison of the first 10 circuit paths extracted from our Block-Based StTA and STA**

| Ranked critical paths from proposed Block-based StTA | The possibilities of being critical path | Rank in STA |
|---|---|---|
| 1st | 9.20% | 22th |
| 2nd | 8.20% | 8th |
| 3rd | 7.10% | 6th |
| 4th | 7.05% | Beyond the 100th |
| 5th | 7.05% | 35th |
| 6th | 6.05% | 7th |
| 7th | 5.95% | 11th |
| 8th | 5.55% | 1th |
| 9th | 4.90% | Beyond the 100th |
| 10th | 3.90% | 70th |

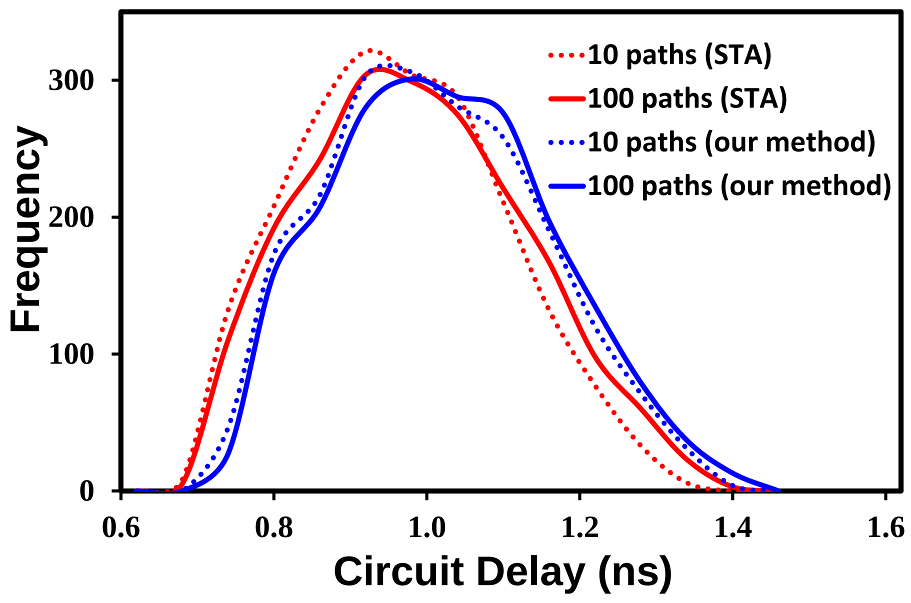


**Figure 18 – Comparison of the circuit delay distributions of circuit c499 from SPICE using STA and our block-based StTA. (2000 samples were run.)**

Because of process variations, the application of traditional STA may not identify all critical paths. Paths vary randomly with process variations. Even though there are errors produced by uncertainty in switching scenarios, this block-based StTA method can identify critical paths under process variations more accurately than the traditional STA method in the presence of process variations. These paths will be used by the more accurate path-based StTA to find the delay distribution for the circuit.

As one critical path is identified for each Monte Carlo sample, a set of critical paths are obtained after a number of samples are run. Therefore, the possibility of each path being critical can be calculated by dividing the frequency that each path is critical by the number of total samples. Table 6 shows the comparison of extracted critical paths by block-based StTA and PrimeTime [80] STA for benchmark c499. It lists the first 10 critical paths along with their possibilities of being critical, and compare with their rank in STA. In Figure 18, it presents a comprehensive comparison of hSpice (using the first 10 paths from STA), hSpice (using the first 100 paths from block-based STA), hSpice (using the first 10 paths from block-based StTA) and hSpice (using the first 100 paths from block-based StTA) for benchmark c499. The experimental results validate the point that STA results are short of covering the real critical paths when process variations are taken into account. As it can be seen from the results, even the first 100 paths from STA has smaller delay than the first 10 paths from StTA. This matches Table 6 where the 4th and 9th paths from StTA are beyond the 100th in STA.

### *5.1.3 Complexity analysis*

The complexity of the proposed block-based framework is *O(MN)*, where *M* is the number of gates and interconnects and *N* is the number of samples. Figure 19 presents the linearity between runtime per sample and the average depth of the extracted paths.

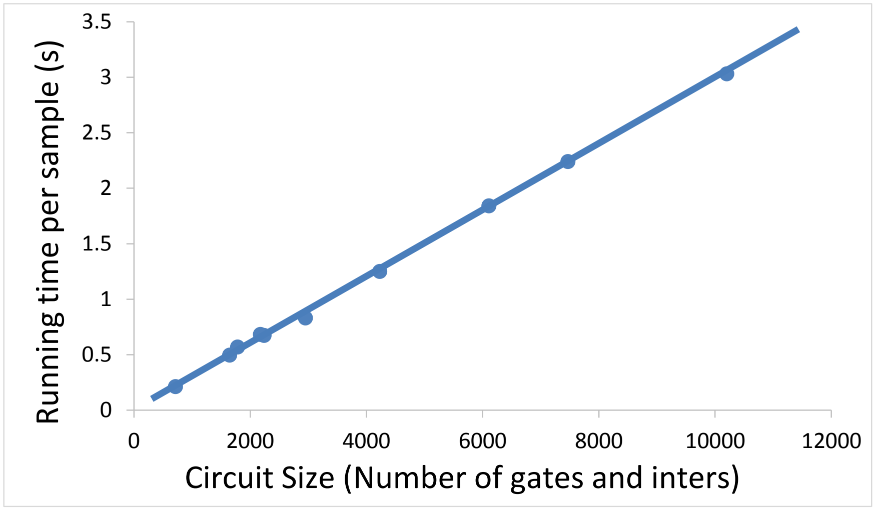


**Figure 19 – The runtime per sample versus the number of gates and interconnects in block-based StTA.**

## 5.2 Paths-Based Statistical Timing Analyzer

### *5.2.1 Implementations*

In probabilistic path-based StTA, a set of paths, which is likely to be critical, is drawn first. Then statistical analysis is performed over these paths to estimate the circuit-delay distribution. In our Monte Carlo based framework, the statistical analysis over those extracted paths is based on samples rather than probabilistic operations.

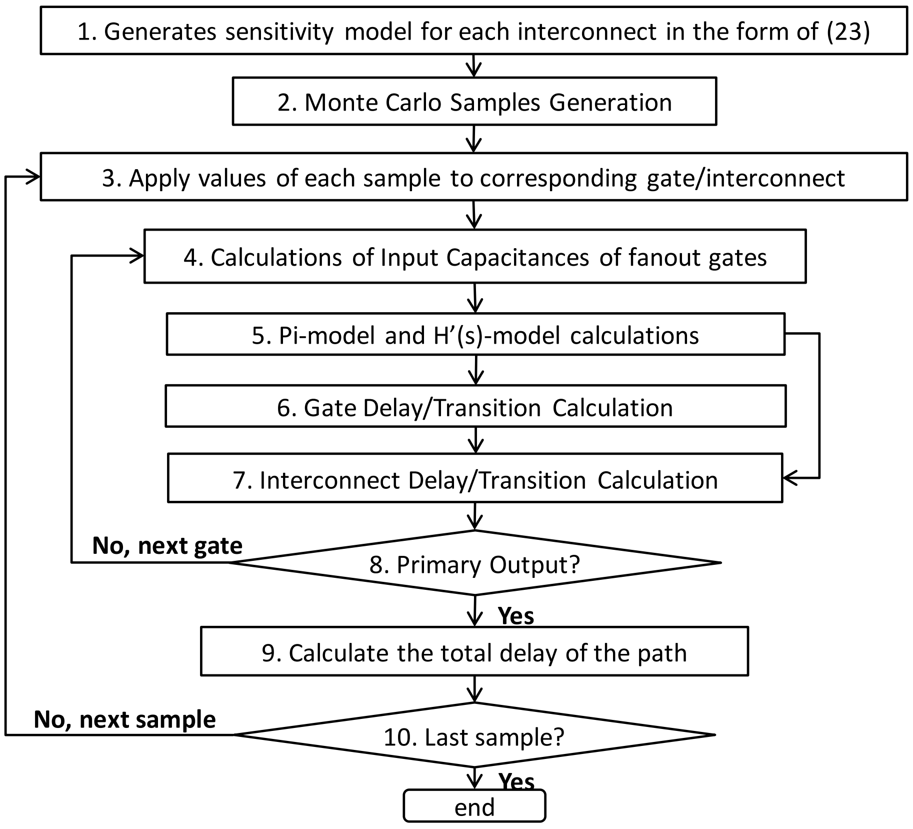

1. Generates sensitivity model for each interconnect in the form of (23)
2. Monte Carlo Samples Generation
3. Apply values of each sample to corresponding gate/interconnect
4. Calculations of Input Capacitances of fanout gates
5. Pi-model and H'(s)-model calculations
6. Gate Delay/Transition Calculation
7. Interconnect Delay/Transition Calculation
No, next gate
8. Primary Output?
Yes
9. Calculate the total delay of the path
No, next sample
10. Last sample?
Yes
end

**Figure 20 – Flow graph of the implementation which applies gate-level models to path-delay analysis.**

More specifically, Monte Carlo samples over the extracted paths are generated, and then the delays of extracted paths for each sample are calculated via the framework in Figure 20. Circuit delay is obtained by taking the maximum of the path delays. With a number of samples, the circuit-delay distribution can be found.

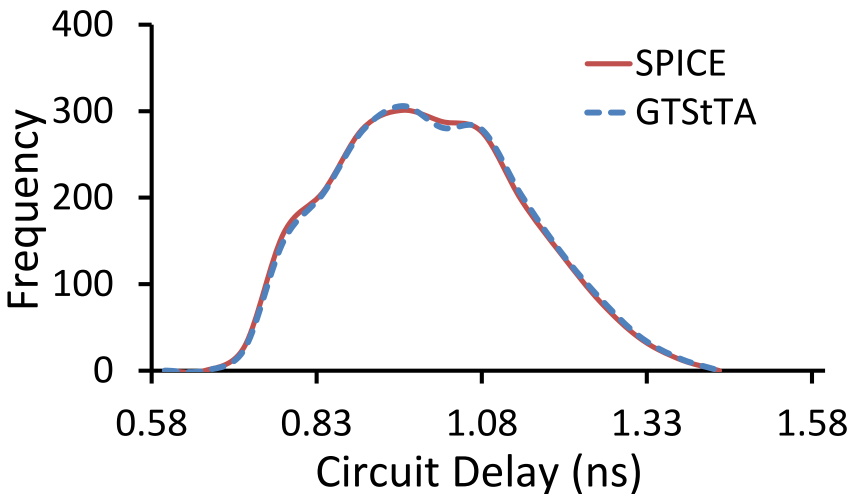
The detailed implementation procedures to perform path-based StTA are shown in Table 7.

**Table 7 – Implementation of Path-Based StTA**

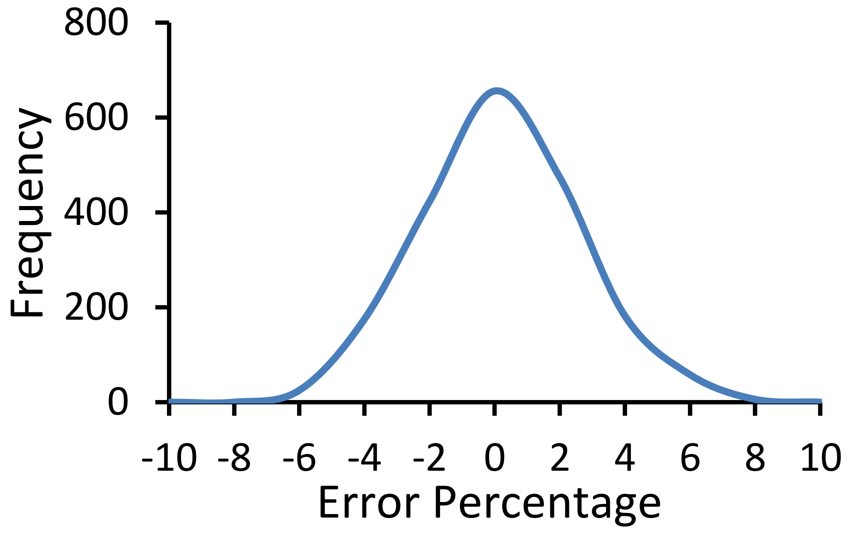
```
Input: Netlists of extracted paths, the number of extracted paths $M$, and the number of samples $N$.
Output: $CD$ (circuit delays for all $N$ samples)

1: generate $N$ samples for the extracted paths (distributions according to Table 1)
2: for i=1 to $N$, do
3:     for j=1 to $M$, do
4:         $PD_j$ = Delay of Path j (using the framework in Figure 20)
5:     end for
6:     $CD_i$ = maximum ($PD_1, PD_2, \ldots, PD_M$)
7: end for
8: return $CD = [CD_1, CD_2, \ldots, CD_N]$
```

### *5.2.2 Experimental Results*

The results of our framework (called GTStTA) are compared to SPICE. On average, the errors of mean and standard deviation (SD) values of the circuit delay distribution are 1.47% and -1.15%, respectively, while the mean and SD values of errors per sample are 1.97% and 1.55%, respectively. The low error percentage in each sample shows that our framework captures the real circuit delays very well with each sample, and thus ensures the accuracy of the circuit delay distribution. Figure 21(a) shows the histogram of our path-based StTA and SPICE for circuit c499, where the sample size is 2000. And Figure 21(b) shows the error distribution of this testcase.



(a)



(b)

**Figure 21 – (a) Comparison of the circuit delay distributions of circuit c499 from SPICE (using different critical path pools). (2000 samples were run.). (b) The error distribution.**

### *5.2.3 Complexity*

The complexity of our path-based framework is *O(PMN),* where *P* is the number of extracted paths, *M* is the average number of stages (depth) of the extracted paths and *N* is the number of samples. Figure 22 presents the linearity between runtime per sample and average depth of the extracted paths.

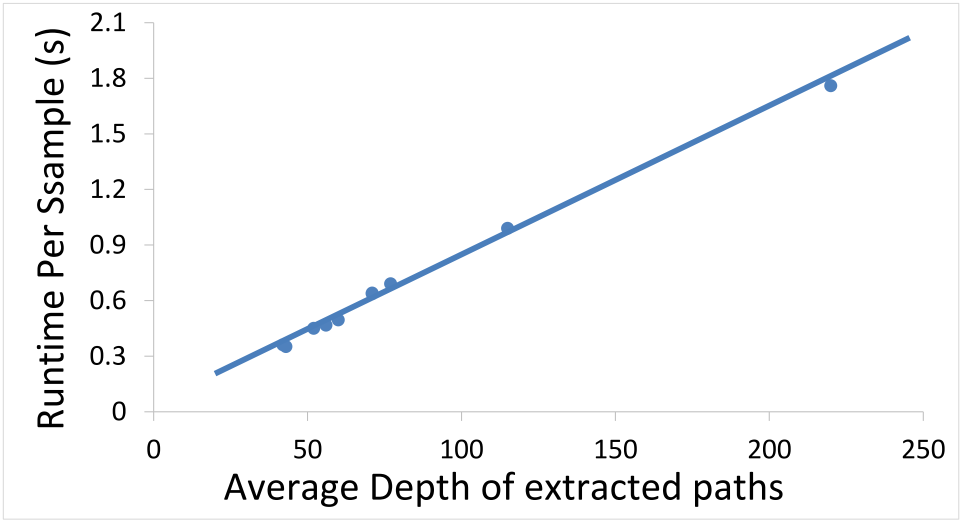


**Figure 22 – The runtime per sample versus the number of gates and interconnects in paths-based StTA.**

## 5.3 Block-Based and Path-Based Combined Statistical Timing Analyzer

### *5.3.1 Implementations*

Block-based StTA conducts statistical critical-path selection, and path-based StTA achieves SPICE-level accuracy. The extracted critical paths from block-based StTA are fed into path-based StTA to perform accurate input-vector-dependent timing analysis to generate circuit-delay distribution. The detailed implementation is shown in Table 8. For each Monte Carlo sample, one critical path is extracted using block-based StTA, then path-based StTA generates the delay of this extracted path according to the same sample. Once a number of samples are run, the circuit-delay distribution is obtained. This method considers all the potential paths extracted under the process variations and maintains the advantage of accuracy when the delay is calculated.

**Table 8 – Implementation of Block-Based and Path-Based Combined StTA**

```
Input: Circuit netlist, and the number of samples $N$.
Output: $CD$ (circuit delay) distribution

1: generate $N$ samples for the circuit (distributions according to Table 1)
2: for i=1 to $N$, do
3:     $CP_i$ = Critical Path extracted using block-based StTA
4:     $CD_j$ = Delay of $CP_i$ (using path-based StTA)
5: end for
6: return $CD = [CD_1, CD_2, \dots, CD_N]$
```

The block-based timing analyzer first abstracts a timing graph from the gate-level netlist, shown in Figure 17. The nodes of the timing graph represent primary inputs/outputs and gate input/output pins. Its edges represent the timing elements of the circuit, namely, the gate input-pin-output-pin delay and the interconnect delay. The weights on these edges is the delay of the corresponding timing elements, which is calculated using the unified gate-level MARSP models. After a forward traversal of the timing graph, the arrival times at primary outputs and D inputs of flip-flops are obtained.  By a backward traversal, the critical path can be extracted using the PERT algorithm [81]. For each Monte Carlo sample, the process of delay evaluation, forward traversal and backward traversal is repeated.

### *5.3.2 Experimental Results*

Table 9 presents the experimental results using ISCAS85 and ISCAS 89 benchmark circuits to verify the accuracy of our timing engine. On average, for ISCAS85 and ISCAS89 benchmarks, we achieve 0.70% and 1.45% error in estimating the mean and the standard deviation (SD) of the circuit delay distribution, respectively. In the experiments, the sampled values of the process variations in each transistor (channel length and threshold voltage) are subject to a Gaussian distribution with three sigma equal to the corners shown in Table 1. And the temperature and voltage values in each transistor are obtained from the thermal profile and IR-drop profile, respectively, determined by emulation. The block-based StTA is input pattern free. For the path-based StTA, the input vectors are generated via Automatic Test Pattern Generation (ATPG) to sensitize the extracted critical paths. Since simulating the whole circuit via SPICE is basically impossible, SPICE simulations are based on the critical paths extracted from the block-based StTA.

Table 10 shows the results of these six large benchmarks for which 200 samples were run. TA1 is a fabricated Floating-Point Unit Processor, and TA2 is a fabricated RISC microprocessor. Other benchmarks in Table 10 are from the IWLS benchmarks. On average, the error in estimating the mean and the SD of the circuit delay distribution is 1.45% and 3.75% respectively.

For IWLS benchmarks, the error is slightly larger than the error for ISCAS benchmarks. The errors have several possible sources. Firstly, large circuits have longer interconnect which might need higher-order *Y'(s)* and *H'(s)* models. Secondly, for large interconnect networks in large circuits, the first-order sensitivity model in (24) may generate some errors.

A quadratic Response Surface Model (RSM) delay model was also implemented and tested to give a comparison. The quadratic RSM first generates a quadratic regression model as follows:

$$D = d_0 + \sum a_i X_i + \sum b_i X_i^2 + \sum_{i \neq k} b_{i,k} X_i X_k \,. \tag{28}$$

*D* denotes gate delay, $X_i$ denotes variational PVT parameters, $d_0$ denotes the constant term, and $a_i$ and $b_i$ denote coefficients of first-order and second-order terms, respectively.

Average errors of RSM in comparison with SPICE are 15.2% and 19.5% for the mean and SD, respectively. The quadratic RSM model has a fixed number of operations, while the operations of the MARSP model depend on the subspace in which the input parameters fall. However, quadratic RSM models require more operations than the maximum number of operations of MARSP models when the number of parameters is larger than eighteen.

**Table 9 – Experimental results of the proposed StTA analyzer on ISCAS85 benchmarks.**

| Circuit Name | Number of samples | Number of critical paths run | Running Time | | | Error per sample (GTStTA) | | SPICE | | Path-based GTStTA | | $\left(\frac{\text{GTStTA} - \text{SPICE}}{\text{SPICE}}\right)\%$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | SPICE (s) | GTStTA (s) | RSM (s) | Mean (%) | SD | Mean (ps) | SD (ps) | Mean (ps) | SD (ps) | Mean (%) | SD (%) |
| c432 | 2000 | 55 | 199452 | 5451 | 8490 | 2.13% | 0.99% | 1434 | 216.7 | 1452 | 216.5 | 1.26% | -0.09% |
| c499 | 2000 | 102 | 241312 | 7160 | 11763 | -0.18% | 1.43% | 971 | 141.2 | 970.2 | 142.1 | -0.09% | 0.64% |
| c880 | 2000 | 80 | 303744 | 7472 | 12232 | -0.17% | 1.89% | 1324 | 205.6 | 1323 | 208.7 | -0.08% | 1.51% |
| c1355 | 2000 | 124 | 280376 | 8953 | 14991 | 3.36% | 2.26% | 960.6 | 144.1 | 991.8 | 142 | 3.25% | -1.46% |
| c1908 | 2000 | 118 | 651230 | 15021 | 24227 | 2.28% | 0.76% | 1601 | 242.7 | 1638 | 240.9 | 2.31% | -0.74% |
| c2670 | 100 | 23 | 4388 | 104 | 170 | 2.62% | 2.19% | 1263 | 171.9 | 1294 | 163.6 | 2.45% | -4.83% |
| c3540 | 100 | 24 | 6694 | 166 | 271 | 2.81% | 1.93% | 1772 | 239.9 | 1820 | 236.1 | 2.71% | -1.58% |
| c5315 | 100 | 26 | 7407 | 166 | 275 | 2.85% | 1.44% | 1589 | 227.9 | 1634 | 237.5 | 2.83% | 4.21% |
| c6288 | 100 | 30 | 74559 | 528 | 876 | 1.99% | 1.27% | 5766 | 389.6 | 5878 | 367.2 | 1.94% | -5.75% |
| c7552 | 100 | 28 | 27636 | 277 | 459 | 2.01% | 1.33% | 2665 | 291.8 | 2616 | 281.8 | -1.84% | -3.43% |
| avg. | - | - | 179679 | 4529 | 7375 | 1.97% | 1.55% | 1935 | 227.1 | 1962 | 223.6 | 1.47% | -1.15% |

**Table 10 – Experimental results of the proposed StTA analyzer on large circuits.**

| **Circuits** | **Cell Number** | **Time Saved** | **SPICE(ps)** | | **Our Engine(ps)** | |
|---|---|---|---|---|---|---|
| | | | **mean** | **S.D.** | **mean** | **S.D.** |
| TA1 | 21,725 | 99.1% | 645 | 79 | 652 | 82 |
| TA2 | 78,623 | 99.5% | 1431 | 130 | 1443 | 136 |

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| leon2 | 454,489 | 99.0% | 1960 | 412 | 2005 | 427 |
| netcard | 425,264 | 99.1% | 1842 | 393 | 1879 | 412 |
| leon3mp | 330,993 | 98.8% | 1211 | 368 | 1256 | 375 |
| leon3-avnet-3s1500 | 542,081 | 99.2% | 1981 | 432 | 2022 | 448 |

### *5.3.3 Complexity*

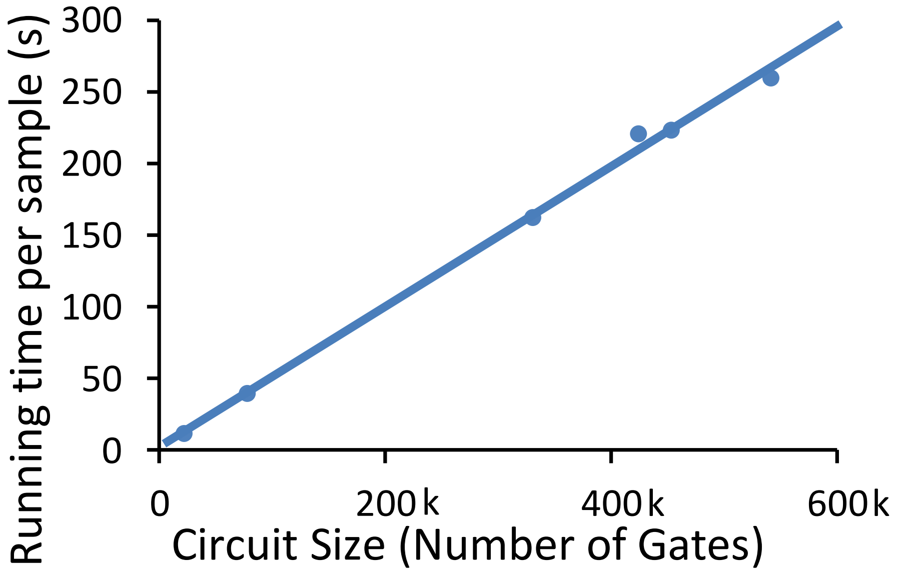


**Figure 23 – The runtime per sample versus the number of gates and interconnects in the proposed block-based and paths-based combined StTA.**

In this path-based and block-based combined framework, most of runtime was spent on the block-based part. The runtime per sample versus circuit size is shown in Figure 23. From the linear relationship between runtime per sample and circuit size, we can deduce that our framework would take ~8 hours to get the circuit-delay distribution of a large circuit with one million gates and interconnects by running 100 Monte Carlo samples. Because this work is based on the Monte Carlo method, the runtime cost may be large if more samples are required or a larger circuit is simulated. One solution for this issue is using a Graphic Processor Unit (GPUs) to facilitate parallel computations as different samples can be independently processed.

# 6. AGING SIMULATION FOR STATE-OF-ART MICROPROCESSORS

This section introduces the aging simulator framework which is shown in Figure 1. The framework firstly extracts the activity, IR-drop, and temperature profiles of a microprocessor while running benchmarks. The extracted profiles are then used to calculate the degradation of device parameters in each transistor. Next, the obtained aging profiles, along with the PVT profiles, are fed into the PVT-aging timing engine to achieve comprehensive timing analysis when BTI, HCI, and GOBD happen simultaneously.

## 6.1 Extraction of the Stress, Thermal, IR-Drop and Process profiles

The wearout mechanisms being studied are activity, supply voltage (*VDD*) and temperature dependent. The degradation of system performance is also directly dependent on the thermal and the IR-drop profile, because temperature and *VDD* directly impact circuit timing. In this work, FPGA emulation is used to simulate the microprocessors, which provides an efficient way to acquire electrical, thermal and IR-drop profiles for any digital system for use in system-level reliability analysis.

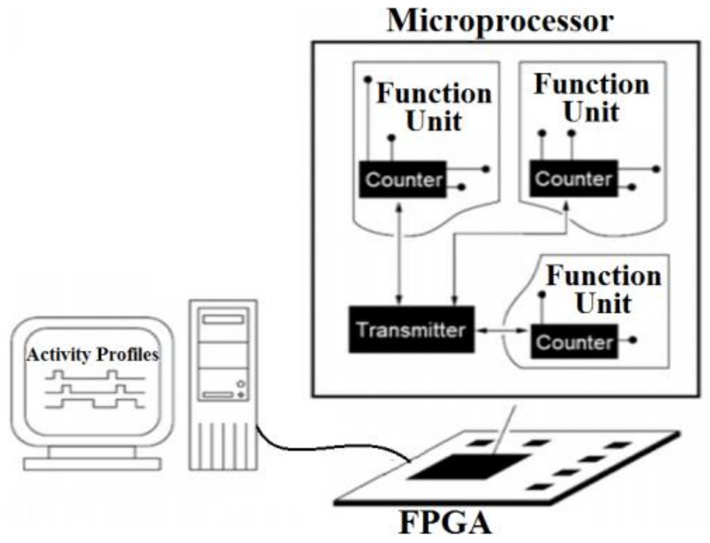
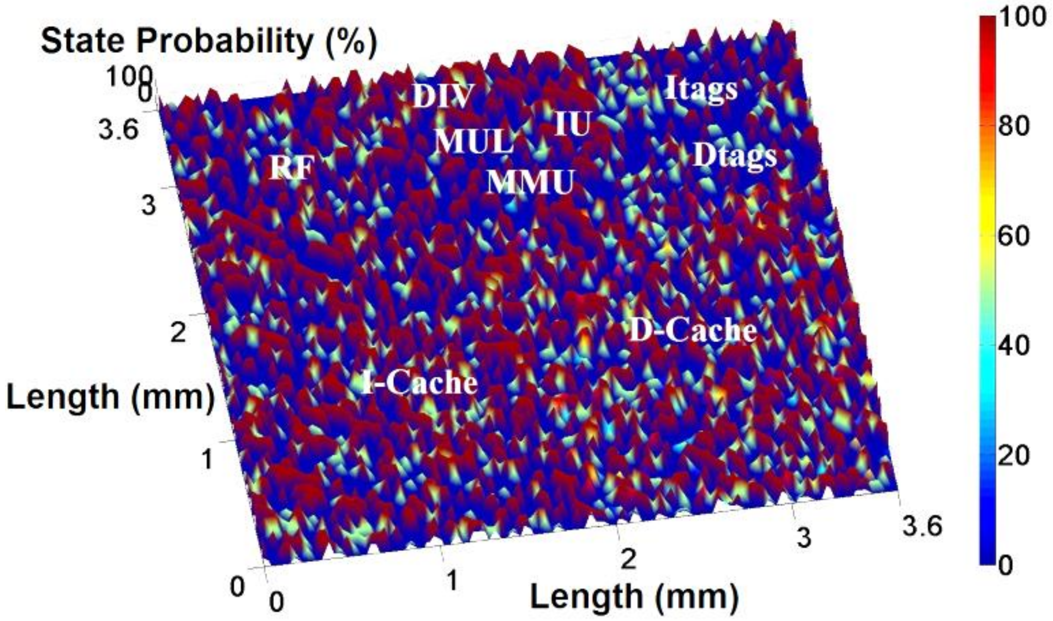


**Figure 24 – The system used to collect activity profile of the microprocessor contains an FPGA board that implements the microprocessor system and exports data on the activity profile to a PC.**

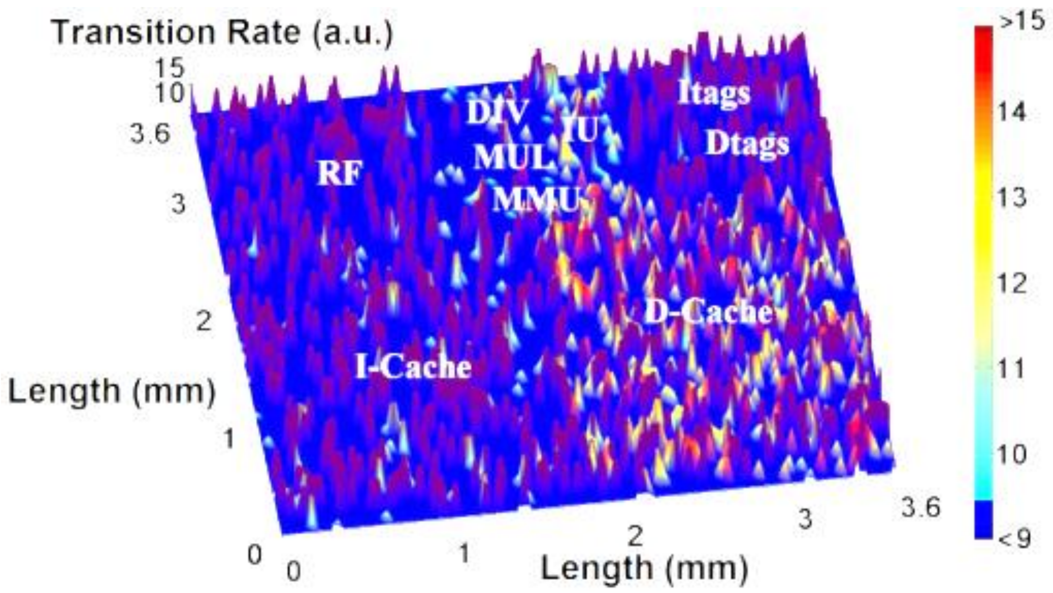
The microprocessor under study is LEON3 microprocessor [82]. The RTL has been synthesized and loaded to the FPGA with the Xilinx ISE (Integrated Software Environment) [83]. Once the FPGA is programmed, the activity can be collected by placing counters at the I/O ports to track the state probabilities and the toggle rates of the ports during application runtime, as illustrated in Figure 24.

Since the I/O ports for each unit can be found on the top of each module, the counters are attached to the ports automatically with a scripting language. The activity transportation unit is inserted into the RTL automatically as well. The complexity of this RTL revision process is O(*n*), where *n* is the number of the number of I/O ports. Since the complexity is linear, the RTL revision process is scalable and can be implemented for large systems. Our current work focuses on implementing a microprocessor on a single FPGA, so the revised RTL is executable as long as the FPGA has enough resources (gates) to support large systems. A set of standard benchmarks [84] were used as the applications for analysis.

The activities and state probabilities are only captured at block I/Os, because monitoring all internal nets would require too many resources. After capturing the I/O activities and state probabilities, they are linked to the netlist for activity and state propagation to each net in the design using PrimeTime [80]. The propagation is a function of the design's logic. It is probabilistic, since the exact signals that produced the I/O activities and state probabilities are unknown. Therefore, the resulting activities and state probabilities at internal nodes may have errors with respect to the exact signal probabilities and activities. The result is a complete stress/transition probability profile of the internal nodes of the microprocessor under study. This activity and state propagation component is done in software on a block-by-block basis. Thus, we have the probability of a transition occurring in any cell and the probability at each state, i.e., the probability at logic “1”.



(a)

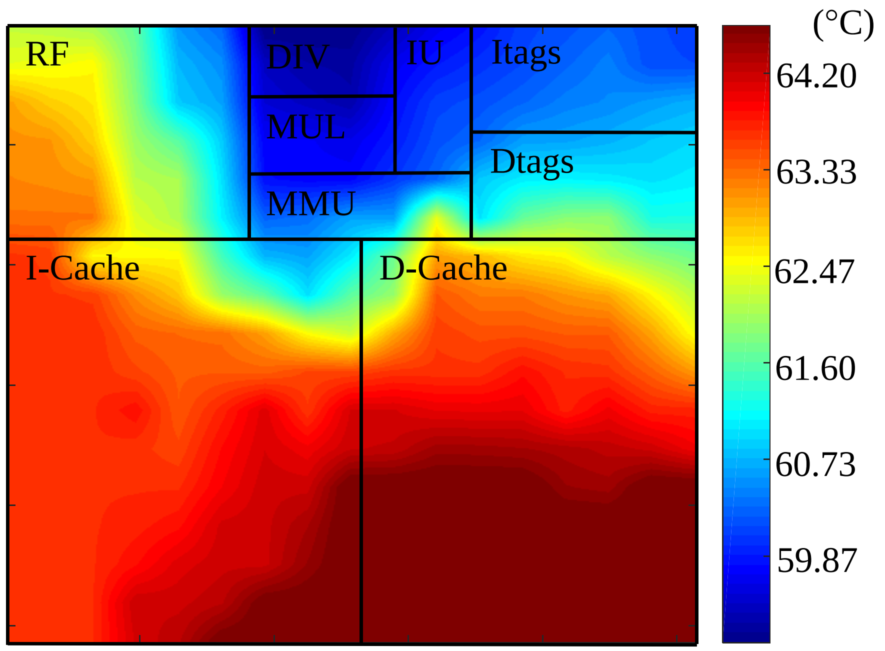


(b)

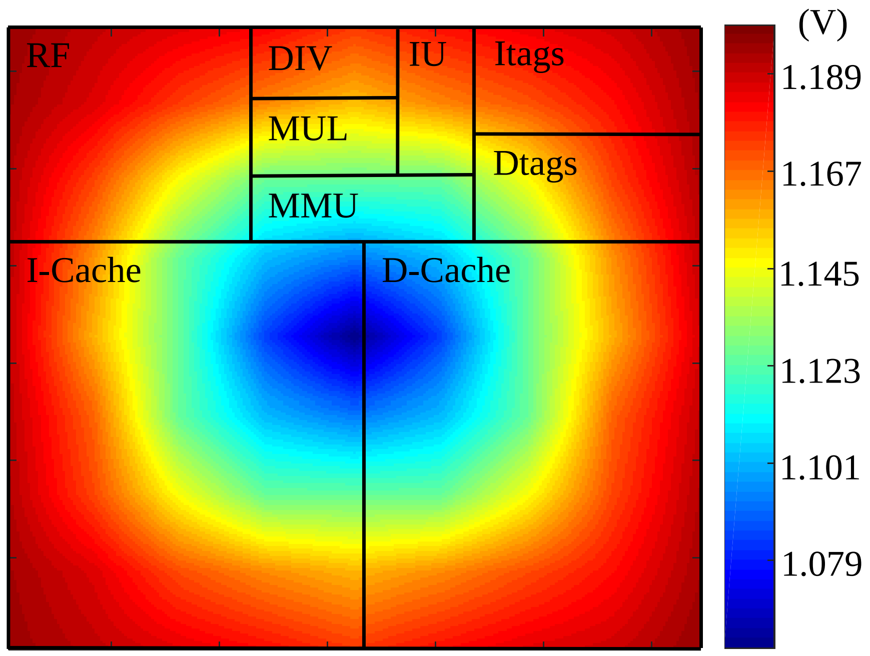
**Figure 25 – (a) The distributions of the DC stress probability and (b) the transition rate (toggle rate) for the LEON3 microprocessor while running a standard benchmark.**

Figure 25 (a) and (b) show the distributions of the stress probability and the transition rate, respectively, when the microprocessor is running a set of standard benchmarks. It can be seen that the distribution of the stress state probability and transition rate is different for each block. For example, Itags and Dtags have a low stress state probability and a high transition rate, while DIV, MUL, IU and MMU have a high stress state probability and a low transition rate.

The propagation of transition rate and state probability was verified by comparing the exact transition numbers and state periods of randomly selected nets from the microprocessor with the ones calculated by propagations. The results in [13] shows that the percent errors for more than 90% of the selected samples are less than 10% for transition rate and more than 80% of the selected samples have errors that are less than 15% for state probability. The high errors are mostly from the nets in deeper locations of the circuit that are far from the I/Os. Since errors are cumulative, activity propagation to deeper stages leads to a larger difference between the real transition rate/state probability and the calculated ones.



(a)



(b)

**Figure 26 – (a) The temperature and (b)VDD distribution of the LEON3 microprocessor running a standard benchmark.**

The temperature variation throughout the microprocessor is also taken into account when modelling different wearout mechanisms. The netlist was used for layout generation using Cadence Encounter [77] and a commercial PDK [85]. The RC information extracted from the layout via [78], together with the net activities, was used for the extraction of the power profile and the consequent thermal profile, through the power simulator [80] and the thermal simulator [86], respectively. The net activities and layout are also used to determine the IR-drop profile throughout the microprocessor via [77]. Figure 26 shows the distributions of the temperature and the supply voltage in Figure 26(a) and Figure 26(b), respectively, when LEON3 is running a standard benchmark.

## 6.2 Proposed PVT-Aware Aging Simulator

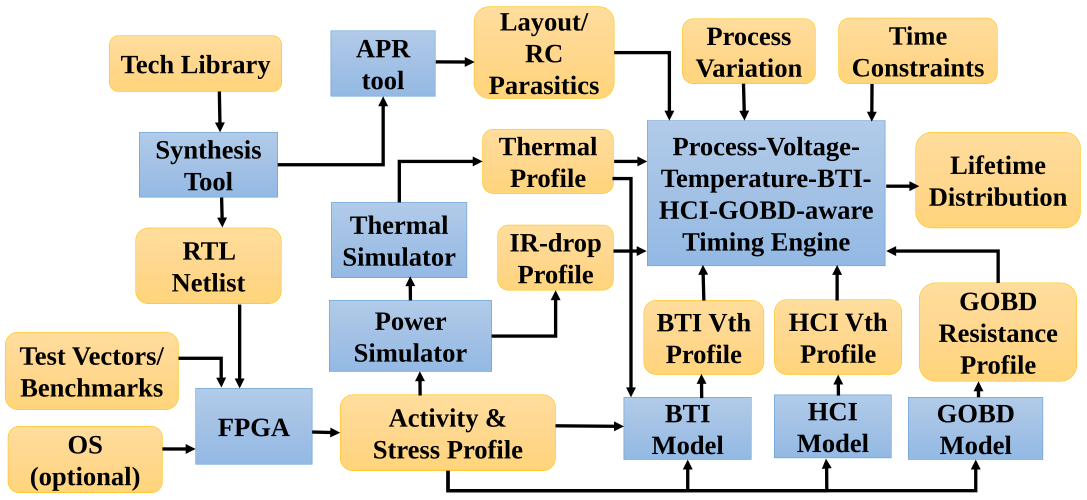


**Figure 27 – The flow chart of proposed aging simulator.**

The overall flow chart of the proposed PVT-aware aging simulator is shown in Figure 27. The core of this aging simulator is the PVT-aging aware timing engine which is constructed based on the StTA framework in Section 5.3. The timing engine is fed by a set of profiles, including process variation, thermal, IR-drop, RC parasitics, BTI, HCI and GOBD. The generation of these profiles was discussed in Section 6.1.

**Algorithm: Statistical Timing Engine**

```
Input: circuit netlist, RC parasitics (.spef), process-variation spatial correlation profile, BTI
$\Delta Vth$ profile, HCI $\Delta Vth$ profile, GOBD $R_{G2S}/R_{G2D}$ profile, thermal profile, IR-drop profile
Output: Circuit-delay distribution

Block-based timing analyzer
    1. generate N Monte Carlo samples for the circuits according to process-variation spatial
correlation profile.
    2. abstract timing graph from circuit netlist, and apply ($\Delta L$, $\Delta Vth$, $R_{G2S}$, $R_{G2D}$)
    3. for i=1 to N   //for each Monte Carlo sample
        3.1. apply to each gate the corresponding values of $\Delta L$, $\Delta Vth$, $R_{G2S}$, $R_{G2D}$, T and etc.
        3.2. do forward traversal to evaluate the edges (gate and wire delay) and propagate the
             arrival time to Primary Output/DFF
        3.3. do backward traversal to extract the first 3 critical paths using PERT algorithm.
    4. end for
    5. get the critical-path collection: $\boldsymbol{P} = \{CP_1, CP_2, \dots, CP_m\}$
    // m is up to 3*N, because some paths are repetitive
Path-based timing analyzer
    6. for i=1 to N   //for each Monte Carlo sample
        6.1. calculate the path delays for each critical path in $\boldsymbol{P}$ as $\{PD^{i}_{1}, PD^{i}_{2}, \dots, PD^{i}_{m}\}$
        6.2. get the maximum path delay $D^{i}=\max\{PD^{i}_{1}, PD^{i}_{2}, \dots, PD^{i}_{m}\}$
    7. end for
    8. circuit-delay distribution from $\{D^{1},D^{2},\dots,D^{N}\}$
```

**Figure 28 – The framework of the PVT-aging-aware timing engine.**

Figure 28 presents the algorithm of the timing engine. As introduced in Chapter 5, both the block-based and path-based timing analyzer are based on Monte Carlo analysis. The electrical stress, thermal, IR-drop, process profiles and RC parasitics, together with the device-level wearout models, generate all the parameters for each gate which are needed to calculate the gate delays. All the information is fed to the PVT-aging aware timing engine as shown in Figure 28 for the analysis of circuit timing degradation.

generates the delay degradation at a variety of stress times. Figure 29 shows the delay distribution of a RISC microprocessor at different stress times.

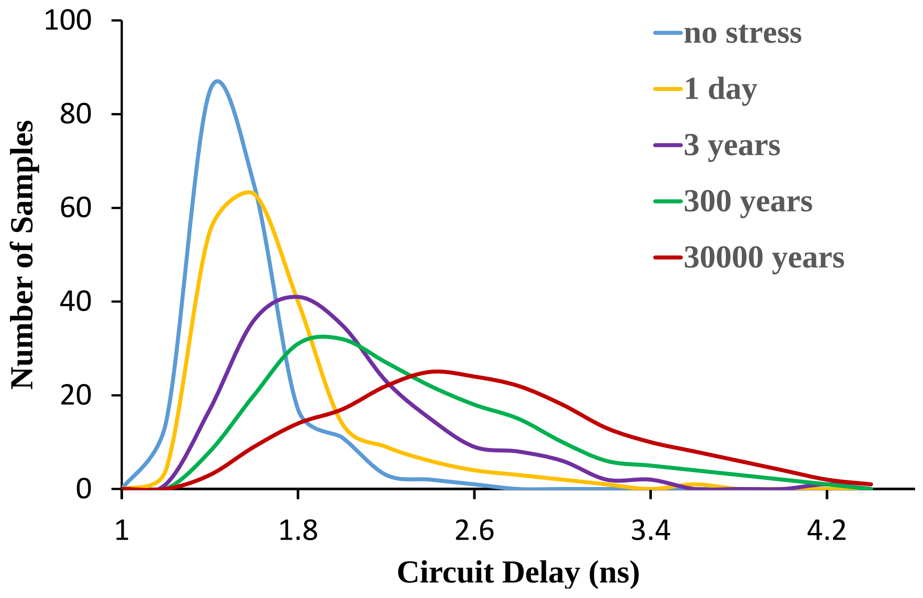


**Figure 29 – The delay distributions of a RISC microprocessor due to the combined effect of PVT and aging under a variety of stress times.**

The system-level aging simulator is constructed by analyzing the performance (circuit delay) degradation at different stress times. When the circuit delay degrades beyond the clock period, the system is fails and the lifetime is thus obtained. Figure 30 shows a comparison of the lifetime distributions due to BTI, HCI, GOBD individually and simultaneously. It can be seen that GOBD is generally the most critical mechanism of the three, but for some samples BTI is dominant. Besides, when all of these three mechanisms happen simultaneously, the lifetime is shorter than the lifetimes due to analyzing each mechanism alone. It is concluded that BTI and HCI also have some effect on circuit lifetimes even if they are not dominant.

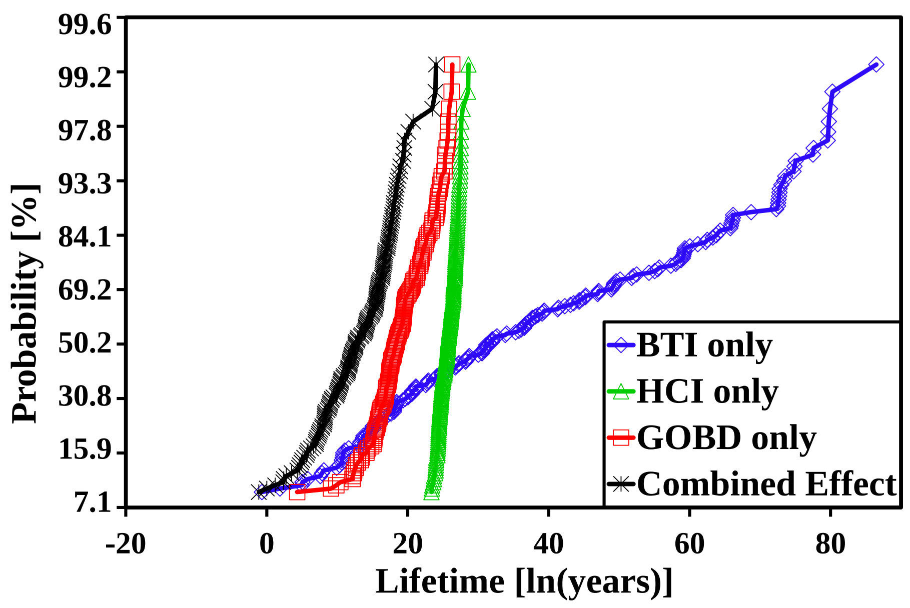


**Figure 30 – The lifetime distributions of the LEON3 microprocessor due to BTI, HCI, GOBD and the combined effect.**

## 6.3 Lifetime Analysis Considering Realistic Workloads

This work not only accounts for activity and temperature, but also accounts for the fact that processors are not in operation at all times. Realistic use conditions include operation modes, standby, and periods of time when the processor is turned off, as illustrated in Figure 31. This research takes these use scenarios into account. These use scenarios determine the two key parameters that are needed to determine the stress and temperature for all wearout mechanisms, the logic state probability for each net, which determines the duty cycle over the full lifetime of the microprocessor, and the activity of each node, which determiens the average current and temperature during each state of operation.

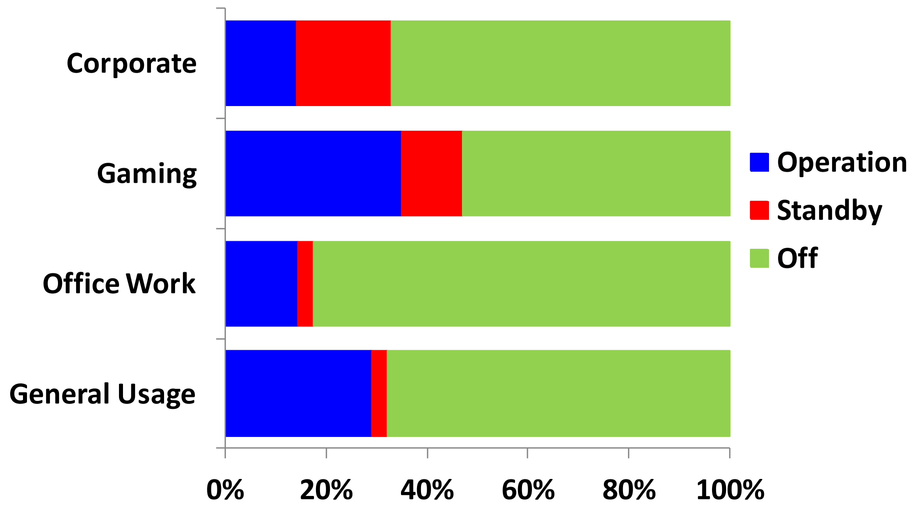


**Figure 31 – The use scenarios provided by Intel are shown [87].**

For BTI, by weighting the lifetimes of operation, standby and off modes in accordance with Figure 31, the lifetimes of the microprocessor under study are estimated based on different operating frequencies for different use scenarios, as shown in Figure 32. The different operating modes impact the values of $t_{stress}$ and $t_{rec}$in equation (2). For example, during the "off" state, $t_{rec}$ is increased. The results clearly indicate that the estimated system lifetimes decrease as the system frequency increases, and gaming has the shortest lifetime. The lifetimes converge to zero at the upper and lower limits of the initial operating frequency.

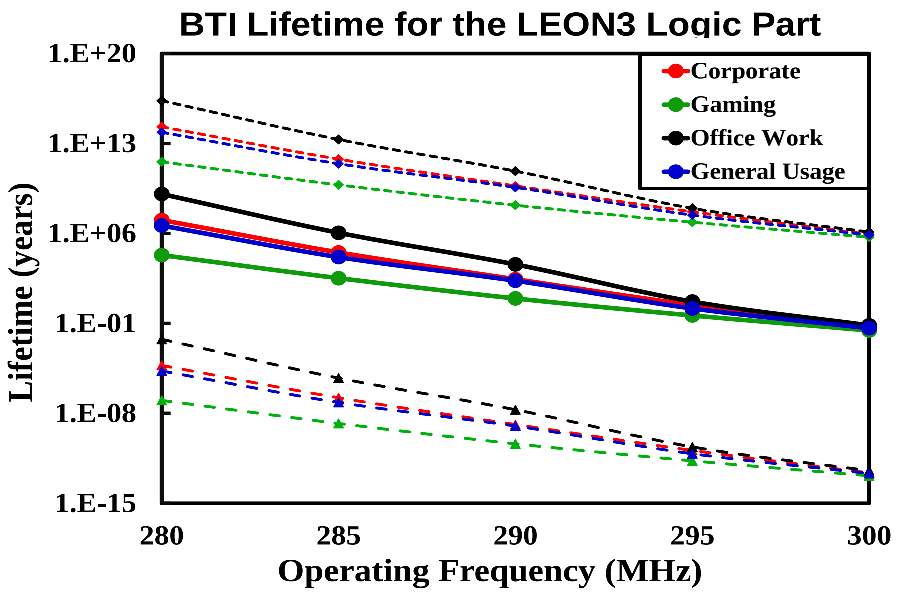


**Figure 32 – The estimated lifetimes of the LEON3 microprocessor due to BTI for different use scenarios and different system frequencies. Dotted lines show the boundaries when considering process variation.**

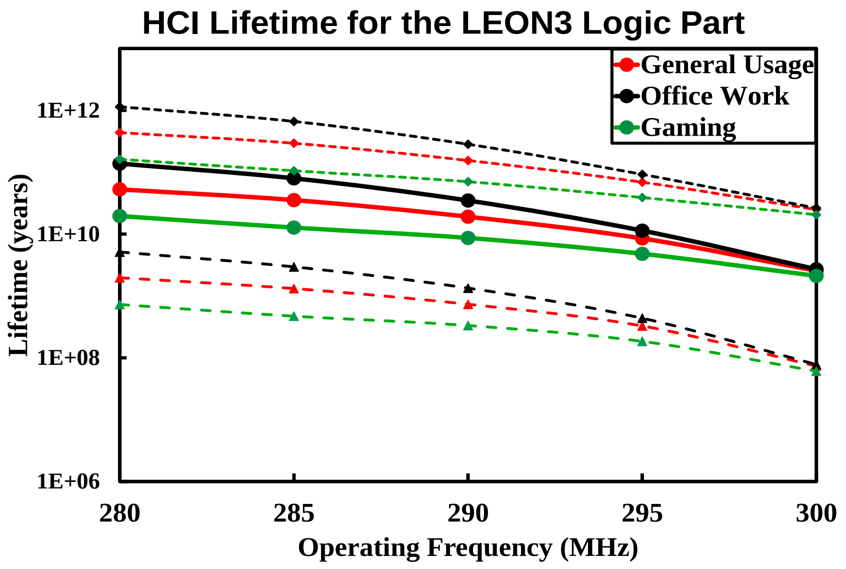


**Figure 33 – The estimated lifetimes of the LEON3 microprocessor due to HCI for different use scenarios and different system frequencies. Dotted lines show the boundaries when considering process variation.**

The microprocessor system lifetimes for different operating frequencies and different use scenarios were also investigated under HCI. The system lifetimes estimated by the proposed methodology are shown in Figure 33. Similar to BTI, the microprocessor lifetimes estimated by our methodology decrease as the system frequency increases, and gaming has the shortest lifetime. Again, the lifetimes converge to zero at the initial operating frequency limits.

Besides LEON3, a 32-bit RISC microprocessor which includes around 73k gates was also studied. The area for the 32-bit RISC microprocessor is around 7 mm$^2$ and the power consumption is around 0.1W~0.2W.

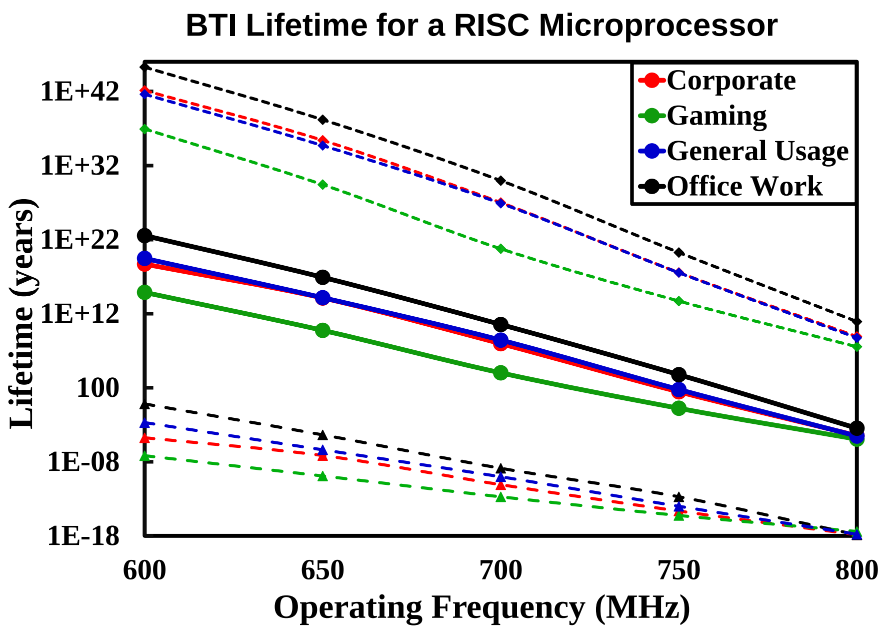


**Figure 34 – The estimated lifetimes of the RISC microprocessor due to BTI for different benchmarks and different system frequencies. Dotted lines show the boundaries when considering process variations.**

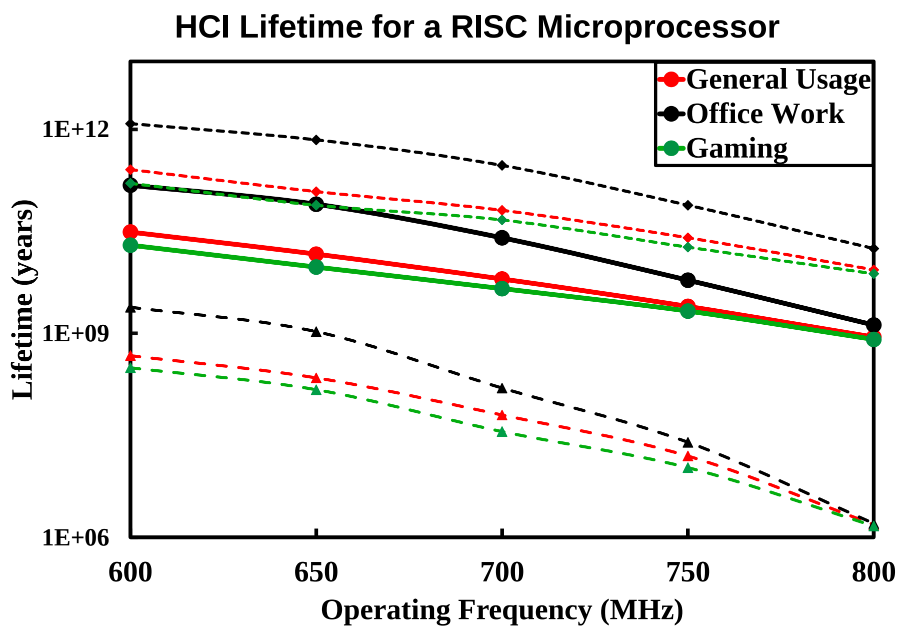


**Figure 35 – The estimated lifetimes of the RISC microprocessor due to HCI for different benchmarks and different system frequencies. Dotted lines show the boundaries when considering process variations.**

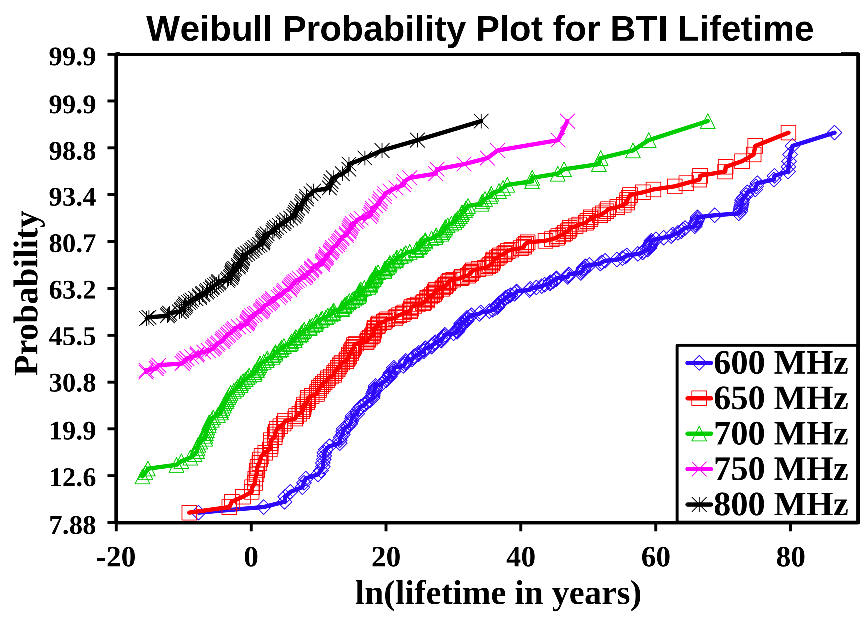


**Figure 36 – The statistical lifetime distribution of the RISC microprocessor due to BTI for different system frequencies, for the gaming use scenario.**

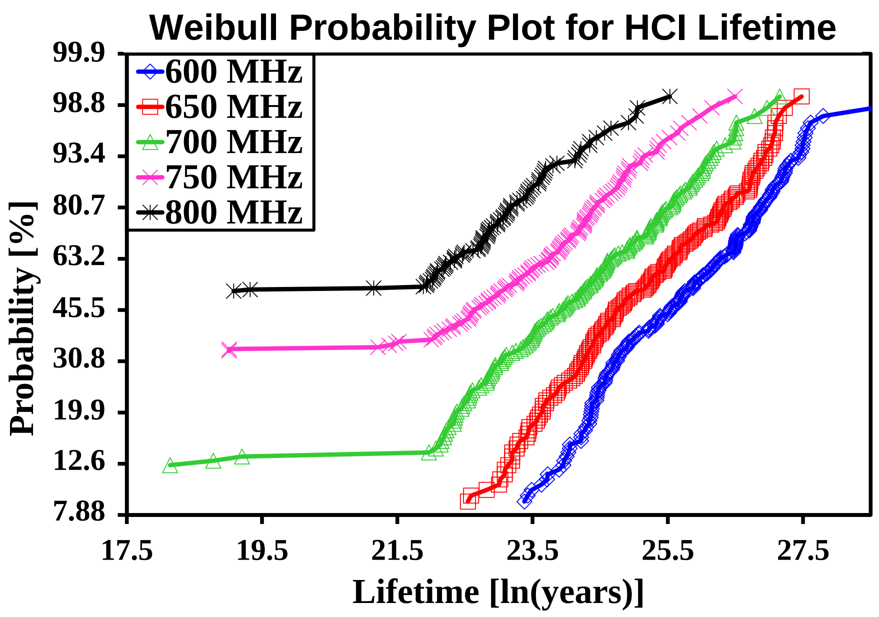


**Figure 37 – The statistical lifetime distribution of the RISC microprocessor due to HCI for different system frequencies, for the gaming use scenario.**

Figure 34 and Figure 35 show the estimated lifetime due to BTI and HCI, respectively, for the RISC microprocessor, for different operating frequencies. As with the LEON3, the estimated system lifetimes decrease as the system frequency increases. The convergence to zero lifetime at initial operating frequency limits does not show in the graphs because these limits are outside the frequency range in the graphs.

The confidence bounds in the above figures were based on simulated distributions of delay. Figure 36 and Figure 37 show the cumulative probability plots of Weibull distributions for BTI and HCI, respectively. By fitting the cumulative probability plot using a Weibull distribution, the shape parameter, $\beta$, and the characteristic lifetime parameter, $\eta$, of the Weibull distribution are obtained. Given $\eta$ and $\beta$, the time-to-failure is computed for any probability point, *P*. Given a confidence level, say, 90%, we compute the corresponding time-to-failure points for the appropriate probability points, i.e., $P=0.05$ and

*P=0.95.* These corresponding time-to-failure points are the confidence bound limits in Figure 34 and Figure 35.

## 6.4 APPLICATION: Finding Optimum Operating Voltages Using Proposed Aging Simulator

In this subsection, the proposed aging simulator was applied to a case study to solve a practical problem: find the optimum operating voltage of a system so that the best performance-reliability tradeoff can be achieved.

To make this application more realistic, the hard breakdown (HBD) of GOBD is included. In previous sections, the soft breakdown (SBD) of GOBD is discussed and modelled by inserting a gate-to-source resistance ($R_{G2S}$) or gate-to-drain resistance ($R_{G2D}$) in a target gate in order to create the current leakage path in the circuit. HBD happens after the soft breakdown, and it is well understood that HBD happens when the gate dielectric layer abruptly loses its insulating properties. HBD can be detected as a large jump in the current vs. time curve, while in soft breakdown (SBD), the leakage current in the gate dielectric slightly increases with time and the gate dielectric still remains its insulation property. The method to analyze the hard failure rate is from [8]. The permanent HBD of GOBD is referred as hard failure hereafter, while soft failure refers to the failures caused by the timing violations due to BTI, HCI and GOBD soft breakdown.

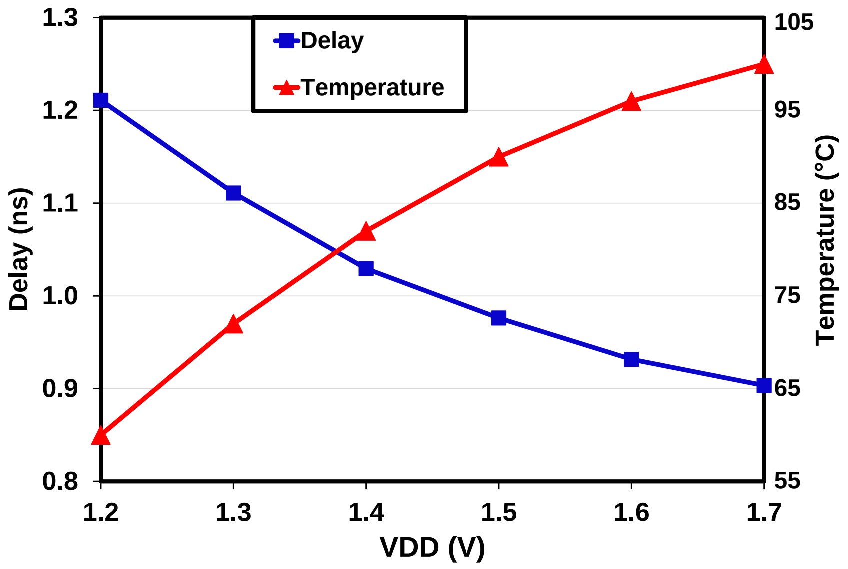


**Figure 38 – Chip temperatures while LEON3 is running at different voltages and circuit delays at each corresponding VDD/Temperature condition.**

The test case is the LEON3 microprocessor where different operating voltages are applied. The power consumption of LEON3 while running a standard benchmark in operation mode is simulated and fed to the thermal simulator to get the operating temperatures. In the thermal simulator, the ambient temperature is set to room temperature (27°C). The operating temperatures under different voltages are shown in Figure 38, along with the average circuit delays under the corresponding operating conditions (voltage and temperature). As expected, higher voltage brings higher temperatures. Moreover, the circuit delay decreases under the conditions of high voltage and high temperature, which means the effect of high voltage on decreasing delay is stronger than the effect of high temperature on increasing delay.

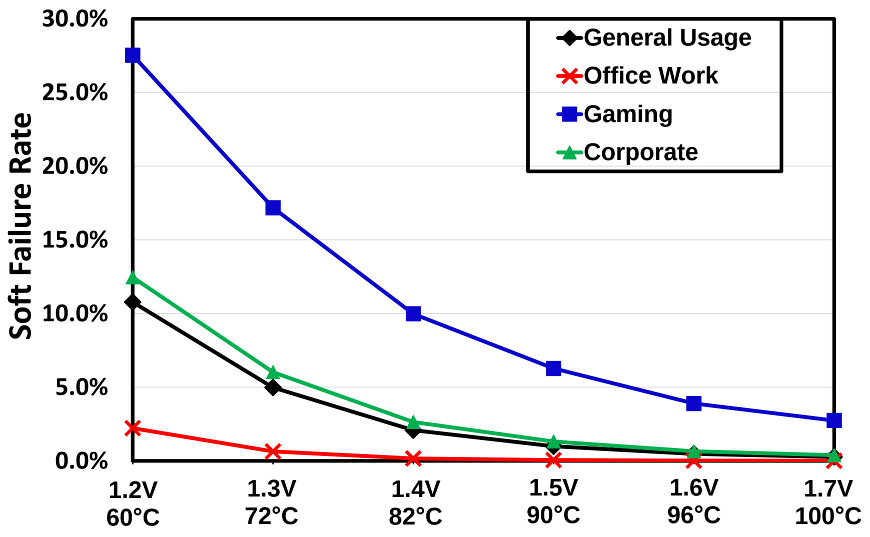


**Figure 39 – The soft failure rate in 10 years due to timing violations for different operating voltages and for different usage scenarios when ambient temperature is set to 27 ℃.**

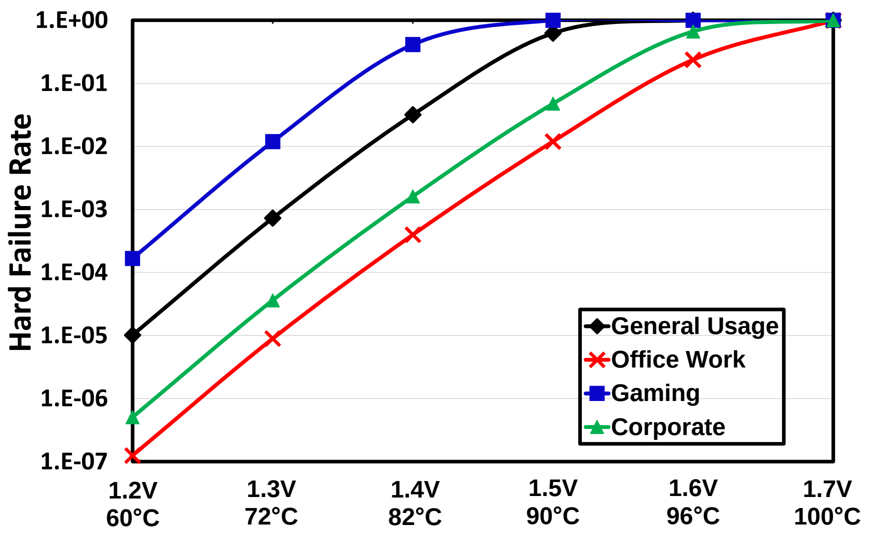


**Figure 40 – The hard failure rate in 10 years due to the permanent GOBD hard breakdown are shown for different operating voltages and for different usage scenarios when ambient temperature is set to 27 ℃.**

Then, the proposed aging simulator is used to get the lifetimes for different operating voltages. Figure 39 shows the failure rate in 10 years for each use scenario. The failure rate is calculated according to the timing failures, and we refer to it as the soft failure rate to distinguish it from permanent catastrophic failures. The soft failure rate decreases as the voltage increases because larger timing slack is observed at higher voltages. On the other hand, however, higher voltage and higher temperature increase the probability of suffering from catastrophic GOBD hard breakdown where the gate dielectric permanently breaks down and leads to a current runaway. The hard failure rate in 10 years under different voltages is shown in Figure 40, where it's seen that the failure rate dramatically increases at higher voltages and higher temperatures. In the worst case (VDD=1.7V, Temperature = 100°C), the hard failure rate in 10 years is close to 100%.

Combining the soft failure rate due to timing failures and the hard failure rate due to GOBD hard breakdown, we get the overall failure rate in 10 years for different usage scenarios in Figure 41. For the 'Gaming' scenario, the optimum voltage to achieve the best lifetime is 1.3V, while for the other three scenarios the optimum voltage is 1.4V. From the results, it's seen that at lower voltage, the soft failure is dominant while the hard failure is dominant at higher voltages.

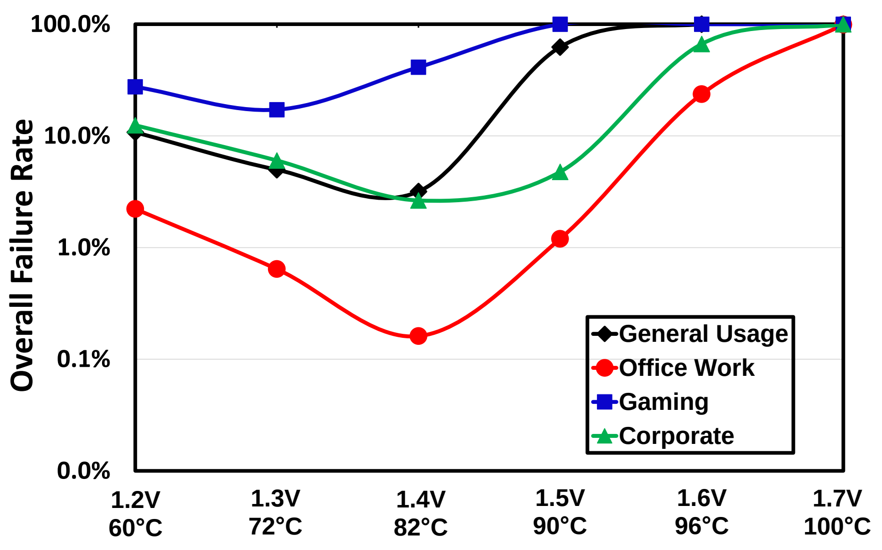


**Figure 41 – The overall failure rate in 10 years due to both soft failure and hard failure is shown for different operating voltages and for different usage scenarios when ambient temperature is set to 27 ℃.**

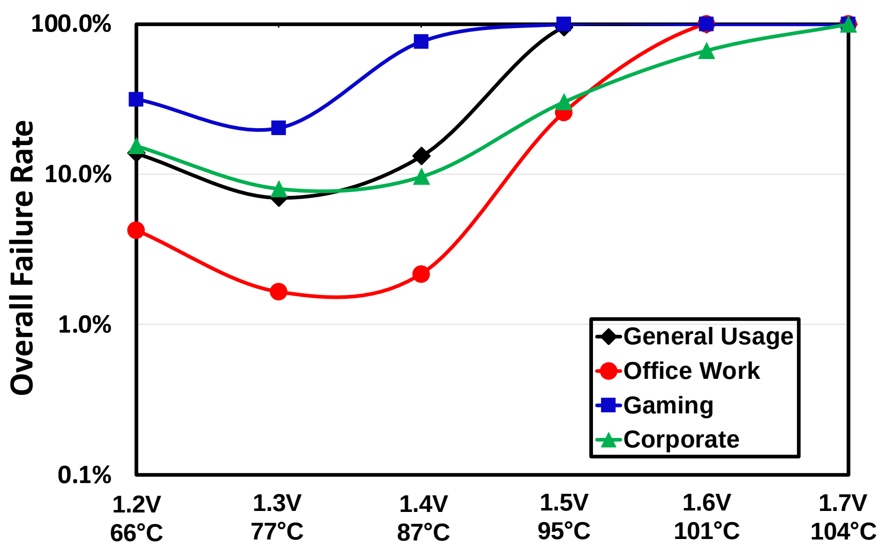


**Figure 42 – The overall failure rate in 10 years due to both soft failure and hard failure is shown for different operating voltages and for different usage scenarios when the ambient temperature is set to 40 ℃.**

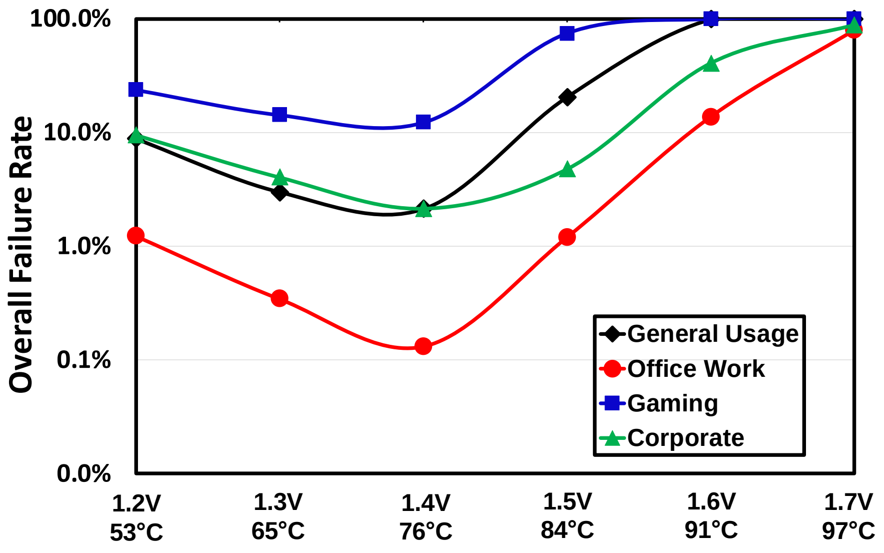


**Figure 43 – The overall failure rate in 10 years due to both soft failure and hard failure is shown for different operating voltages and for different usage scenarios when the ambient temperature is set to 15 ℃.**

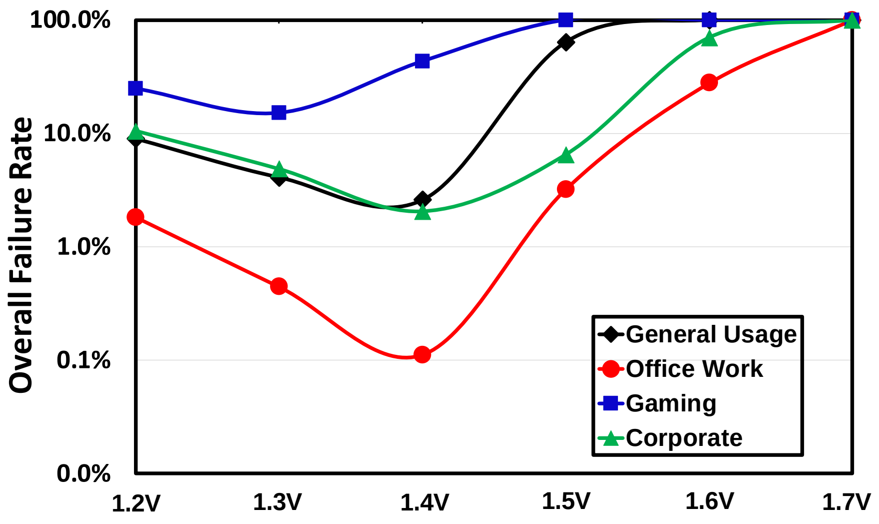


**Figure 44 – The overall failure rate in 10 years due to both soft failure and hard failure is shown for different operating voltages and for different usage scenarios when the ambient temperature is set to 27 ℃ and the IR-drop effect is ignored.**

To investigate the impact of ambient temperature on circuit reliability, we also vary the ambient temperature. The chip temperature and the overall failure rate under different supply voltages are shown in Figure 42 and Figure 43, for ambient temperatures equal to 40°C and 15°C, respectively. Experimental results see the trend that the optimum voltage shifts to the left (lower) when the ambient temperature increases. More specifically, for the use scenarios of Office Work, Corporate and General Usage, the optimum voltage shifts from 1.4V to 1.3V when ambient temperature increases from 27°C to 40°C. This is because the higher ambient temperature brings up the chip temperature and ultimately makes hard breakdown failure more severe when the supply voltage is higher than 1.4V. When the ambient temperature decreases from 27°C to 15°C, the optimum voltage shifts from 1.3V to 1.4V for the Gaming scenario. Lower temperature decreases the hard breakdown failure rate, and the soft breakdown dominates at low supply voltages. Figure 44 is shown to investigate the impact of the IR-drop effect. Results show that ignoring the IR-drop effect would underestimate the soft failure rate and overestimate hard failure rate. However, the optimum operating voltages are not affected.

# 7. SRAM LIFETIME ANALYSIS FOR DIFFERENT CACHE CONFIGURATIONS

This research work also includes the lifetime analysis of SRAMs, particularly the caches within in state-of-art microprocessors.

The majority of transistors in a modern microprocessor are used to implement Static Random Access Memories (SRAM). Therefore, it is important to analyze the reliability of SRAM blocks. The first-level (L1) data cache is a prime candidate, since it experiences frequent read and write operations, yet stores data for a significant amounts of time. Besides, cache efficiency is also very critical for system performance. Much prior work has focused on the cache architectures needed to achieve high hit rates. However, the question of how the reliability of the cache is affected when higher performance is achieved remains unanswered. In this work, the reliability (failure probability) of the L1 data cache in a state-of-art microprocessor is investigated for different design configurations: associativity, cache line size, cache size, and replacement algorithm. By analyzing the reliability and performances for different cache designs, we provide insight on the performance-reliability tradeoff in cache system design. The effect of error correcting codes (ECCs) is also considered, as ECCs enhance cache lifetime.

SRAMs are highly sensitive to BTI-induced transistor-strength mismatch [6], [10], [12], [88]–[90]. SRAM stability is analyzed in [91]–[93] by assuming two ideal stress conditions, static stress and alternating stress. To consider realistic stress conditions in

SRAM cells, [94]–[97] estimate the SRAM degradation due to BTI based on a customer usage workload profile.

SRAM stability due to HCI is less studied in prior research because BTI is usually dominant due to its frequency independence. However, nowadays, since chips are running at higher frequencies, HCI is becoming an issue [98],[99]. In [100],[101], the impact of HCI on SRAM cell stability is analyzed, and [100] compares the simulation results with silicon experimental results.

In this work, the stability degradations of SRAM cells due to BTI and HCI are studied under various stress conditions, and failure probability of the L1 data cache within a state-of-art microprocessor is calculated by taking into account realistic workloads when a set of benchmarks are running on the microprocessor. The work is new in the following ways.

- Cache reliability is analyzed for different cache configurations relating to: associativity, cache line size, cache size, and the replacement algorithm.

- The effect of Error Correcting Codes (ECCs) is studied in the failure-probability analysis of the data cache.

- The impact of process variations is included using Monte Carlo simulations, and the lifetime distribution of SRAM cells with different stress conditions is extracted. The distribution is parameterized, where we found in this work that the Log-Normal distribution is the best fit.

- This work takes into account real temperature and IR-drop profiles of a microprocessor when analyzing cache reliability by using an FPGA-based aging assessment framework.

- This work considers the combined effect of BTI and HCI, rather than studying them individually. We have found that HCI can mitigate SRAM stability degradation due to BTI for some of the stability metrics. Therefore, studying BTI and HCI separately leads to an overestimation of SRAM stability degradation.

To the best of our knowledge, this is the first study on the reliability-performance tradeoff for caches with different cache configurations. However, much prior work has focused on mitigating cache degradation in the presence of BTI. [102]–[105] have proposed to reduce the impact BTI aging by balancing the amount of time that ‘0’ and ‘1’ values are stored in the cells. In [106], the authors have proposed to exploit microarchitectural redundancy to extend cache lifetimes in the presence of BTI. Other methods reduce BTI aging by dealing with the parameters that have a strong impact on BTI, such as the temperature and supply voltage [107],[108]. In addition, [105] has proposed an indexing scheme to balance the usage of entries in the cache to combat HCI. In [109], the authors reduce HCI aging by shifting the incoming data to spread the bit flips evenly among the memory cells.

## 7.1 SRAM Stability

Each cache bit is implemented with an SRAM cell consisting of 6 transistors (6T), as shown in Figure 45. The labeled transistors form an inverter loop that holds the stored logic value, whereas the remaining pass transistors controlled by the wordline (WL) signal allow read and write operations to the cell through the bitline (BL) and its complement ($\overline{\text{BL}}$).

In a 6T cell, HCI affects all the transistors on a write if the logic value flips. On the other hand, when the cell is stable and storing a ‘0’, the PMOS transistor $T_{P1}$ and the NMOS transistor $T_{N2}$ are under stress, and they suffer from NBTI and PBTI, respectively. On the contrary, when the cell stores a ‘1’, transistors $T_{P2}$ and $T_{N1}$ are affected by NBTI and PBTI, respectively. Note that the wearout effects induced by each type of duty cycle are complementary, meaning that, for a given duty cycle, the pair of transistors not under stress are partially under recovery from BTI degradation. Overall, the four transistors of the inverter loop are continuously aging regardless of whether the cell stores ‘0’ or ‘1’, or is transitioning. This fact makes such transistors particularly sensitive to wearout [102]. Note that the NMOS pass transistors just age from BTI when the SRAM cell is being accessed, making them much less sensitive to aging than the inverter loop transistors. Thus this work focuses on the wearout of the inverter loop transistors.

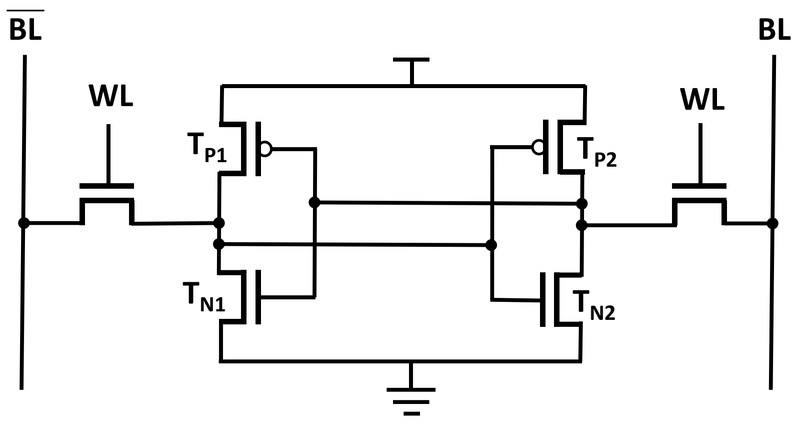
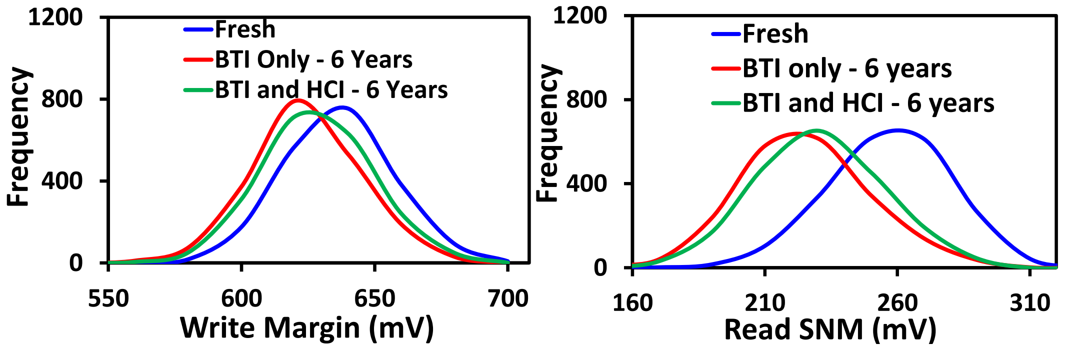


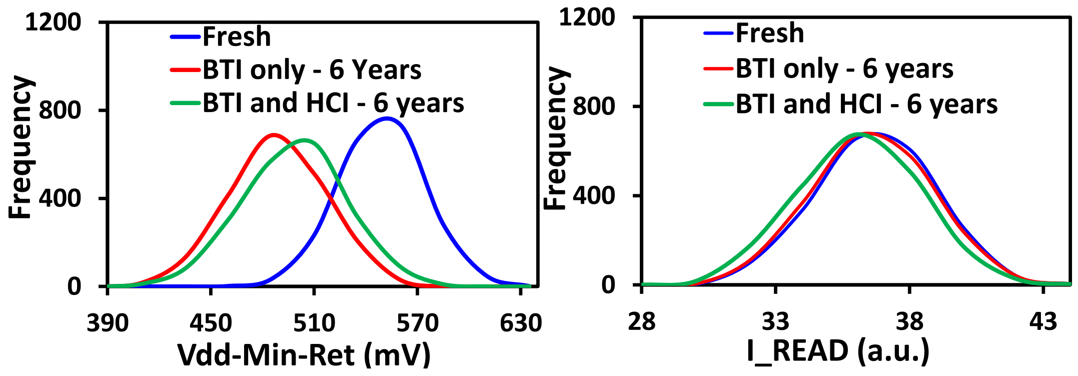
**Figure 45 – A typical 6T SRAM cell.**

## 7.2 SRAM Lifetime Characterizations

SRAMs are characterized with several performance metrics. These include the read and retention static noise margins (SNMs), the write margin, the read current ($I_{READ}$), and the minimum retention voltage (Vdd-min-ret). The static noise margins are defined as the minimum DC noise voltage necessary to change the state of an SRAM cell. The read SNM is measured with the access transistors turned on, while the access transistors are off for the retention SNM. The write margin is the minimum voltage needed to flip the state of the cell, with the access transistors turned on. Vdd-min is the minimum voltage in which the SRAM retains its state. Finally, the read current, which is inversely proportional to access time, is the current flow through pull-down devices when performing a read operation. The stability margins are extracted by fitting squares between the static noise margin (SNM) curves and observing the diagonal length of the smaller of the two squares [112]. When any of these four performance metrics degrade to a predefined threshold, the SRAM cell is said to have failed, and thus the lifetime of the cell is obtained. Variations of two process parameters have been included: the channel length and the threshold voltage of each transistor in the SRAM cell, both of which are subject to a Gaussian distribution with three sigma equal to 30% of the corresponding nominal value. Monte Carlo SPICE simulations were implemented to obtain 2000 samples.



(a) (b)



(c) (d)

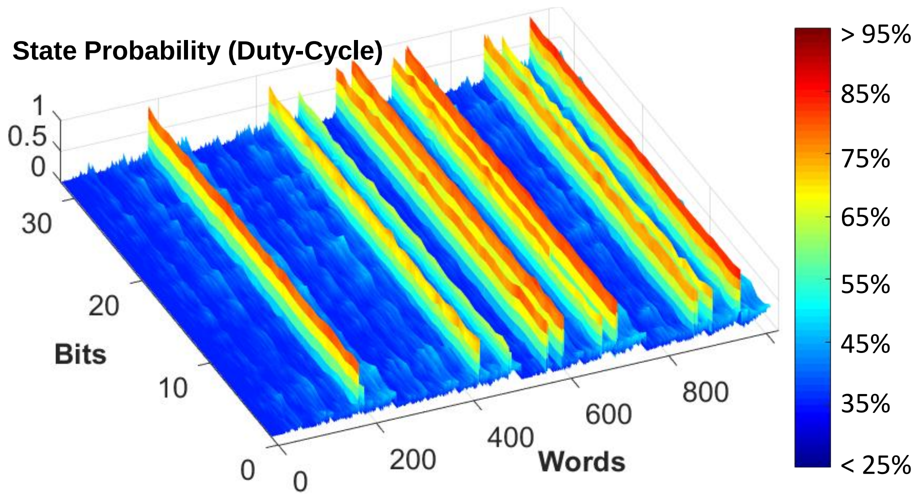
**Figure 46 – The degradation of the write margin, the read SNM, the Vdd-min-ret, and the IREAD of a memory cell due to BTI, BTI and HCI, shown in (a)-(d), respectively.**

Figure 46 shows the degradation of the read SNM, the write margin, Vdd-min-ret, and $I_{READ}$ of a memory cell. It shows a comparison between degradation due to BTI alone and due to both BTI and HCI. BTI severely degrades the read SNM, as well as the write margin. Vdd-min-ret is also affected, while $I_{READ}$ is relatively unaffected.

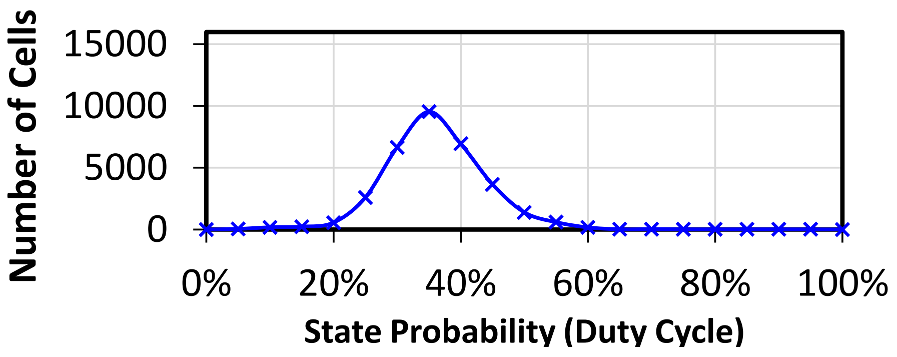
When BTI and HCI are considered simultaneously, as shown in Figure 46, the stability degradation of the write margin, the read SNM, and Vdd-min-ret is smaller than the degradation when only BTI is present, while for $I_{READ}$, the combined effect of BTI and HCI produces larger degradation than BTI alone. As $I_{READ}$ has limited degradation, SRAM lifetimes are mostly determined by the degradation of the write margin, the read SNM and Vdd-min-ret. Therefore, studying BTI and HCI separately leads to a significant overestimation of SRAM stability degradation, as the inclusion of HCI actually 'mitigates' the stability degradation due to BTI.

## 7.3 Activity Extraction of SRAM cells

The activity profile of the SRAM cells in the data cache was extracted using the same framework mentioned in Section 6.1. The thermal and IR-drop profiles were extracted in the same way, too. The probability is obtained of a transition occurring in each cell, together with the probability at each logic state, i.e., logic ‘1’ and ‘0’. Figure 47 and Figure 48 show the distributions of the state probabilities and the transition rate, respectively, of the data cache, when the microprocessor is running a standard benchmark.

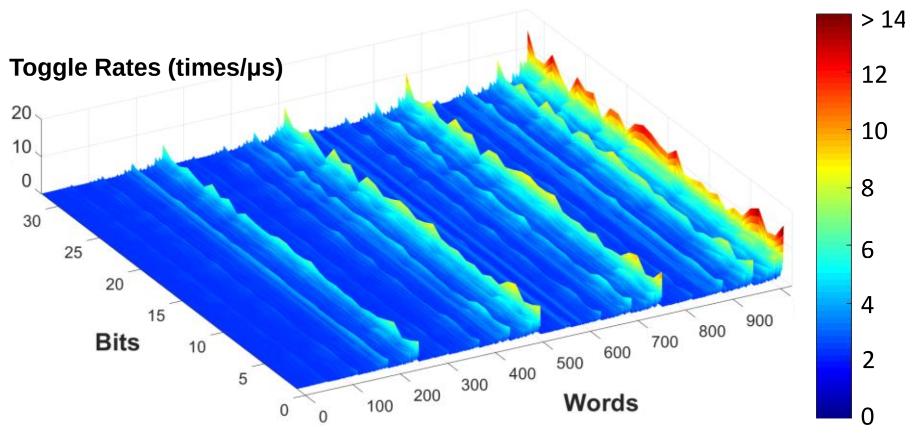


(a)

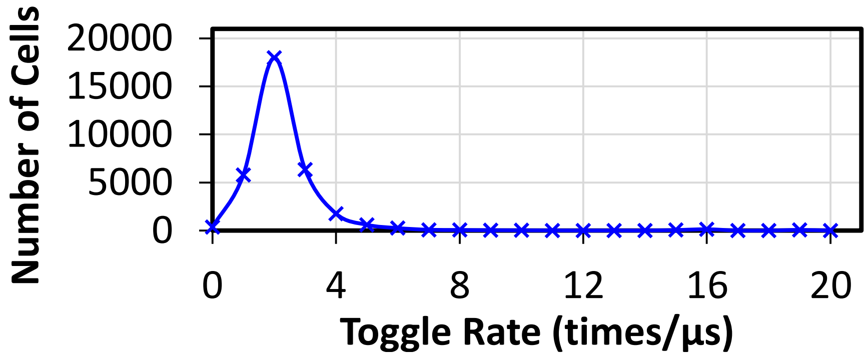


(b)

**Figure 47 – (a) The distribution of state probability for the 32KB data cache shown in 1024 words; (b) The histogram of the state probability distribution in the number of SRAM cells.**



(a)



(b)

**Figure 48 – (a) The distribution of transition rate for the 32KB data cache shown in 1024 words; (b) The histogram of the transition-rate distribution in the number of SRAM cells.**

The reason for this mitigation effect of HCI can be attributed to the impaired degradation of the transistors within one cell. BTI stress is different for PMOS and NMOS devices in the same cell. The cell becomes increasingly skewed under BTI as some devices degrade more than the others. This leads to impaired noise immunity. On the other hand, all the devices undergo the same stress due to HCI, as mentioned in Section 7.1. As a result, the inclusion of HCI mitigates the unevenness of the *Vth* degradation for the transistors in an SRAM cell.

Please also note that the HCI effect has a strong dependence on the operating frequency. Throughout the work, the LEON3 microprocessor is assumed to run at 250MHz. For this situation, BTI is dominant and HCI has a smaller influence. HCI could have more influence when the operating frequency reaches the GHz range.

When any of the four performance metrics mentioned in Section 7.2 degrade to a predefined threshold, the SRAM cell is said to have failed, and thus the lifetime of the cell is obtained. Using Monte Carlo simulations, the lifetime distribution of an SRAM cell is obtained for the given performance constraints.

Running SPICE simulations for each SRAM cell is very computationally-expensive. In order to manage the large volume of SRAM cells and to limit the number of SPICE simulations, we partition both the static stress probability and switching activity into 21 states (0%, 5%, 10%, … , 95%, 100%) to balance the accuracy and computational cost of the simulations for BTI and HCI, respectively.  It's assumed that the cells in the same stress state have the same stress.

For BTI, the 21 stress states represent duty cycles, that is, 0%, 5%, 10%, … , 100% duty cycle. 0% duty cycle means the cell has 0% time storing a ‘1’, while 100% duty cycle corresponds to 100% time storing a ‘1’. For HCI, the 21 stress states are proportional to the maximum observed transition rate, that is, 0%, 5%, 10%, … , 100% of the maximum transition rate. One example of the distributions of stress states for BTI and HCI is illustrated in Figure 47(b) and Figure 48(b), respectively, for a 32KB data cache. Note that the stress distribution not only depends on the applications being run, but also depends on the memory allocation of the cache system, which will be discussed in detail in Section

7.4. When BTI and HCI are combined, the stress states are combinations of the duty-cycle state and the toggle-rate state. For example, a stress state could have a low duty cycle and a high toggle rate, or a high duty cycle and a low toggle rate. The number of combinations is 21×21=441. Figure 49 shows an example of the stress-state distribution when BTI and HCI are combined.

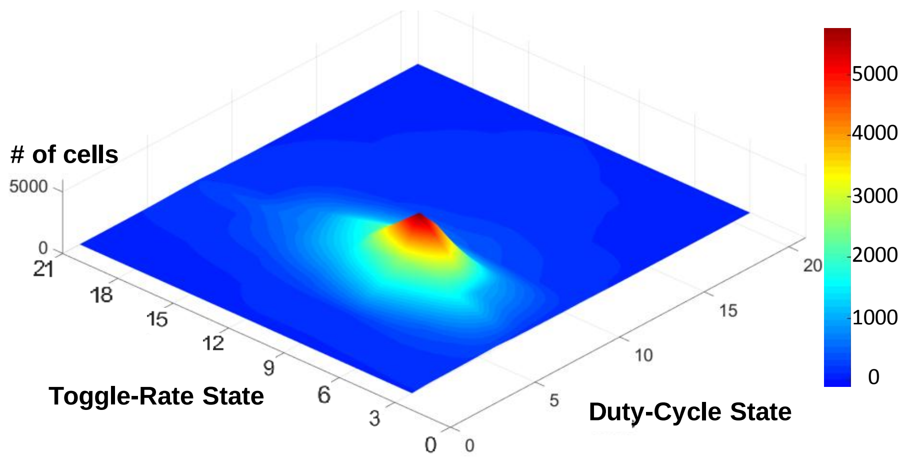


**Figure 49 – An example of a stress-state distribution for a 32KB memory is shown. A stress state is a combination of the duty-cycle state and the toggle-rate state, when the combined effect of BTI and HCI is considered. The z axis is the number of cells.**

Since process variations are considered, the lifetime of each SRAM cell is a distribution rather than a fixed value. Because we assume all the cells in the same stress state have the same stress, all the cells in one stress state share the same lifetime distribution. By running Monte Carlo simulations in SPICE, the lifetime distribution is computed for each stress state. Importance sampling [113] ensures adequate sampling of the tails of the distribution. Example lifetime distributions for four different stress state are illustrated in Figure 50 for the combined effect of BTI and HCI. As can be seen from this figure, an SRAM cell has a longer lifetime when it has a 50% duty cycle than when it has a 0% duty cycle. Moreover, higher toggle rates result in better lifetimes.

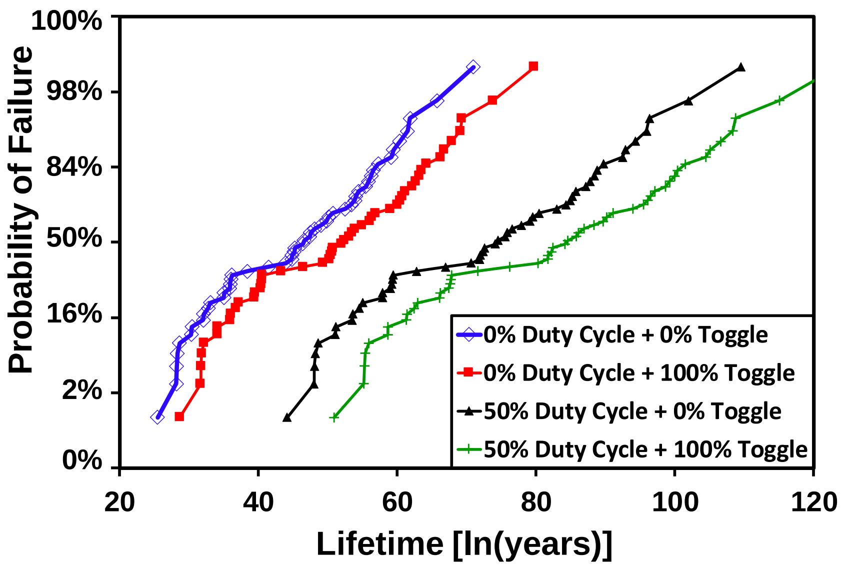


**Figure 50 – The lifetime distribution of a SRAM cell when it's in a specific stress state. Each stress state is a combination of the duty-cycle state and the toggle-rate state. Four stress states are shown in this figure.**

Because SRAM stability is very sensitive to temperature and supply voltage, the actual temperature and IR-drop profiles are needed for accurate lifetime estimation. The lifetime distribution of the 441 stress states is characterized for two different temperatures and two different supply voltages, that is, both temperature and supply voltage are partitioned into two states.

The lifetime distribution of SRAM cells shown in Figure 50 is best fit with a Log-Normal distribution. (Figure 50 is actually a Log-Normal probability plot.) The fitted Log-Normal distributions are used to determine the probability of failure of an SRAM cell, $PF_{bit}$, which is a function of time, $t$:

$$PF_{bit} = Probability\ of\ (Lifetime < t). \tag{29}$$

The probability of failure of a word is then calculated by

$$PF_{word} = 1 - \prod_{i=1}^{N} \left(1 - PF_{bit_i}\right) \tag{30}$$

where $PF_{word}$ is the probability of failure of a word, $PF_{bit}$ is the probability of failure of a bit, and *N* is the number of bits in one word. The word size is *N=32* for the data cache of the LEON3. Since $PF_{bit}$ changes as a function of time, $PF_{word}$ also changes as a function of time.

If the SRAM does not use error correcting codes, the memory fails when the first cell fails to work. The probability of failure of the SRAM block is obtained accordingly as a function of time:

$$PF_{SRAM} = 1 - \prod_{i=1}^{N_{word}} \left(1 - FP_{word_i}\right) \tag{31}$$

where $PF_{SRAM}$ is the probability of failure of the whole memory block, $FP_{word_i}$ is the probability of failure of word $i$, and $N_{word}$ is the number of words.

Error correcting codes can ensure that a memory system can tolerate faults. BCH codes [114] require seven additional bits per word and can correct one bit per word. The relationship between failures of single bits, $P_{fail}$, and the failure of the word is modeled with a binomial distribution. For a word containing *N* bits, the probability of failure of a word, $F_{word}$, is

$$PF_{word} = 1 - \prod_{i=1}^{N}\left(1 - PF_{bit_i}\right) - \sum_{j=1}^{N}[PF_{bit_j} * \prod_{i \neq j}\left(1 - PF_{bit_i}\right)] \tag{32}$$

The word size when there are ECCs is $N=39$ for the D-Cache, I-Cache, and RF blocks of the LEON3. The failure probability of the memory, $PF_{SRAM}$, is calculated using (31).

## 7.4 Performance-Reliability Analysis for Different Cache Configurations

Based on the method for memory lifetime characterization in Section 7.2, the reliability (failure rate) of the LEON3 L1 data cache was studied for different cache configurations: associativity, cache line size, cache size, and the replacement algorithm. The impact of Error Correcting Codes (ECC) was also analyzed.

Six representative benchmarks from MiBench [84] were run on the microprocessor: Basicmath, Qsort, SHA, CRC32, FFT and Dijkstra. The Basicmath benchmark performs simple mathematical calculations that often don’t have dedicated hardware support in embedded processors. Qsort sorts a large array of strings into ascending order using the well-known quick sort algorithm. SHA is the secure hash algorithm that produces a 160-bit digest for a given input. CRC32 is a benchmark performing a 32-bit Cyclic Redundancy Check (CRC) on a file to detect errors in data transmission. FFT performs a Fast Fourier Transform on an array of data. The Dijkstra benchmark constructs a large graph in an adjacency matrix representation and then calculates the shortest path between every pair of nodes using repeated applications of Dijkstra’s algorithm.

The framework in Section 6.1 was used to extract the Duty-Cycle/Toggle-Rate, temperature, and IR-drop profiles of the data cache for different cache designs and for the six applications above. The method for memory lifetime characterization described in Section 7.2 was then used to calculate the failure rate of the data cache while running each specific application.

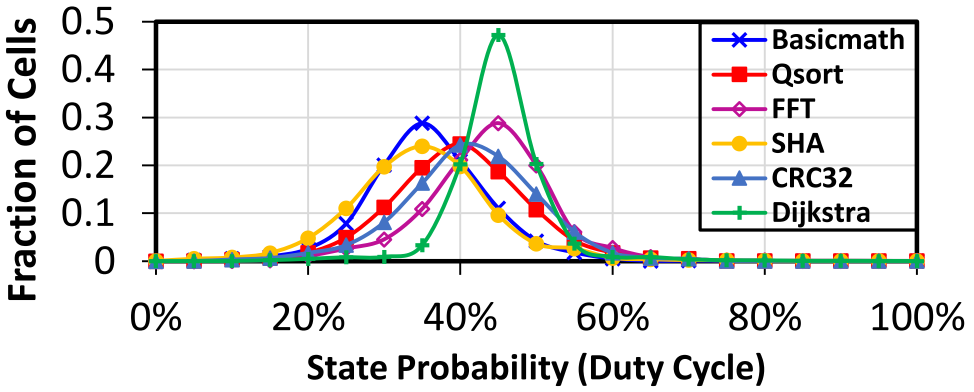


**Figure 51 – The duty-cycle distributions of SRAM cells in a 2-way 32KB data cache, while the microprocessor is running six different benchmarks.**

Figure 51 shows the state probability (duty-cycle) distribution for each application using a 2-way 32KB data cache with 16Byte line size. Clearly, logic ‘0’ is the predominant state. Memories in a processor contain more 0s than 1s throughout normal operations [115]. In general, ‘0’ is stored longer than ‘1’ because the memory is usually initialized to zero when it’s allocated. Thus, even if there is an equal likelihood of an application writing a ‘0’ or a ‘1’ in any bit position, this initialization will always mean that ‘0’ is stored longer. Other reasons for ‘0’ being stored longer are that false Boolean values and NULL pointers are represented with zero, as well as most data in dense-form sparse matrices [114].

### *7.4.1 Associativity*

Cache associativity can be seen as bookshelves in different shapes and sizes. Caches fall into one of three categories: direct mapped, n-way set associative, and fully associative. Direct mapped caches are designed so that a cache block can only go in one spot in the cache. 2-way set associative caches are made up of sets that can each fit two blocks, while in 4-way set associative cache, each set fits four blocks. For a fully associative cache, a cache block can go anywhere in the cache. It is worth noting that the direct mapped cache is actually a 1-way set associative cache and a fully associative cache of m blocks is an *m*-way set associative cache. Higher associativity can improve the hit rate, but will reduce cycle time and cost more area because of the need for more comparators. The L1 data cache of the LEON3 microprocessor was implemented with three different associativities: 1-way, 2-way and 4-way, while the cache line size (16Byte), cache size (32KB), and the replacement algorithm (LRU) were kept the same.

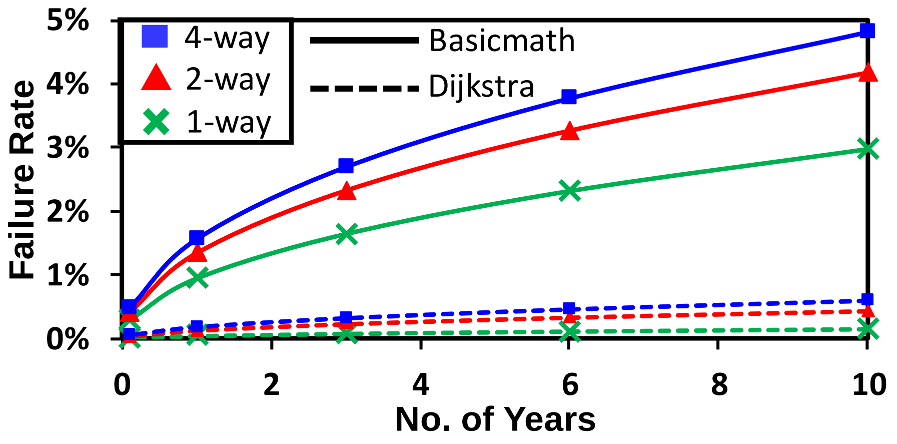


**Figure 52 – The failure probability as a function of time for the three different associativities and two benchmarks.**

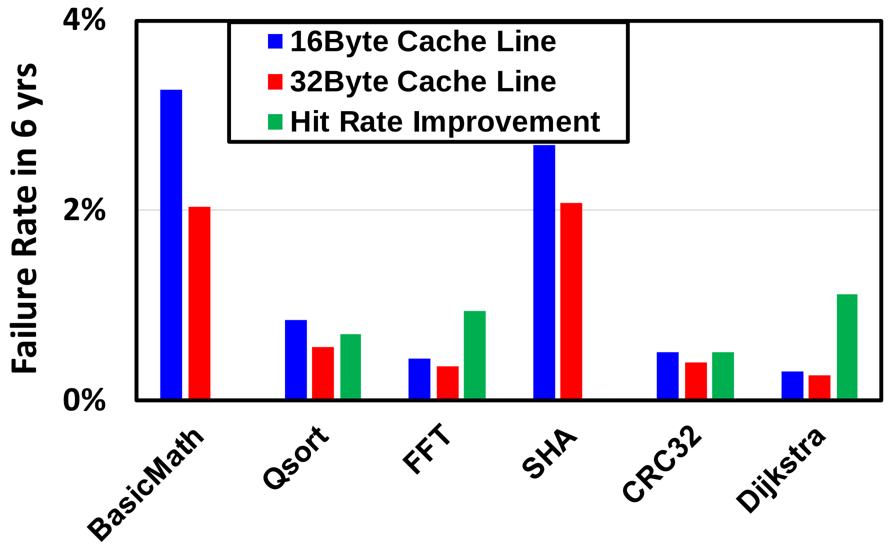


**Figure 53 – The failure probabilities in 6 years for a 16 Byte cache line and a 32 Byte cache line for six applications. The hit-rate improvement is also shown, defined as the improvement of using a 32 Byte cache line compared to a 16 Byte line.**

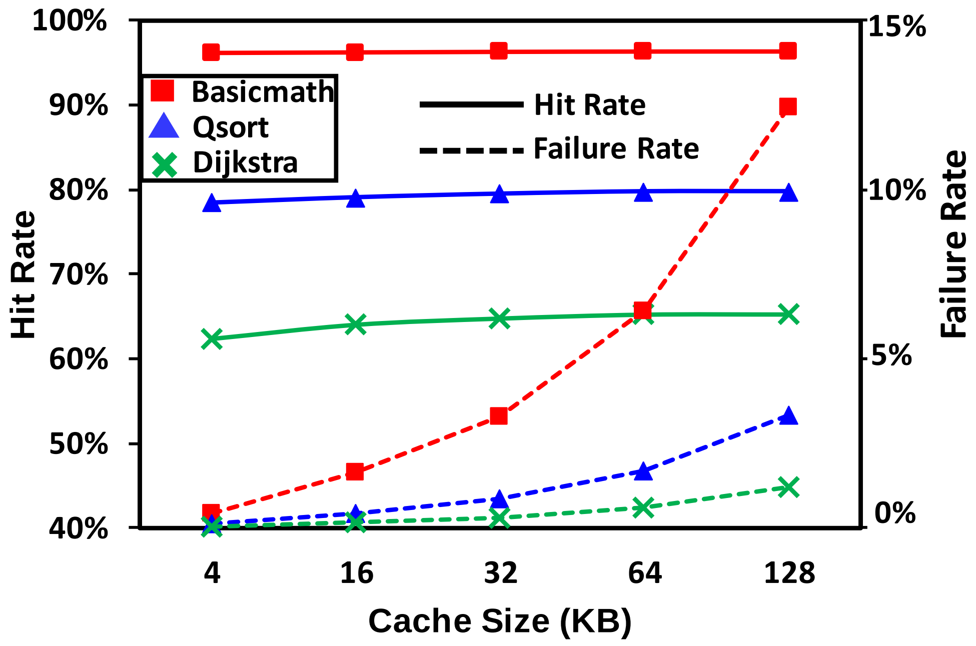


**Figure 54 – The hit rate and the failure probability in 6 years are shown for five different cache sizes and for three applications.**

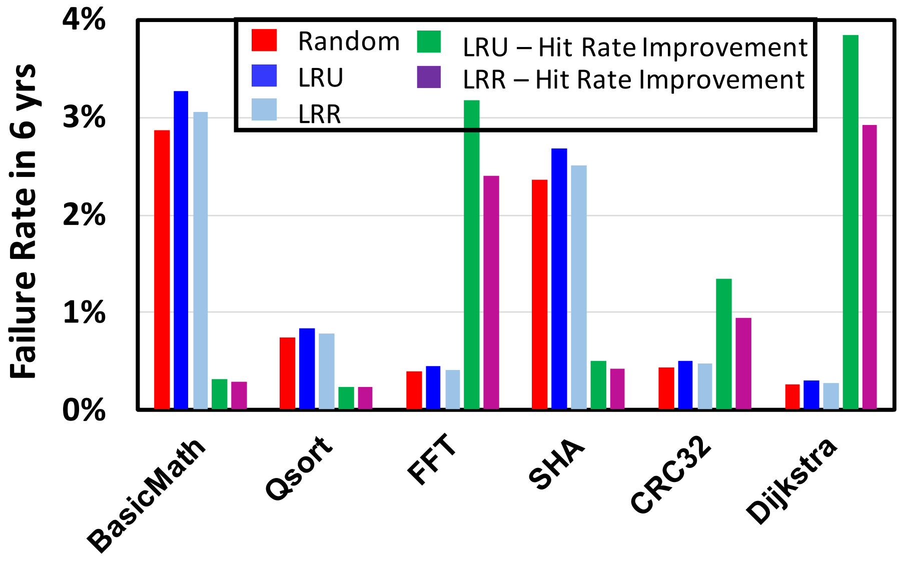


**Figure 55 – The failure probabilities in 6 years for three different replacement algorithms for six benchmarks, as well as the hit rate improvements of LRU and LRR. The hit-rate improvement is defined as the improvement compared to the 'Random' replacement policy.**

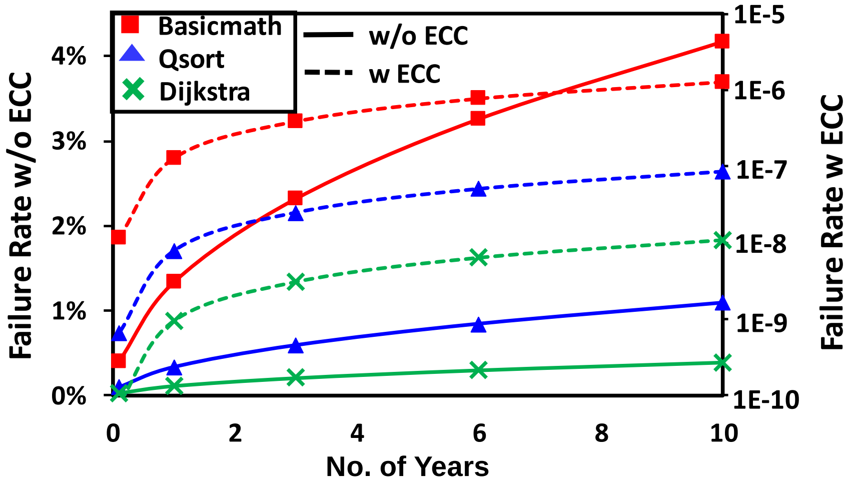


**Figure 56 – The failure probabilities of the 2-way 32KB data cache with and without ECCs are shown as a function of time for three applications.**

Figure 52 shows the failure rate for different associativities. For illustration purposes, the results from two applications are shown: Basicmath and Dijkstra, since other applications produce the same trend. The hit rate of 1-way, 2-way and 4-way associativities are 96.12%, 96.33%, 96.36%, respectively, for Basicmath, and are 62.23%, 64.81%, 65.54%, respectively, for Dijkstra. Higher associativity results in a higher hit rate, but also increases the failure rate. A higher hit rate produces fewer misses, and thus the cells are more likely to keep their stored values unchanged, which aggravates the BTI effect. From the perspective of aging, a cache miss is potentially useful as it flips the value stored in a cell and therefore mitigates BTI.

### *7.4.2 Cache Line Size*

Data is transferred between the main memory and the cache in blocks of fixed size, called cache lines. When a cache line is copied from the main memory into the cache, a cache entry is created. The cache entry includes the copied data as well as the requested memory location (called a Tag).

We implemented the data cache with two different cache line sizes: 16 Byte and 32 Byte, while the associativity (2-way), cache size (32KB), and the replacement algorithm (LRU) are kept the same. Figure 53 shows the failure probabilities of the six benchmarks for a 16 Byte and a 32 Byte cache line. It’s observed that the 32 Byte cache line has a lower failure rate than the 16 Byte cache line for all six benchmarks. Except for Basicmath and SHA, which see little hit-rate improvement, the 32 Byte cache line achieves better performance than the 16 Byte cache line. Overall, the 32 Byte cache line is better than the 16 Byte cache line in both performance and reliability. This observation is a little counter-intuitive as we’ve shown that a higher hit rate results in lower reliability in Section 7.4.1. So it might be straightforward to think that the 32 Byte cache line would have a higher failure rate because of its higher hit rate. However, a cache miss in a 32 Byte cache line produces recovery cycles for up to 256 ($32\times8$) SRAM cells which is twice as many as with a 16 Byte cache line ($16\times8$ SRAM cells). Therefore, although a 32 Byte cache line has less misses, it actually has a larger number of BTI stress recovery cycles than a 16 Byte cache line, which results in improved reliability.

### *7.4.3 Cache Size*

The total size of the data cache is another important metric in cache system design. In our experiments, the data cache is implemented with five different cache sizes: 4KB, 16KB, 32KB, 64KB and 128KB, while the associativity (2-way), cache line size (16Byte), and replacement algorithm (LRU) are kept the same. Figure 54 shows the hit rate and failure rate (in 6 years) for five different cache sizes for three applications. The failure rate increases dramatically as the cache size increases. This can be easily understood because the failure probability is larger when there are more SRAM cells.

It’s also observed that the hit rate increases as the cache size increases. However, when the cache size is larger than 32KB, little improvement is seen in the hit rate. According to the performance specification and reliability budget, cache designers could determine an optimal cache size balancing both performance and reliability requirements.

### *7.4.4 Replacement Algorithm*

In order to make room for a new entry on a cache miss, the cache may have to evict one of the existing entries. The heuristic that it uses to choose the entry to evict is called the replacement policy. The fundamental problem with any replacement policy is that it must predict which existing cache entry is least likely to be used in the future.

In this work, three different replacement algorithms were implemented, namely, Random, Least-Recently-Replaced (LRR) and Least-Recently-Used (LRU), while the associativity (2-way), cache line size (16 Byte) and cache size (32KB) were kept the same. The 'random' algorithm randomly selects a cache entry to evict. It uses a simple 1- or 2-bit counter to select the eviction entry and has low area overhead. The LRR algorithm evicts the cache entry which was least recently replaced. It uses one extra bit in the Tag part and has therefore also low area overhead. The LRU algorithm, a popular replacement policy, evicts the entry which was least recently used. The LRU scheme has typically the best performance, but also the highest area overhead. A two-way LRU uses one flip-flop per cache line, a three-way LRU uses three flip-flops per cache line, and a four-way LRU uses five flip-flops per cache line to store the access history. Overall, LRU favors the most recently accessed data, while LRR favors recently loaded data.

The failure probabilities for the three replacement algorithms under study are shown in Figure 55, as well as the hit-rate improvement of LRU and LRR compared to the random algorithm. It's observed that LRU achieves the best hit rate in all the applications. However, it always has lower reliability compared to LRR and the random algorithm. So tradeoffs between performance and reliability, together with other important metrics such as leakage power etc., should be made according to the priority associated with each metric.

The reason for the above results is similar to the observations on the impact of associativity. A higher hit rate means fewer misses, which results in fewer recovery cycles. As a result, the cells suffer from more BTI degradation.

### *7.4.5 Error Correcting Codes (ECC)*

ECC is a technique which can detect and correct the most common sources of internal data corruption. ECCs add some redundancy (some extra bits) to check the consistency of the data and to recover data determined to be corrupted. The word size containing the ECC codes for the data cache of LEON3 is N=39 when the implemented ECC is designed to correct single-bit errors.

The failure probabilities of the data cache for a 2-way 32KB data cache with and without ECC codes are shown in Figure 56 as a function of time. Again, three benchmarks are illustrated as examples, and other benchmarks produce similar results. It can be seen that ECCs lead to a substantial improvement in the failure probability.

## 7.5 Conclusions

The reliability and performance of the data cache is analyzed while varying five different configuration parameters: associativity, cache line size, cache size, the replacement algorithm, and ECC codes. A general rule is that higher performance (higher hit rate) results in lower reliability (higher failure probability). This can be attributed to the fact that cache misses are helpful for reliability. When a cache miss happens, the old data in the specific cache line needs to be replaced by new data. When the value stored in a SRAM cell flips, the stressed transistors recover from BTI stress, and this improves reliability. If a cache configuration has a higher hit rate, it's expected to see a higher failure probability. For example, for associativity, larger associativity results in better performance, but worse reliability. For the replacement algorithm, the 'random' replacement policy has the worst hit rate and the best reliability, while the popular LRU algorithm has the best hit rate and the worst reliability.

One exception of this rule happens for different cache line sizes. The 32 Byte cache line is better than the 16 Byte cache line in both performance and reliability. One cache miss for a 32 Byte cache line can recover as many as 256 ($32 \times 8$) cells from BTI stress, which is twice as many as for a 16 Byte cache line. Therefore, despite the fact that the 32 Byte cache line has fewer cache misses, it actually results in more BTI stress recovery. Thus, the 32 Byte cache line achieves higher performance and better reliability.

Cache size is of great significance to both cache performance and reliability. It is observed that when the cache size increases to larger than 16KB, the cache reliability dramatically drops, while the performance (hit rate) has very limited improvement. Moreover, ECC always improves reliability at the cost of area and power overhead.

Overall, the proposed framework can efficiently evaluate the performance and reliability of the cache memory and can provide insight to cache designers to help them optimize performance-reliability tradeoffs by selecting the appropriate cache configurations based on the specification budget and lifetime requirements.

# 8. CONCLUSIONS

## 8.1 Conclusions of This Research

This research presents a framework of aging simulator which can assess the lifetimes of complex systems like state-of-art microprocessors, while taking into account the effect of PVT variations and a variety of Front-end-of-line (FEOL) device wearout mechanisms. The proposed aging simulator achieves accurate statistical timing analysis to study the combined impact of aging effect, manufacturing variability, ambient conditions and realistic workload. The timing analyzer in this work is the first attempt in the literature to achieve a comprehensive process-voltage-temperature-aging-aware statistical timing analysis (StTA) while considering the effect of bias temperature instability (BTI), hot carrier injection (HCI), gate oxide breakdown (GOBD) simultaneously.

The proposed aging simulator presents the relationship between circuit performance (speed) and circuit reliability, and gives insights for designers to achieve optimal tradeoff between performance and reliability. Moreover, circuit designers can benefit from the proposed work to avoid excessive guard-banding to achieve higher performance while still maintaining the required reliability.

This research also includes the SRAM lifetime analysis, particularly the lifetimes of the data cache within state-of-art microprocessors. The work studies the performance and reliability for different cache configurations, and provides insight to cache designers to help them optimize performance-reliability tradeoffs by selecting the appropriate cache configurations based on the specification budget and lifetime requirements.

## 8.2 Future Work

While the proposed aging simulator can study the system lifetimes of state-of-art microprocessors due to a variety of wearout mechanisms, this research is limited to microprocessors with only one core in the system. Future work might involve the lifetime analysis of multi-core microprocessors and heterogeneous systems that use more than one kind of processor or cores.

Reliability of FinFET (Fin Field Effect Transistor) technology has been widely used in some performance-driven applications. The device-level wearout mechanisms, as well as the system lifetime estimations, would be quite different in FinFET technology and remain to be discovered and explored.

The cache reliability discussed in Chapter 7 deal with front-end wearout mechanisms only, and the proposed framework can be easily extended to study the cache reliability due to backend wearout mechanisms for different cache configurations.

# REFERENCES

[1] Taizhi Liu, Chang-Chih Chen and Linda Milor, "Comprehensive Reliability-Aware Statistical Timing Analysis Using a Unified Gate-Delay Model for Microprocessors," *IEEE Transactions on Emerging Topics in Computing*, 2016.

[2] Taizhi Liu, Chang-Chih Chen and Linda Milor, "Accurate Standard Cell Characterization using Multivariate Adaptive Regression Splines," *Proc. of International Symposium on Quality Electronic Design (ISQED)*, pp. 272–279, 2015.

[3] Taizhi Liu, Chang-Chih Chen and Linda Milor, "Process Variation Aware Timing Analysis Based on a Monte Carlo Framework Using a Novel Delay Model," *Proc. of TAU Workshop*, 2015.

[4] Taizhi Liu, Seyed-Abdollah Aftabjahani and Linda S. Milor, "Compact variation-aware models for standard cells with interconnect-dominated loads for statistical static timing analysis," *Proc. of Design of Circuits and Integrated Systems (DCIS)*, 2013.

[5] Taizhi Liu, Chang-Chih Chen, Soonyoung and Linda Milor, "System-level variation-aware aging simulator using a unified novel gate-delay model for bias temperature instability, hot carrier injection, and gate oxide breakdown," *Microelectronics Reliability*, vol. 55, no. 9–10, pp. 1334–1340, 2015.

[6] Taizhi Liu, Chang-Chih Chen, Woongrae Kim and Linda Milor, "Comprehensive reliability and aging analysis on SRAMs within microprocessor systems," *Microelectronics Reliability*, vol. 55, no. 9–10, pp. 1290–1296, 2015.

[7] Taizhi Liu, Chang-Chih Chen, Jiadong Wu and Linda Milor, "SRAM stability analysis for different cache configurations due to Bias Temperature Instability and Hot Carrier Injection," *Proc. of IEEE 34th International Conference on Computer Design (ICCD)*, pp. 225–232, 2016.

[8] Chang-Chih Chen, Taizhi Liu, Soonyoung Cha, Linda Milor, "Processor-level reliability simulator for time-dependent gate dielectric breakdown", *Microprocessors and Microsystems*, vol. 39, no. 8, pp. 950–960, 2015.

[9] Soonyoung Cha, Chang-Chih Chen, Taizhi Liu and Linda S. Milor, “Extraction of threshold voltage degradation modeling due to negative bias temperature instability in circuits with I/O measurements,” *Proc. of VLSI Test Symposium (VTS)*, pp. 1–6, 2014.

[10] Chang-Chih Chen, Soonyoung Cha, Taizhi Liu and Linda S. Milor, “System-Level modeling of microprocessor reliability degradation due to BTI and HCI,” *Proc. International Reliability Physics Symposium (IRPS)*, pp. CA 8.1–CA 8.9, 2014.

[11] Soonyoung Cha, Dae-Hyun Kim, Taizhi Liu and Linda S. Milor, “The die-to-die calibrated combined model of negative bias temperature instability and gate oxide breakdown from device to system,” *Microelectronics Reliability*, vol. 55, no. 9–10, pp. 1404–1411, 2015.

[12] Woongrae Kim, Chang-Chih Chen, Taizhi Liu, Soonyoung Cha and Linda Milor, “Estimation of remaining life using embedded SRAM for wearout parameter extraction”, *Proc. of IEEE International Workshop on Advances in Sensors and Interfaces (IWASI)*, pp. 243–248, 2015.

[13] Chang-Chih Chen, Taizhi Liu, and Linda Milor, “System-Level Modeling of Microprocessor Reliability Degradation Due to Bias Temperature Instability and Hot Carrier Injection”, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 8, pp. 2712–2725, 2016.

[14] Y. Cao et al., "Cross-layer modeling and simulation of circuit reliability", *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 1, pp. 8-23, Jan. 2014.

[15] A. Singhee and R. A. Rutenbar, “From finance to flip flops: A study offast quasi-Monte Carlo methods from computational finance applied to statistical circuit analysis,” in *Proc. IEEE Int. Symp. Quality Electronic Design (ISQED)*, 2007, pp. 685–692.

[16] M. Keramat and R. Kielbasa, “Worst case efficiency of LHSMC yield estimator of electrical circuits,” in *Proc. Int. Symp. on Circuits and Systems*, vol. 3. 1997, pp. 1660–1663.

[17] R. Kanj, R. Joshi, and S. Nassif, “Mixture importance sampling and its application to the analysis of SRAM designs in the presence of rare failure events,” in *Proc. IEEE Design Automation Conf. (DAC)*, 2006, pp. 69–72.

[18] A. Singhee, S. Singhal, and R. A. Rutenbar, “Practical, fast Monte Carlo statistical static timing analysis: Why and how,” in *Proc. International Conference on Computer-Aided Design (ICCAD)*, 2008, pp. 190–195.

[19] W. Zhang, A. Singhee, J.Xiong, P. Habitz, A. Joshi, C. Visweswariah and J. Sundquist, “A dynamic method for efficient random mismatch characterization of standard cells,” in *Proc. International Conference on Computer-Aided Design (ICCAD)*, 2012, pp. 180–186.

[20] K. Okada, K. Yamaoka, and H. Onodera, “A statistical gate-delay model considering intra-gate variability,” in *Proc. International Conference on Computer-Aided Design (ICCAD)*, 2003, pp. 908–913.

[21] K. Okada, K.Yamaoka, and H. Onodera, “A statistical gate delay model for intra-chip and inter-chip variabilities,” in *Proc. Asia & South Pacific Design Automation Conf. (ASPDAC)*, 2003.

[22] S. Sundareswaran, J. Abraham, R. Panda and A. Ardelea, “Characterization of standard cells for intra-cell mismatch variations,” *IEEE Trans. Semiconductor Manufacturing*, vol. 22, no. 1, pp. 40–49, Feb. 2009.

[23] L. Brusamarello, G. I. Wirth, P. Roussel and M. Miranda, “Fast and accurate statistical characterization of standard cell libraries,” *Microelectronics Reliability*, vol. 51, no. 12, pp. 2341–2350, Dec. 2011.

[24] S. Gupta and S. S. Sapatnekar, “Compact current source models for timing analysis under temperature and body bias variations,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 11, pp. 2104–2117, Nov. 2012.

[25] B. Amelifard, S. Hatami, H. Fatemi, and M. Pedram, “A current source model for CMOS logic cells considering multiple input switching and stack effect,” in *Proc. Design, Automation & Test in Europe Conf. (DATE)*, 2008, pp. 568–574.

[26] A. Goel and S. Vrudhula, “Current source based standard cell model for accurate signal integrity and timing analysis,” in *Proc. Design, Automation & Test in Europe Conf. (DATE)*, 2008, pp. 574–579.

[27] B. Liu and A. B. Kahng, “Statistical gate level simulation via voltage controlled current source models,” in *Proc. IEEE Int. Workshop Behavioral Modeling Simulation*, Sep. 2006, pp. 23-27.

[28] A. Goel and S. Vrudhula, “Statistical waveform and current source based standard cell models for accurate timing analysis,” in *Proc. IEEE Design Automation Conf. (DAC)*, 2008, pp. 227–230.

[29] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan, “First-order incremental block-based statistical timing analysis,” in *Proc. IEEE Design Automation Conf. (DAC)*, 2004, pp. 331–336.

[30] H. Chang and S. S. Sapatnekar, “Statistical timing analysis considering spatial correlations using a single pert-like traversal,” in *Proc. International Conference on Computer-Aided Design (ICCAD)*, 2003, pp. 621–625.

[31] K. Chopra, S. Shah, A. Srivastava, D. Blaauw, and D. Sylvester, “Parametric yield maximization using gate sizing based on efficient statistical power and delay gradient computation,” in *Proc. International Conference on Computer-Aided Design (ICCAD)*, 2005, pp. 1023–1028.

[32] Y. Zhan, A. J. Strojwas, X. Li, L. T. Pileggi, D. Newmark, and M. Sharma, “Correlation-aware statistical timing analysis with non-Gaussian delay distributions,” in *Proc. IEEE Design Automation Conf. (DAC)*, 2005, pp. 77–82.

[33] S. Bhardwaj, P. Ghanta, and S. Vrudhula, “A framework for statistical timing analysis using non-linear delay and slew models,” in *Proc. International Conference on Computer-Aided Design (ICCAD)*, 2006, pp. 225–230.

[34] L. Cheng, J. Xiong, and L. He, “Non-Gaussian statistical timing analysis using second-order polynomial fitting,” *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 28, no. 1, pp. 130–140, Jan. 2009.

[35] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, “Equation of state calculations by fast computing machines,” *J. Chem. Phys.*, vol. 21, no. 6, pp. 1087–1092, Jun. 1953.

[36] L. Scheffer, “The count of Monte Carlo,” in *Proc. of TAU Workshop*, 2004.

[37] 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 Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 2, pp. 173–183, Feb. 2010.

[38] F. Firouzi, S. Kiamehr, and M.B. Tahoori, “Statistical analysis of BTI in the presence of process-induced voltage and temperature variations,” *Proc. Asia & South Pacific Design Automation Conf. (ASPDAC)*, 2013, pp. 594–600.

[39] F. Firouzi, S. Kiamehr, M. Tahoori, and S. Nassif, “Incorporating the impacts of workload-dependent runtime variations into timing analysis,” in *Proc. Design, Automation & Test in Europe Conf. (DATE)*, 2013, pp. 1022-1025.

[40] J. Fang and S.S. Sapatnekar, “The impact of BTI variations on timing in digital logic circuits,” *IEEE Trans. Device & Materials Reliability*, vol. 13, no. 1, pp. 277–286, March 2013.

[41] Y. Lu, L. Shang, H. Zhou, H. Zhu, F. Yang, and X. Zeng, “Statistical reliability analysis under process variation and aging effects,” in *Proc. IEEE Design Automation Conf. (DAC)*, 2009, pp. 514–519.

[42] S. Han and J. Kim, “NBTI-aware statistical timing analysis framework,” in *Proc. IEEE Int. SoC Conf.*, 2010, pp. 158–163.

[43] K. Ramakrishnan, R. Rajaraman, S. Suresh, N. Vijaykrishnan, Y. Xie, and M. Irwin, “Variation impact on SER of combinational circuits,” in *Proc. IEEE Int. Symp. Quality Electronic Design (ISQED)*, 2007, pp. 911–916.

[44] J. Fang and S.S. Sapatnekar, “Incorporating Hot Carrier Injection Effects into Timing Analysis for Large Circuits,” *IEEE Trans. Very Large Scale Integration Systems (TVLSI)*, vol. 22, no. 12, pp. 2738–2751, Dec. 2014.

[45] M. Choudhury, V. Chandra, K. Mohanram, and R. Aitken., "Analytical model for TDDB-based performance degradation in combinational logic." in *Proc. IEEE Design, Automation and Test in Europe (DATE)*, 2010, pp. 423-428.

[46] B. Kaczer, R. Degraeve, M. Rasras, K. Van de Mieroop, P. J. Roussel and G. Groeseneken, “Impact of MOSFET gate oxide breakdown on digital circuit operation and reliability,” *IEEE Trans. Electron Devices*, vol. 49, no. 3, pp. 500–506, Mar. 2002.

[47] S. Kiamehr, F. Firouzi, and M.B. Tahoori, “Aging-aware timing analysis considering combined effects of NBTI and PBTI,” *Proc. IEEE Int. Symp. Quality Electronic Design (ISQED)*, 2013, pp. 53-59.

[48] G.I. Wirth, R. da Silva, and B. Kaczer, “Statistical model for MOSFET bias temperature instability component due to charge trapping,” *IEEE Trans. Electron Devices*, vol. 58, no. 8, pp. 2743-2751, 2011.

[49] R. Fernandez, B. Kaczer, A. Nackaerts, S. Demuynck, R. Rodriguez, M. Nafria, and G. Groeseneken, “AC NBTI studies in the 1 Hz – 2 GHz range on dedicated on-chip CMOS circuit,” *Proc. Int. Electron Devices Meeting*, 2006.

[50] S. Zafar, Y.H. Kim, V. Narayanan, C. Cabral, V. Paruchuri, B. Doris, J. Stathis, A. Callegari, and M. Chudzik, “A comparative study of NBTI and PBTI (charge trapping) in SiO2/HFO2 stacks with FUSI, TiN, Re Gates,” *Proc. Symp. VLSI Tech.*, 2006, pp. 23-25.

[51] S.-Y. Chen, C.-H. Tu, P.-W. Kao, M.-H. Lin, H.-S. Haung, J.-C. Lin, M.-C. Wang, S.-H. Wu, Z.-W. Jhou, S. Chou, and J. Ko, “Investigation of DC hot-carrier degradation at elevated temperatures for p-channel metal-oxide-semiconductor field-effect transistors,” *Jpn. J. Appl. Phys*, vol. 47, no. 3, pp. 1527-1531, 2008.

[52] W. Wang, V. Reddy, A.T. Krishnan, R. Vattikonda, S. Krishnan, and Y. Cao, “Compact Modeling and Simulation of Circuit Reliability for 65-nm CMOS Technology,” *IEEE Trans. Device and Materials Reliability*, vol. 7, no. 4, pp. 509-517, 2007.

[53] C. Ma, B. Li, L. Zhang, J. He, X. Zhang, X. Lin, and M. Chan, “A Unified FinFET Reliability Model Including High K Gate Stack Dynamic Threshold Voltage, Hot Carrier Injection, and Negative Bias Temperature Instability,” *Proc. Int. Symp. Quality Electronic Design, 2009*, pp. 7-12.

[54] C.H. Tu, S.Y. Chen, A.E. Chuang, H.S. Huang, Z.W. Jhou, C.J. Chang, S. Chou, and J. Ko, “Transistor variability after CHC and NBTI stress in 90 nm pMOSFET technology,” *Electronics Letters*, vol. 45, no. 15, pp. 854-856, 2009.

[55] K. Okada, "Analysis of the relationship between defect site generation and dielectric breakdown utilizing A-mode stress induced leakage current." *IEEE Trans. on Electron Devices*, vol. 47, no. 6, pp. 1225-1230, 2000.

[56] R. Degraeve, J.L. Ogier, R. Bellens, P.J. Roussel, G. Groeseneken, and H.E. Maes, "A new model for the field dependence of intrinsic and extrinsic time-dependent dielectric breakdown." *IEEE Trans. Electron Devices*, vol. 45, no. 2, pp. 472-481, 1998.

[57] S. Takagi, N. Yasuda, and A. Toriumi, “Experimental evidence of in- elastic tunneling and new I–V model for stress-induced leakage current,” *IEDM Tech. Dig.*, 1996., pp. 323-326.

[58] R. Degraeve, B. Kaczer, A. De Keersgieter, and G. Groeseneken. "Relation between breakdown mode and breakdown location in short channel NMOSFETs and its impact on reliability specifications." *IEEE Int. Reliability Physics Symp. (IRPS)*, 2001, pp. 360-366.

[59] B.P. Linder, D.J. Frank, J.H. Stathis, and S.A. Cohen, “Transistor-lim- ited constant voltage stress of gate dielectrics,” in *Symp. VLSI Technol. Dig.*, 2001, pp. 93–94.

[60] E. Y. Wu, E. J. Nowak, A. Vayshenker, W. L. Lai, and D. L. Harmon, “CMOS scaling beyond the 100-nm node with silicon-dioxide-based gate dielectrics,” *IBM Journal of Research and Development*, vol. 46, 2002.

[61] B. Kaczer, R. Degraeve, M. Rasras, K. Van de Mieroop, P.J. Roussel, and G. Groeseneken, "Impact of MOSFET gate oxide breakdown on digital circuit operation and reliability," *IEEE Trans. Electron Devices*, vol. 49, no. 3, pp. 500-506, March 2002.

[62] T. Nigam, A. Kerber, and P. Peumans. "Accurate model for time-dependent dielectric breakdown of high-k metal gate stacks." *IEEE Int. Reliability Physics Symposium (IRPS)*, 2009, pp. 523-530.

[63] M. Choudhury, V. Chandra, K. Mohanram, and R. Aitken., "Analytical model for TDDB-based performance degradation in combinational logic." in *Proc. Design, Automation & Test in Europe Conf. (DATE)*, 2010, pp. 423-428.

[64] S.Y. Kim, G. Panagopoulos, C.-H. Ho, M. Katoozi, E. Cannon, and K. Roy. "A Compact SPICE Model for Statistical Post-Breakdown Gate Current Increase Due to TDDB." *IEEE Int. Reliability Physics Symposium (IRPS)*, 2013.

[65] J.H. Stathis, “Percolation models for gate oxide breakdown,” *J. Appl. Phys.*, vol. 86, no. 10, pp. 5757–5766, Nov. 1999.

[66] M. Houssa, P.W. Mertens, and M.M. Heyns. “Relation between stressinduced leakage current and time-dependent dielectric breakdown in ultra-thin gate oxides.” *Semiconductor Sci. Technol.*, vol. 14, no. 10, pp. 892–896, 1999.

[67] S. Lombardo, J. H. Stathis, B. P. Linder, K. L. Pey, F. Palumbo, and C. H. Tung. "Dielectric breakdown mechanisms in gate oxides." *Journal of Applied Physics*, vol. 98, no. 12, p. 121301, Dec. 2005.

[68] E. Miranda and J. Su ñé "Analytic modeling of leakage current through multiple breakdown paths in SiO2 films." *IEEE Int. Reliability Physics Symp. (IRPS)*, 2001, pp. 367-379.

[69] G.E.P. Box, W.G. Hunter, and J.S. Hunger, Statistics for Experimenters, 2nd Edition. *New York: John Wiley and Sons*, 2005.

[70] *HSPICE User Manual*: www.synopsys.com

[71] M. Celik, L. Pileggi, and A. Odabasioglu, IC Interconnect Analysis. *New York: Springer Science & Business Media*, 2002.

[72] E. Acar, A. Odabasioglu, M. Celik, and L. T. Pileggi, "S2P: A stable 2-pole RC delay and coupling noise metric," in *Proc. Great Lakes Symp. on VLSI*, 1999.

[73] J. H. Friedman, "Multivariate adaptive regression splines," *The Annual of Statistics*, vol. 19, no. 1, pp. 1–67, March 1991.

[74] G. Jekabsons, *ARESLab*: Adaptive Regression Splines toolbox for Matlab/Octave, 2011, available at http://www.cs.rtu.lv/jekabsons/.

[75] *FreePDK*, North Carolina State University [Online], http://www.eda.ncsu.edu/wiki/FreePDK.

[76] *ISCAS85 Benchmarks*. [Online]. Available: http://web.eecs.umich.edu/~jhayes/iscas.restore/benchmark.html, accessed July 14, 2016.

[77] *Cadence Soc Encounter*. [Online]. Available: http://www.cadence.com/products/di/soc_encounter/pages/default.aspx, accessed Jan. 29, 2016.

[78] *QRC Extraction Users Manual*, Cadence, Sep. 2009.

[79] Available: http://iwls.org/iwls2005/benchmarks.html, 2015.

[80] *PrimeTime*: www.synopsys.com

[81] T. Kirkpatrick and N. Clark, “PERT as an aid to logic design,” *IBM J. Res. Develop.*, vol. 10, no. 2, pp. 135–141, Jun. 1966.

[82] LEON3 processor: www.gaisler.com

[83] *Xilinx ISE*, http://www.xilinx.com/products/design-tools/ise-design-suite

[84] *Mibench benchmark*: http://www.eecs.umich.edu/mibench

[85] S. Assefa et al., “A 90nm CMOS integrated nano-photonics technology for 25Gbps WDM optical communications applications,” in Proc. IEEE Int. Electron. Devices Meet. (IEDM), Dec. 10–12 2012.

[86] *HotSpot*: www.lava.cs.virginia.edu/HotSpot

[87] R. Kwasnick, A.E. Papathanasiou, M. Reilly, A. Rashid, B. Zaknoon, and J. Falk, “Determination of CPU use conditions*,” Proc. Int. Reliability Physics Symp. (IRPS)*, 2011, pp. 2C.3.1-2C.3.6.

[88] B. Kaczer et al., "Atomistic approach to variability of bias-temperature instability in circuit simulations ", in *IEEE Int. Reliability Physics Symposium (IRPS)*, 2011, pp. XT.3.1-XT.3.5.

[89] V. Huard et al. "NBTI degradation: From transistor to SRAM arrays", in *Proc. IEEE Int. Reliability Physics Symposium (IRPS)*, 2008. pp. 289-300.

[90] A. Bansal et al., "Impact of NBTI and PBTI in SRAM Bit-cells: Relative Sensitivities and Guidelines for Application-Specific Target Stability/Performance", in *Proc. IEEE Int. Reliability Physics Symposium (IRPS)*, 2009, pp. 745-749.

[91] J. C. Lin et al., “Time dependent vccmin degradation of SRAM fabricated with high-k gate dielectris,” in *Proc. IEEE Int. Reliability Physics Symposium (IRPS)*, 2007, pp. 439-444.

[92] K. Kang et al., “Impact of negative-bias temperature instability in nanoscale SRAM array: Modeling and analysis,” *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 26, no. 8, pp. 1770-1781, 2007.

[93] A. Bansal et. al., “Impacts of NBTI and PBTI on SRAM static/dynamic noise margins and cell failure probability,” *Journal Micro. Rel.*, pp. 642-649, 2009.

[94] A. Bansal, J.-J. Kim and Rahul Rao, "Usage-based degradation of SRAM arrays due to bias temperature instability," in *Proc. IEEE Int. Reliability Physics Symposium (IRPS)*, 2012, pp. 2F.6.1-2F.6.4.

[95] P. Weckx et al., "Defect-based methodology for workload-dependent circuit lifetime projections-Application to SRAM," in *Proc. IEEE Int. Reliability Physics Symposium (IRPS)*, 2013, pp. 3A.4.1-3A.4.7.

[96] D. Angot et al., “The impact of high Vth drifts tail and real workloads on SRAM reliability,” in *Proc. IEEE Int. Reliability Physics Symposium (IRPS)*, 2014, pp. CA.10.1-CA.10.6.

[97] E. Mintarno et al, "Workload dependent NBTI and PBTI analysis for a sub-45nm commercial microprocessor", in *Proc. IEEE Int. Reliability Physics Symposium (IRPS)*, pp. 3A.1.1-3A.1.6, 2013.

[98] S. Khan et al., “Trends and challenges of SRAM reliability in the nano-scale era,” in *Proc. Design and Technology of Integrated Systems in Nanoscale Era (DTIS)*, 2010.

[99] M. Indaco et al., “On the impact of process variability and aging on the reliability of emerging memories (Embedded tutorial),” in *Proc. European Test Symposium (ETS)*, 2014.

[100] V. Huard et al., “Managing SRAM reliability from bitcell to library level,” in *Proc. IEEE Int. Reliability Physics Symposium (IRPS)*, 2010, pp. 655-664.

[101] J. Qin et al., “SRAM stability analysis considering gate oxide SBD, NBTI and HCI,” in *Proc. International Integrated Reliability Workshop (IIRW)*, 2007, pp. 33-37.

[102] A. Calimera et al., “Paritioned cache architectures for reduced NBTI-induced aging,” in *Proc. Design, Automation & Test in Europe Conf. (DATE)*, 2011.

[103] A. Gebregiorgis et al., “Aging mitigation in memory arrays using self-controlled bit-flipping technique,” in *Proc. Asia & South Pacific Design Automation Conf. (ASPDAC)*, 2015, pp. 231-236.

[104] S. Ganapathy et al., “IRMW: A low-cost technique to reduce NBTI-dependent parametric failrues in L1 data caches,” in *Proc. International Conference on Computer Design (ICCD)*, 2014, pp .68-74.

[105] E. Gunadi et al., “Combating aging with the colt duty cycle equalizer,” in *Proc. IEEE/ACM International Symposium on Microarchitecture (MICRO-43)*, 2010, pp 103-114.

[106] J. Shin et al., “A proactive wearout recovery approach for exploiting microarchitectural redundancy to extend cache SRAM lifetime,” in *Proc. International Symposium on Computer Architecture (ISCA)*, 2008, pp. 353-362.

[107] M. Basoglu et al., “NBTI-aware DVFS: A new approach to saving energy and increasing processor lifetime,” in *Proc. International Symposium on Low Power Electronics and Design (ISLPED)*, 2010, pp. 253-258.

[108] O. Khan et al., “A self-adaptive system architecture to address transistor aging,” in *Proc. Design, Automation & Test in Europe Conf. (DATE)*, 2009, pp. 253-258.

[109] A. Valero et al., “Enhancing the L1 data cache design to mitigate HCI,” *IEEE Computer Architecture Letters*, in press.

[110] G.I. Wirth et al., “Statistical model for MOSFET bias temperature instability component due to charge trapping,” *IEEE Trans. Electron Devices*, vol. 58, no. 8, pp. 2743-2751, 2011.

[111] W. Wang, et al., “Compact Modeling and Simulation of Circuit Reliability for 65-nm CMOS Technology,” *IEEE Trans. Device and Materials Reliability*, vol. 7, no. 4, pp. 509-517, 2007.

[112] E. Seevinck, F.J. List, and J. Lohstroh, "Static-noise margin analysis of MOS SRAM cells," *IEEE J. Solid-State Circuits*, vol. 22, no. 5, pp. 748-754, Oct. 1987.

[113] R. Kang, R. Joshi, and S. Nassif, "Mixture importance sampling and its application to the analysis of SRAM designs in the presence of rare failure events," in *Proc. Design Automation Conf. (DAC)*, 2006, pp. 69-72.

[114] B. Sklar and F.J. Harris, “The ABCs of Linear Block Codes,” *IEEE Signal Processing Magazine*, pp. 14-35, July 2004.

[115] A. Ricketts et al., “Investigating the impact of NBTI on different power saving cache strategies,” in *Proc. Design, Automation & Test in Europe Conf. (DATE),* 2010, pp. 592-597.