ALGORITHMS AND METHODOLOGY FOR POST-MANUFACTURE ADAPTATION TO PROCESS VARIATIONS AND INDUCED NOISE IN DEEPLY SCALED CMOS TECHNOLOGIES

A Thesis
Presented to
The Academic Faculty

by

Maryam Ashouei

In partial fulfillment
of the Requirement for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

December 2007

ALGORITHMS AND METHODOLOGY FOR POST-MANUFACTURE ADAPTATION TO PROCESS VARIATIONS AND INDUCED NOISE IN DEEPLY SCALED CMOS TECHNOLOGIES

Approved by:

Dr. AbhijitChatterjee, Advisor
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Dr. Sudhakar Yalamanchili
School of Electrical and Computer

Georgia Institute of Technology

Dr. Jeffery A. Davis
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Dr. Linda Milor
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Dr. Adit D. Singh
School of Electrical Engineering
Engineering
Auburn University

Date Approved: September 24, 2007

ACKNOWLEDGMENTS

I would like to thank my advisor, Prof. Abhijit Chatterjee, for accepting me in his research group and for his support and guidance during the course of my research. I could not have imagined a better advisor for my PhD. I am also thankful to Prof. Adit Singh. His continuous help and insights had a significant impact on the research presented here. I also would like to thank my committee members, Dr. Sudhakar Yalamanchili, Dr. Jeff Davis, and Dr. Linda Milor for taking the time to serve on my proposal and defense committees.

I would like to acknowledge the help and support of my past and present lab-mates. I need to especially thank Utku Diril for patiently answering many of my questions and helping me even after he graduated from Georgia Tech.

I would also like to thank Gigasale Research Center (GSRC) and National Science Foundation (NSF) for funding this research and providing different venues for interacting with industry experts and getting their valuable inputs.

Finally, I would like to thank the members of the department staff (in particular from the administrative, financial, and IT offices) for their extraordinary patience and diligence.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

In the last two decades, VLSI technology scaling has spurred a rapid growth in the semiconductor industry. With CMOS device dimensions falling below 100 nm, achieving higher performance and packing more complex functionalities into digital integrated circuits have become easier. However, the scaling trend poses new challenges to design and process engineers. First, larger process parameter variations in the current technologies cause larger spread in the delay and power distribution of circuits and result in the parametric yield loss. In addition, ensuring the reliability of deep sub-micron (DSM) technologies under soft/transient errors is a significant challenge. These errors occur because of the combined effects of the atmospheric radiations and the significantly reduced noise margins of scaled technologies.

This thesis focuses on addressing the issues related to the process variations and reliability in deeply scaled CMOS technologies. The objective of this research has been to develop circuit-level techniques to address process variations, transient errors, and the reliability concern. The proposed techniques can be divided into two parts. The first part addresses the process variation concern and proposes techniques to reduce the variation effects on power and performance distribution. The second part deals with the transient errors and techniques to reduce the effect of transient errors with minimum hardware or computational overhead.

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

Technology scaling, following Moore's law, brought new challenges to process, design, and test engineers in DSM era. As transistor dimensions shrink and more functionality are packed into an integrated circuit, huge power consumption, considerable process parameter variations, and susceptibility to transient errors and noise are some of the many challenges facing the industry. The problems at hand need to be addressed at different levels of IC production: process level, circuit level, architecture level, physical-design level, and debug and production test level. The objective of this research is to address some of the aforementioned challenges. A goal of this research was to address the impact of process variations and to provide circuit-level techniques to mitigate the impact. Another concern addressed in this thesis is reliability issues and susceptibility of deeply scaled CMOS circuits to transient errors.

Process variations are significant in the DSM technologies because of processing and masking limitations. In current technologies, they can result in up to 20X variation in the leakage power and 1.3X variation in the circuit delay. The trend is increasing with scaling. Such large variations in circuit performance and leakage power negatively impact the manufacturing yield. Therefore, techniques are needed to alleviate the effects of such variations. Research on the effects of large process parameter variations can be categorized into two groups. The first group aims at estimating the leakage power and delay distributions of circuits under large process variations. The second group tries to mitigate the effects of large process variations by making the circuit more variation-tolerant. This is done either by applying different circuit optimization techniques such as gate sizing, V_{dd} scaling, and V_t modulation or by applying post-manufacture circuit tuning techniques involving the application of forward or reverse body bias to bring the delay and leakage power of the circuit within an acceptable range. In this thesis, we propose a post-manufacture tuning methodology, which reduces the power and performance variations and results in significant parametric yield improvement. We also look at the effect of placement strategies on power and performance variation. We show that with careful placement, we can achieve considerable reduction in leakage variation with little impact on circuit delay.

In addition, DSM circuits are more vulnerable to noise and soft-error sources than before. The reasons for the increase in susceptibility lie in the use of lower supply voltage, smaller transistor sizes, and shorter depth of pipeline stages in modern VLSI circuits. Traditionally, memory elements and flip flops were the ones susceptible to transient errors. It has been projected that the soft-error rate of combinational circuits will approach that of unprotected memory elements over the next decade. The effort to deal with the reliability issue focuses both on designing soft-error-hardened flip flops and sizing combinational circuits optimally to achieve soft-error tolerance. In this research, a partial correction technique for minimizing the effect of transient errors is introduced for applications that can tolerate some degree of accuracy as long as the system-level quality of service (QoS) metrics are satisfied.

The objective of the proposed research is to develop techniques to address the delay and leakage variations as well as the reliability issue related to transient errors in DSM circuits.

1.2 THESIS ORGANIZATION

In this work, the effect of process parameter variation on power and performance is addressed. The work also concentrates on reducing the impact of increasing transient error on system level performance.

Chapter 2 provides an overview of process variations, their sources, the trends, and their impacts on circuits. The chapter also provides a summary of previous work to curb the impact of process variation on circuit level power and performance characteristics.

Chapter 3 introduces placement-based techniques for leakage variability optimization in the DSM circuits: These include algorithms for the placement of gates in a dual-V_t circuit to mitigate the large leakage variation. The goal is to reduce the leakage variation caused by the correlated within-die process variations with little impact on circuit delay. In addition, an efficient approach for statistical estimation of the leakage power variation caused by correlated within-die process variations is developed.

Chapter 4 provides a post-manufacture leakage and performance tuning methodology for scaled CMOS technologies: Here, specific hardware tuning “knobs” (control mechanisms) such as tunable gates that can operate in either low-speed/low-power mode or high-speed/high-power mode are introduced to deal with the delay and leakage variation. These control mechanisms are actuated by tests that implicitly measure the delay and leakage power dissipation values of embedded logic circuits. A hardware framework that can support such self-adaptation is developed and algorithms are designed for optimizing the various enabling hardware design parameters.

Chapter 5 presents an overview of the scaling impact on susceptibility of DSM circuitry to soft errors and noise. The chapter also provides a summary of previous work on how to alleviate the problem.

Chapter 6 provides a partial correction technique for minimizing the effect of transient errors: Here, the focus is on developing partial correction techniques for applications that can tolerate some degree of accuracy as long as the system-level QoS metrics are satisfied or degraded within acceptable levels. The proposed technique must impose little delay, power, and area overheads and must perform the correction in the real time.

Chapter 7 highlights the main contributions of the thesis and provides directions of future work.

CHAPTER 2

PROCESS VARIATIONS AND ITS IMPACT ON CIRCUIT

Digital circuits experience two types of variations, namely, environmental variations and physical variations [1]-[2]. Environmental variations that occur during the operation of a circuit include variations in the supply voltage, temperature, and switching activity. A physical variation is due to the processing and masking imperfections and affects both transistors and interconnects [1]. The focus of this work is on physical variations and that how they affect different transistor characteristics. Next, different components of variations are described. The effect of process variations on circuit delay and leakage is explained. Then, a survey of techniques to estimate the delay and leakage power distributions and techniques to mitigate the variation effects are presented.

2.1 PROCESS VARIATION

Stine [3] considers the components for process variations: wafer-to-wafer variations, within-wafer variations, and within-die variations. Within-wafer variations, also called die-to-die or inter-die variations, affect all devices on a die in the same way and are usually modeled as a shift in the circuit device parameters such as V_t. Within-die variations, also called intra-die or across-die variations, consist of systematic and random variations. While systematic variations are deterministic and generally caused by lithography or chemical and mechanical polishing (CMP) [4], random variations such as dopant fluctuations are not predictable. Random variations could be correlated, which affect devices in close proximity of one another in a correlated way or could be independent, which affect individual devices on a die.

The inter-die and intra-die variations of the inter-level dielectric thickness (T_{ILD}) after oxide decomposition and chemical and mechanical polishing (CMP) are shown in

Figure 1. It can be seen that while the die-to-die variation is smooth, there is a large within-die variation.

The nominal values and 3δ variations of the effective channel length (L_{eff}), the gate oxide thickness (T_{ox}), and the threshold voltage (V_t) for different technology generations are shown in Table 1 [2]. The variation is defined as the ratio of 3δ to the nominal value. From the table, it is evident that the variability is increasing with technology generations. Furthermore, it can be seen that while T_{ox} and V_t observe a moderate variation increase, L_{eff} experiences a large variation increase. The within-die portion of the process variation is also increasing, for instance, with technology scaling, the channel length variation caused by the within-die variation raised from 40% to 65% [2].

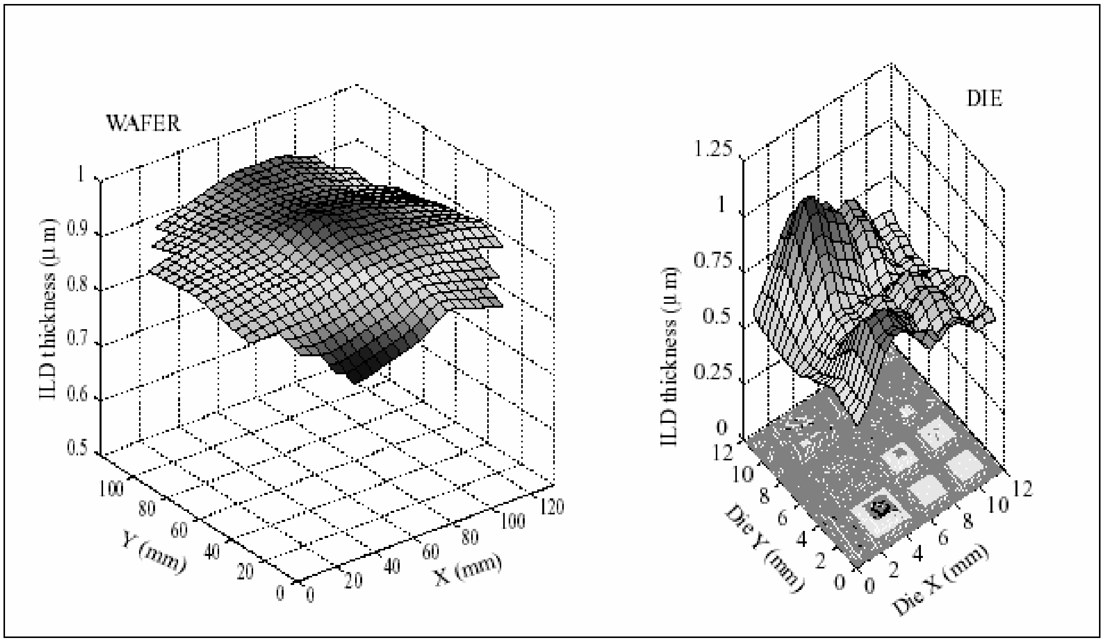


Figure 1. Variation in the ILD thickness across the wafer (left) and across the die (right)

(Courtesy of [5])

Table 1. Technology parameter variations

Parameter	1997	1999	2002	2005	2006
L_{eff} (nominal)	250 *nm*	180 *nm*	130 *nm*	100 *nm*	70 *nm*
L_{eff} (3δ)	80 *nm*	60 *nm*	45 *nm*	40 *nm*	33 *nm*
L_{eff} (variation)	32 %	33 %	35 %	40 %	47 %
T_{ox} (nominal)	5 *nm*	4.5 *nm*	4 *nm*	3.5 *nm*	3 *nm*
T_{ox} (3δ)	0.4 *nm*	0.36 *nm*	0.39 *nm*	0.42 *nm*	0.48 *nm*
T_{ox} (variation)	8 %	8 %	9.8 %	12 %	16 %
V_t (nominal)	0.5 *V*	0.45 *V*	0.4 *V*	0.35 *V*	0.3 *V*
V_t (3δ)	50 *mV*	45 *mV*	40 *mV*	40 *mV*	40 *mV*
V_t (variation)	10 %	10 %	10 %	11 %	13.3 %

2.1.1 Process Variation Effect on Transistor Characteristics

Process variations affect a CMOS transistor in terms of its geometric characteristics and its material parameters. The geometric variations consist of the T_{ox} variation, the result of *film thickness variation*, and *lateral dimension* (the channel length and device width) variations. The T_{ox} variation is well behaved and generally observed from wafer to wafer. Variations in the lateral dimensions are caused by the photolithography proximity effect, mask, lens, as well as photo system deviations, and plasma etch dependencies [5].

The variation in material parameters is most significantly observed in the channel doping deviation, which results in the threshold voltage variation [5]. The variation in transistor parameters affects its speed and power characteristics and might result in a design that does not meet the target specifications in terms of delay and power characteristics.

2.1.2 Process Variation Impact on Circuit Delay and Leakage Power

Process parameter variations cause a large spread in the delay and leakage distributions of circuits. The frequency and standby leakage current (I_{sb}) of different microprocessor dies across a manufactured wafer are shown in Figure 2. There is a 30% variation in the frequency and a 20X variation in the standby leakage current [6]. Dies with a high frequency and high leakage power consumption must be discarded. Dies with an acceptable standby leakage are binned based on their frequencies and are priced accordingly. The large variation in the standby leakage current is mainly due to variation in the sub-threshold leakage, which is the main contributor to the leakage current. The sub-threshold leakage is expressed as:

$$I_{sub-vth} = I_s e^{\frac{V_{gs}-V_t}{nV_T}} (1 - e^{\frac{V_{DS}}{V_T}}) \qquad (1)$$

where I_s is a circuit and process-dependent constant, n is the sub-threshold swing coefficient, V_T is the thermal voltage, and V_t is the threshold voltage.

Because of the inverse exponential relationship between the sub-threshold leakage and the threshold voltage, a small variation in V_t results in a large variation in the leakage current. Also, in the high-performance CMOS design, the leakage power consumption can be responsible for 40% or more of the total power consumption of the circuit. To address the high leakage consumption, a dual-V_t process is used, where high-V_t gates are used along the off-critical circuit paths.

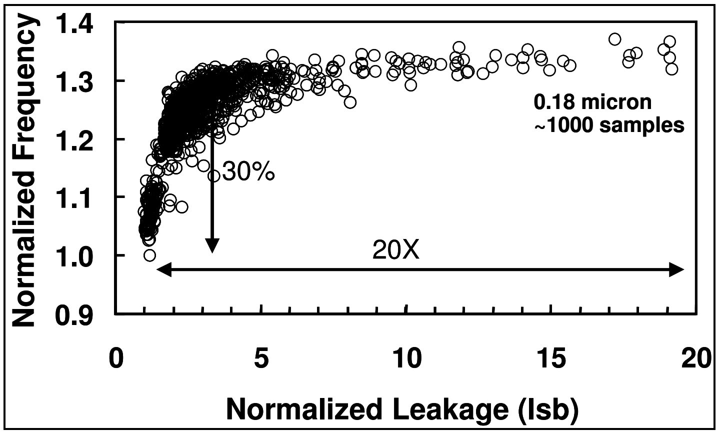


Figure 2. Leakage and frequency variations (courtesy of [6])

2.2 VARIATION-TOLERANT DESIGN

Techniques to narrow the leakage and timing distributions can be divided into two groups, namely the post-manufacture techniques and the design-level optimization techniques.

2.2.1 Post-Manufacture Variation-Tolerant Techniques

Adaptive Body Bias

The use of substrate biasing to modulate the threshold voltages (V_t) is a key post-manufacture technique to reduce the leakage and performance variation. V_t is a function of the source-bulk voltage, as shown below [8]:

$$V_t = V_{T_0} + \gamma(\sqrt{|-2\phi_F + V_{SB}} - \sqrt{-2\phi_F}) \qquad (2)$$

where V_{T0} is the threshold voltage at $V_{SB} = 0$, V_{SB} is the source-bulk (substrate) voltage, γ is the body-effect coefficient, and ϕ_F is the substrate Fermi potential.

The substrate is normally grounded. For an NMOS device, a negative bias, called reverse body bias (RBB), causes V_t to increase. On the contrary, a positive bias, called forward body bias (FBB), reduces V_t. In the case of a PMOS device, the substrate is normally tied to V_{dd} and a voltage lower (higher) than the V_{dd} is used as RBB (FBB).

As the leakage power consumption became more pronounced, adaptively changing the bias voltage was considered as a promising technique to reduce the sub-threshold leakage current in the standby mode while maintaining the circuit performance in the active mode. The idea is to reduce the circuit leakage while the circuit is idle by applying an RBB to increase the effective threshold voltage, hence reducing the sub-threshold leakage.

Alternatively, an FBB may be applied during the active mode to improve circuit performance and withdrawn when the circuit is idle to control the sub-threshold leakage. If applying the FBB in the active mode, the process threshold voltage could be higher than the targeted one for that technology.

In recent years, the bias techniques are also utilized to tackle the large process variations observed in the circuit power and performance in DSM circuits [6], [9]. It has been shown that these techniques can be used to tackle both the die-to-die and within-die process variations. The adaptive body bias (ABB), as the name suggests, adaptively adjusts the bias value of a die according to the variation observed by the die. This mitigates the die-to-die process variation. An RBB is applied to dies that are unnecessarily fast to control the leakage. Conversely, slow devices can satisfy the performance target using an FBB (Figure 3). As is shown in Figure 3, the ABB reduces variations in F_{MAX} by moving the operation frequency of fast dies to the left using an RBB and moving the frequency of slow dies to the right using an FBB. As a result, ABB narrows the performances distribution of different circuits. In [9], it is shown that using a single PMOS/NMOS bias combination per die can reduce the die frequency variation by a factor of 7.

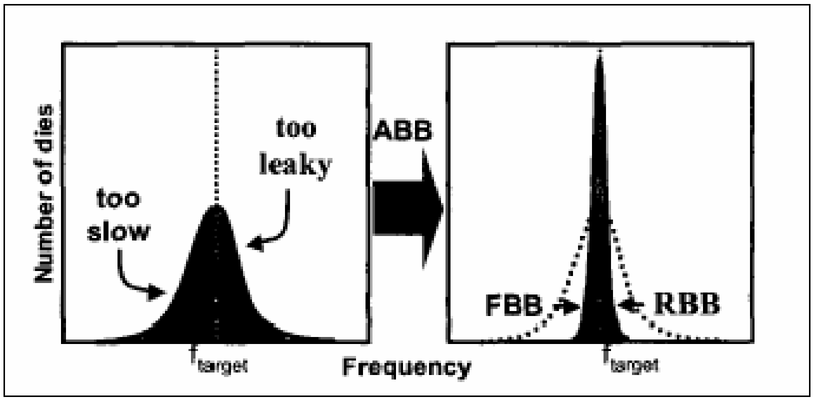


Figure 3. Performance variation reduction using ABB (courtesy of [6])

The ABB technique does not compensate for the within-die process variations since it provides a single NMOS/PMOS bias combination for all the blocks in a circuit. In [9], an improved ABB, called within-die ABB (WID-ABB) was proposed. This technique allows large blocks in a circuit to have their own body bias values, which control the delay and leakage within each individual block. The WID-ABB reduces the delay variation by a factor of 3 compared to the ABB technique and makes it possible for most dies to be accepted in the highest frequency bin. One concern about the WID-ABB is the complex bias generation and distribution circuitry. Furthermore, to be able to bias NMOS devices in different blocks to different bias values, multiple wells are necessary. In [9], a triple-well process was used. Another issue regarding the use of ABB is that the reverse body bias loses its effectiveness as the technology scales [10] and the junction tunneling leakage starts to become significant by applying an RBB. It is shown in [10] that there is an optimum RBB value that is unique for every technology generation. This optimum value is decreasing by approximately 2X every technology generation. This will reduce the effectiveness of the RBB technique on the sub-threshold leakage current by 4X every technology generation.

Adaptive Voltage Scaling

Adjusting the circuit supply voltage, also called adaptive voltage scaling (AVS), is a well-studied technique for optimizing the circuit power consumption (mainly the active power) under delay constrains ([11], [12]). The most common form of AVS is the use of two supply voltages. The authors in [13] have proposed the AVS as a post-manufacture tuning technique to reduce variability in the circuit delay and power. They show that the AVS is as effective as the ABB in reducing performance and power variations. Deciding between the two techniques depends on design requirements such as the extra design complexity, voltage generation and regulation, and reliability. Moreover, it is shown in [13] that applying both the ABB and the AVS do not provide a significant improvement in reducing performance variability, while [14] claims otherwise.

Self-Calibrating Dynamic Circuits

The large leakage consumption of CMOS devices requires larger keeper transistors for dynamic logic to be able to hold the logic values between two pre-charge intervals. At the same time, the large leakage variation makes the choice of the right keeper size a difficult task. If the size is chosen to be too small, many dies do not operate reliably. If it is chosen to be too large to compensate for dies with larger leakage, it unnecessarily slows down dies with lower leakage. The concept of variable-size keeper, which can be programmed based on the die leakage, was proposed in [15]. A leakage sensor was proposed in [15] that can be used on multiple locations of a die to estimate the leakage in different regions of the die. The leakage information, obtained by the sensors, can be used to choose the best keeper size among an array of available sizes. The variable-size keeper approach, implemented on a 2-read, 2-write ported 128×32b register file at 90 nm CMOS, has shown to improve the performance by 10%. The overhead associated with this technique is that as the within-die process variation increases, more leakage sensors must be integrated on a die to have a good estimate of the leakage for different regions. The second concern is the routing overhead necessary for programming the variable-size keeper. The third concern is that this approach could be only applied to dynamic logic and cannot be generalized to other classes of logic design.

2.2.2 Variation-Tolerant Optimization Techniques

The focus of the previous section was on post-fabrication tuning techniques to compensate for process variations. This section focuses on circuit optimization techniques for designing variation-tolerant VLSI circuits. Typically, different circuit parameters such as the threshold voltage, supply voltage, and transistor sizing are used to optimize a circuit in terms of its performance and power consumption. There have been numerous articles that optimize circuit power consumption given a set of delay constraints or vise versa [16]-[20]. In these studies, the convention is to apply a static timing analysis using the nominal gate delay. However, because of the excessive variations in the DSM technologies, many dies that are optimized using the nominal case will fail the timing or power constraints. One solution could be to use the worst-case gate delay. This will improve the expected yield at a high power and area cost.

Recently, efforts have concentrated on the power-performance optimization while considering process variations. The newly proposed techniques use a statistical timing analyzer instead of using a static timing analyzer [21]-[24]. All these techniques focus on the gate sizing, with the exception being the technique in [22], which performs the gate sizing and dual-V_t assignment. The differences among the techniques are in the approximation of the statistical timing analysis and their optimization methods.

In [21], a Lagrangian-based relaxation is proposed to size different transistors considering the inter-die and within-die variations. The objective is to guarantee that the delay requirement is met with a certain degree of confidence while keeping the area and power within a given set of constrains. The complexity of this method is linear. The result shows a 19% area/power savings compared to the worst-case analysis.

The authors in [22] use a simplified sensitivity-based heuristic technique for dual-V_t assignment and gate sizing, with the objective of reducing the leakage power. First, they find a second-order polynomial relation between the gate delay and the gate channel length using SPICE. Similarly, the gate leakage is presented as an exponential function of the gate channel length. They also define two statistical sensitivity metrics for each gate. One is the sensitivity of a gate to changes in the threshold voltage and the other is the sensitivity to changes in the gate size. Using the sensitivity metrics, the algorithm tries to find the best gates to have high V_t and the best gates to be sized up such that the leakage and its variation are minimized. The worst-case complexity of this approach is $O(n^3)$. It is shown that the leakage power can be reduced by 15-35% compared to the deterministic analysis.

In [23], another statistical gate sizing technique is presented with the objective of reducing the delay variation. The authors introduced an efficient way of finding the mean and variance of the maximum of random variables. This fast statistical parameter extraction is used along with a more detailed statistical timing analysis to perform gate sizing. This technique can reduce the delay variation by an average of 72% at the cost of a 20% increase in design area.

While [21]-[24] focus on either leakage or delay optimization, the technique in [25] performs yield enhancement with simultaneous delay and leakage constrains. This technique utilizes a non-linear optimizer, which uses the gradient of yield with respect to gate sizes to perform yield optimization. A 40% yield improvement compared to the deterministic approach was reported.

2.3 ESTIMATING DELAY AND LEAKAGE VARIATIONS

With large variations in the circuit delay and power, it is necessary to have an accurate estimate of these metrics at the design stage. Traditionally, the circuit design was based on the corner cases. The worst cases for the delay and power were used. But in the future technologies, with more process variations, the worst-case design could be too pessimistic and result in a very expensive solution. Therefore, efforts have been made to estimate the delay and leakage power as accurately as possible by considering both the within-die and die-to-die process variations. In this subsection, a survey of such techniques is presented and the shortcomings of each technique are discussed briefly.

2.3.1 Leakage Power Estimation Techniques

Most of the research in the area of leakage power estimation focuses on the evaluating the sub-threshold leakage power. These techniques find a mathematical representation of leakage power distribution using the well-known relation between the sub-threshold leakage power and the device channel length or threshold voltage. Few studies aim at estimating all the components of leakage power [44]. This is a major shortcoming, as with technology scaling, the tunneling leakage will become more significant and can no longer be ignored.

In [26], the authors present a mathematical model for predicting the sub-threshold leakage power considering the within-die threshold voltage variation:

$$I_{leak} = \frac{I_p{}^o w_p}{k_p} e^{\frac{\delta_p{}^2}{2\lambda_p{}^2}} + \frac{I_n{}^o w_n}{k_n} e^{\frac{\delta_n{}^2}{2\lambda_n{}^2}} \qquad (3)$$

where w_p and w_n are the total PMOS and NMOS device widths in the chip, k_p and k_n are factors that determine the percentage of PMOS and NMOS device widths in the off state, and $I_p{}^o$ and $I_n{}^o$ are the expected mean sub-threshold leakage current per unit width of PMOS and NMOS devices in a particular chip. δ_p and δ_n are the standard deviation of the channel length variation within a particular chip. λ_p and λ_n are constants that relate the channel length of PMOS and NMOS devices to their corresponding sub-threshold leakages. The leakage calculated using this model is within ±20% of the actual sub-threshold leakage for more than half of the dies under test.

The authors in [27] only address the sub-threshold leakage current caused by the channel length variations. They first find the sub-threshold current as a function of channel length in the form of

$$I = h(L) = q_1 \exp(q_2 L + q_3 L^2) \qquad (4)$$

where q_1, q_2, and q_3 are fitting parameters found through SPICE simulations. These parameters change depending on the gate type. Therefore such a function must be found for each individual gate used in a circuit. Assuming that the channel length variation has a normal distribution, *I,* as defined in (4), is a lognormal function. The sub-threshold leakage current of a circuit is the sum of sub-threshold leakage current of its individual gates. Since the sum of lognormal distributions is also lognormal, the full-chip leakage current is lognormal. The authors find the mean and standard deviation of the full-chip leakage using the corresponding values for each individual gate. The authors extend this work in [28] to include multiple independent process parameter variations, namely, the channel length, doping concentration, and oxide thickness. The gate tunneling leakage was also estimated in [28]. In [29], the authors extend the work in [28] to include the correlated within-die variation. A similar technique is proposed in [30]. These techniques cannot be extended to variations with non-normal distributions, which is the case for environmental variations. Another drawback of these techniques is that they cannot be used when the process parameter variations are not independent random variables. The technique in [31] provides an estimate of the sub-threshold leakage current considering the variations in the channel length, temperature, and supply voltage.

2.3.2 Delay Estimation Techniques

Many statistical timing analysis techniques have been proposed in recent years [32]-[37]. In theory, a statistical timing analysis is similar to a static timing analysis where the delay of each gate/interconnect has a probability distribution function instead of a nominal value. Two statistical operations are needed in the statistical timing analysis: adding two probability distributions and finding the maximum of two probability distributions. The complexity of the analysis is in computing these two operations. Although these operations can be done easily, the problem size grows exponentially with the circuit size. One way to reduce the computation time is to find a bound on the probability distribution function [33], [34], [36]. For example, [36] uses a methodology based on Bayesian networks for computing the exact probability distribution of the circuit delay. This method is impractical for large circuits, since its time complexity grows exponentially with the circuit maximum clique size. The authors also propose a technique to reduce the problem size and get a tight lower bound on the exact distribution.

Most statistical timing analysis techniques consider the gate or interconnect delay as an independent random variable. A technique proposed in [32] takes into account the correlated within-die process variation. This technique has a run time of $O(n\times(N_g+N_i))$, where n is the number of grid points and N_g and N_i are the number of gates and interconnects respectively. Considering the re-convergence fan-out and delay distribution makes the simplifications of the addition and maximum operations used in the above proposed techniques incorrect. The authors in [33] also consider the correlated within-die variation and propose a technique with a linear run time to find a bound on the probability distribution of the circuit delay. In [37], the authors consider the inter-die, intra-die, and intra-gate variations to model the gate delay variation.

CHAPTER 3

LEAKAGE-AWARE PLACEMENT FOR NANO-SCALE CMOS

In this chapter, the problem of leakage power variation minimization in the presence of spatially correlated across-die process variations is addressed. The objective is to analyze the effect of the placement of low-V_t gates in a dual-V_t design on the statistics of leakage power in the presence of correlated across-die process variations.

New placement strategies are proposed and compared with a more conventional placement method, which optimizes for the total wire length, in terms of their leakage power variations. Our proposed technique reduces the total leakage variation by reducing the variation of sub-threshold leakage current, which is currently the main component of leakage. The effectiveness of the new placement methodology is studied using the Monte Carlo simulation. It is shown that with minimal impact on delay, the placement of low-V_t gates in a layout can be performed in such a way to maximize the yield for a specified leakage power upper bound. For the obtained placement of low-V_t gates, the layout can then be optimized for other important criteria such as the wire length. Simulations are performed on ISCAS benchmarks and guidelines for distributing the low-V_t gates across the die are developed. Our results indicate that an average reduction of 17% and a maximum reduction of 31% in the leakage variation can be achieved. We also analyze the impact of the placement-based leakage minimization technique on circuit delay and wire length.

In the following, first the model of the correlated inter-die process variation models is discussed. This is followed by a discussion of the proposed layout procedure to maximize yield, focusing primarily on minimizing the leakage power variation. Next, simulation results on ISCAS benchmarks are presented. The impact of proposed placement technique on wire length and circuit timing is also analyzed. Later, we propose an effective technique for estimating the power variation caused by correlated within-die process variation. The last section presents conclusions and recommendations.

3.1 INTER-DIE PROCESS VARIATION MODELS

As explained in Section 2.1, there are die-to-die process variation and within-die process variation. The within-die process variation consists of random independent variations and (random and systematic) correlated variations. The correlated variations affect neighboring devices on a die in a correlated fashion. The model considered for the correlated within-die process variations in this research is called the cluster model. The cluster model is inspired by few industrial patterns available for within-die process variation (Figure 4 and [1]).

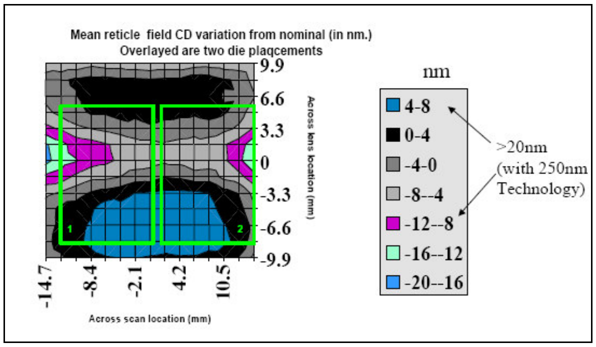


Figure 4. Within-die process variation (courtesy of Intel Corp.)

For simplicity, it is assumed that the layout is an $m \times n$ grid, each cell of which can hold exactly one gate. The assumption of placing exactly one gate in each cell of the grid causes some discrepancies with the compact physical layout of a circuit containing gates of different sizes but does not affect the overall effectiveness of the proposed approach. Below, we describe our modeling of within-die correlated variations.

<u>Definition</u>: A *cluster* is defined to be a set of gates lying within a circle of radius *r,* and the center gate, c_o.

It should be noted that the circular shape of the cluster model is not essential for what is presented here. It is just a simple way of representing correlated within-die variation, which indeed can capture the contour shape shown in Figure 4. The notion of a cluster is used to model local process variations across a die in the following manner. It is assumed that the process parameters of gates within a cluster are correlated and the correlation is specified as a function of the distance, *d*, of a gate from the center of the cluster. Every die may have one or more of these clusters at random locations. For each process parameter, there is a random variable, s_l, associated with each cluster, where $1 \leq l \leq C$ and C is the total number of clusters affecting the die. The variables s_l are statistically independent and have distributions that represents the deviation from the expected value of the process parameter of interest in the l^{th} cluster. Let the distance of the gate at the $(i,j)^{\text{th}}$ location of the layout from the l^{th} cluster center be given by $d_{i,j}(l)$. Two logic gates, one located inside a cluster and one located outside it, are shown in Figure 5.

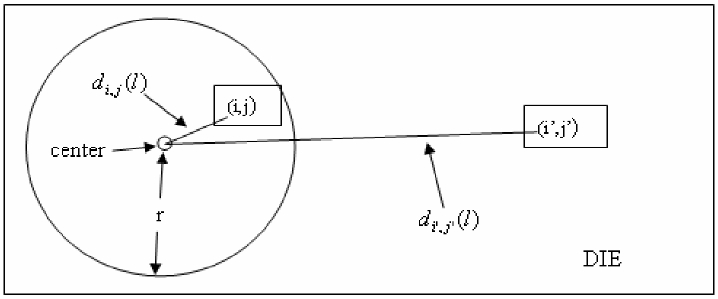


Figure 5. Cluster model

The statistical variable s_l only factors into the leakage/delay characteristics of gates inside the l^{th} cluster. The effect of s_l on the gate at the $(i,j)^{\text{th}}$ location of the layout is given by a function $f(d_{(i,j)}(l), s_l)$, where f = 0 for all gates with $d_{(i,j)}(l) > r$. In another words, $f(d_{(i,j)}(l), s_l)$ represents the variation in the process parameter of interest of the gate at the $(i,j)^{\text{th}}$ location caused by the l^{th} cluster. Two different plausible f functions are shown in Figure 6.

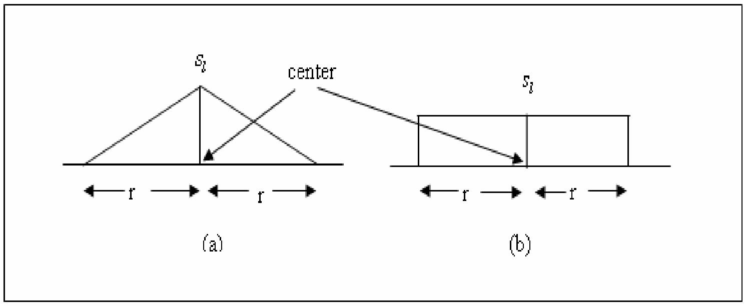


Figure 6. Two plausible functions for correlation within a cluster

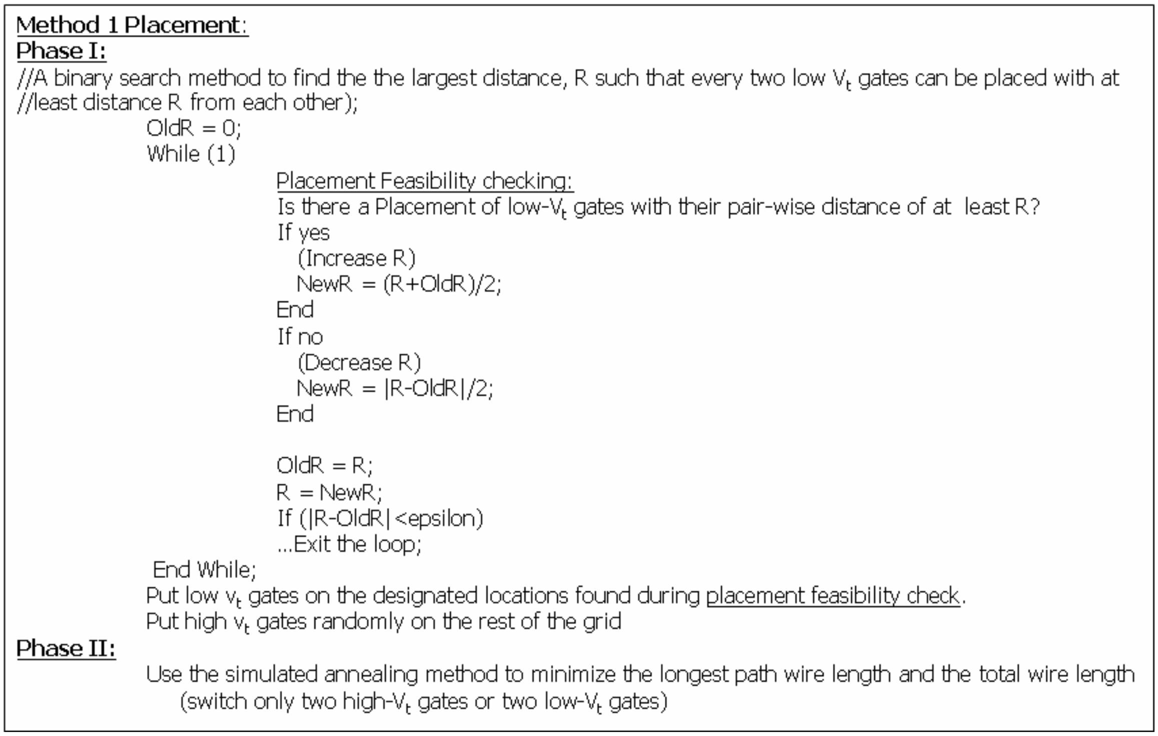
In the above, the distance between two gates is defined to be the straight-line distance between the upper-left corners of their corresponding grid cells. To perform statistical layout analysis, without loss of accuracy, the inter-die process variation is not considered since it affects all devices the same way. Therefore, its effect is independent of the placement strategy. In this work, it is assumed that the source of variation is the channel length (L_{eff}). Variations in the threshold voltage of transistors caused by the channel length variation are taken into account implicitly. In general, there can be more than one cluster. If a gate falls into the intersection of two or more clusters, the variations in L_{eff} caused by different clusters is superimposed linearly to get the final the L_{eff} value.

3.2 VARIATION AWARE LAYOUT

A common approach to minimizing leakage power dissipation is to reduce the sub-threshold leakage power and involves the use of high-V_t gates on the off-critical paths. This reduces the leakage as high-V_t devices have considerably less sub-threshold leakage than low-V_t devices due to the inverse exponential relationship between the sub-threshold leakage current and V_t referred to earlier. The goal of traditional dual-V_t optimization algorithms [38]-[40] is to use as many high-V_t gates as possible in the design without violating the overall circuit timing constraint (high-V_t gates are slower than low-V_t gates). In this work, the aim is to reduce the leakage variability in a dual-V_t design. We revisit the problem of dual-V_t assignment, while considering correlated within-die process variations for leakage optimization. The objective here is to analyze the effect of the placement of low-V_t gates in a dual-V_t design on the statistics of leakage variation in the presence of correlated within-die process variations. Below, two new placement strategies are proposed and compared against a conventional placement method, which optimizes for total wire length, in terms of leakage power variation.

Method 1: Before presenting the first proposed placement methodology, first we explain the intuition inspiring this method. For the same L_{eff} variation, low-V_t gates experience more leakage variation than their high-V_t gate counter-parts. Furthermore, a significant portion of intra-die channel length variation is systematic correlated variation caused by lens aberration and chemical and mechanical polishing imperfection [4]. Therefore, the more low-V_t gates are in a region of a die, the more variation in the (sub-threshold) leakage power is observed if the region is affected by a cluster of correlated within-die process variation. Using this intuition, a new placement method is proposed that distributes the low-V_t gates on the die as evenly as possible by maximizing their pair-wise distance. This ensures that irrespective of where a cluster (defined in Section 3.1) is located on the die, the expected number of low-V_t gates affected by the cluster is the same across the die.

The proposed placement algorithm has two phases. During the first phase, the location of low-V_t gates on the grid is determined using a binary search procedure to maximize the pair-wise distance of low-V_t gates. After finding the relative positions of low-V_t gates on the grid, the low-V_t gates are randomly placed on the allowed positions and the high-V_t gates are randomly placed on the rest of the grid. The second phase of the algorithm utilizes a simulated annealing procedure, a common placement technique used in physical design [41], to minimize the longest path wire length. Compare to the conventional simulated annealing-based placement, an extra restriction is imposed during the second phase of our proposed method. The restriction is that the annealing procedure cannot swap a low-V_t gate and a high-V_t gate with each other, i.e., only two low-V_t gates or two high-V_t gates can swap their positions in the layout grid. This guarantees that the pair-wise distance between the low-V_t gates stays invariant during the simulated annealing phase. The details of the method are presented in Figure 7.



```
Method 1 Placement:
Phase I:
//A binary search method to find the the largest distance, R such that every two low $V_t$ gates can be placed with at
//least distance R from each other);
            OldR = 0;
            While (1)
                        Placement Feasibility checking:
                        Is there a Placement of low-$V_t$ gates with their pair-wise distance of at  least R?
                        If yes
                           (Increase R)
                           NewR = (R+OldR)/2;
                        End
                        If no
                           (Decrease R)
                           NewR = |R-OldR|/2;
                        End

                        OldR = R;
                        R = NewR;
                        If (|R-OldR|<epsilon)
                        ...Exit the loop;
             End While;
            Put low $v_t$ gates on the designated locations found during placement feasibility check.
            Put high $v_t$ gates randomly on the rest of the grid
Phase II:
            Use the simulated annealing method to minimize the longest path wire length and the total wire length
               (switch only two high-$V_t$ gates or two low-$V_t$ gates)
```

Figure 7. A binary search based method to distributed the low-V_t gates evenly across the die

Method 2: The second layout strategy places all the low-V_t gates in one area of the die as close to each other as possible. The simple intuition behind this method is that if a cluster hits the low-V_t area, a large variation of leakage power will be observed. But if the variation does not hit the low-V_t gate region, the variation in leakage power is small. When the ratio of low-V_t gates to high-V_t is not high, the probability that the single low-V_t region is affected is small, and hence most of the dies observe a very small variation in their leakage power.

In the experimental result of this chapter, we compare the two proposed placement strategies with a conventional placement method, which places gates to minimize the total wire length (called method 3 from here on). The distribution of low-V_t gates across the layout for circuit c432 using the three placement methods are shown in Figure 8.

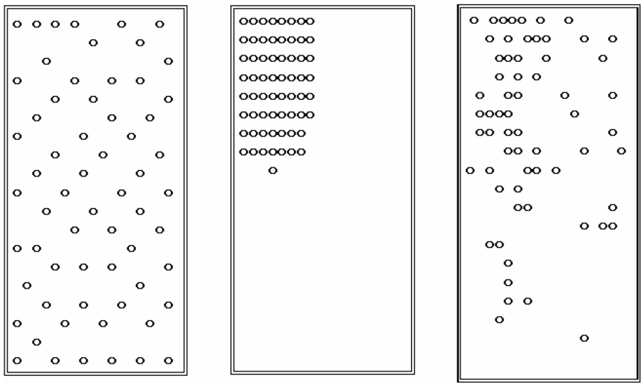


Figure 8. Three placements of low-V_t gates (C432 circuit). Left to right: method 1, 2, and 3

3.3 ESTIMATION OF CORRELATED INTER-DIE LEAKAGE POWER VARIATION

This section presents a method to estimate the distribution of the correlated leakage power variation under the cluster model. The technique takes the following information as inputs and estimates the leakage power of the circuit using a simple counting method.

- The layout of the circuit
- The number of clusters and their size

- The deviation in the leakage power of each gate, used in the synthesized circuit, for *K* different values of the process parameter of interests such as channel length. (The *K* different channel length variations are obtained by equally dividing the range of channel length variation to *K* segments and computing the deviation in the leakage power for only one fixed point in each segment) using SPICE.

Such an estimator can be used to choose the best placement for a given leakage and delay limit as shown in Figure 9.The rest of this section describes the technique for computing correlated leakage power.

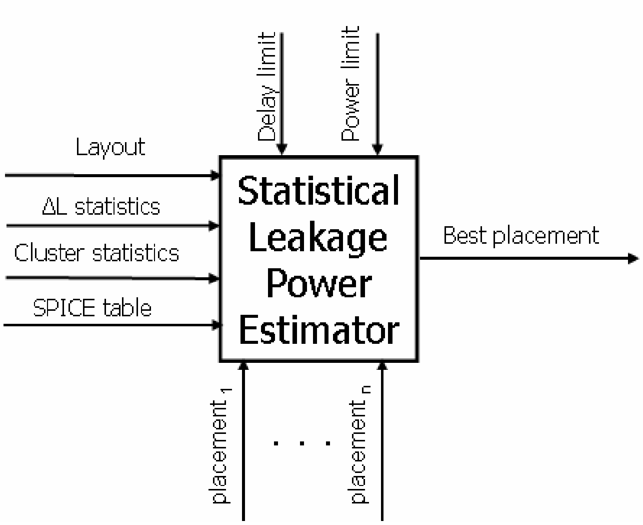


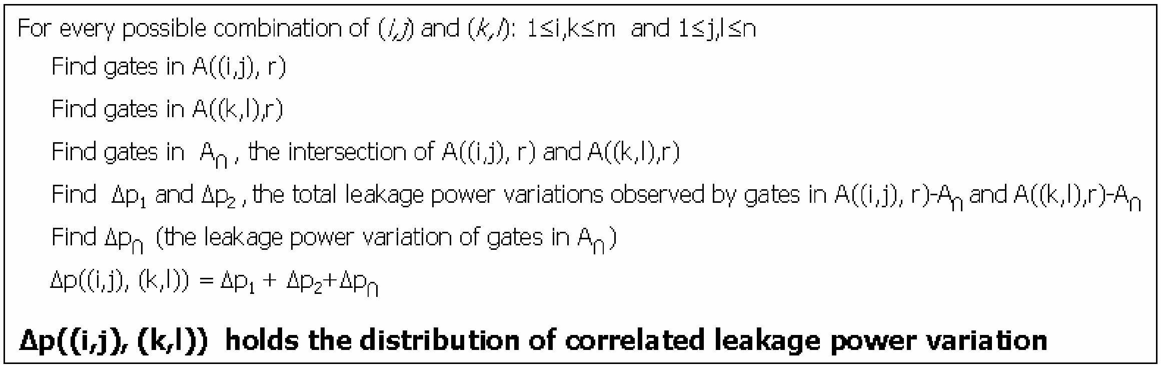
Figure 9. Best circuit placement using correlated leakage power estimator

3.3.1 An Enumerating Technique for Correlated Intra-Die Leakage Variation

Based on the cluster model, every *c* points on the grid can be the centers of *c* co-existing clusters with the same probability. Let *Area*((*i,j*), *r*) be the cluster area centered at the (*i,j*) th location. The technique for computing leakage power variation caused by the correlated within-die variation for the case of two clusters is shown in Figure 10. In

Figure 10, (*i,j*) and (*k,l*) are the centers of two co-existing clusters with variations $\Delta l_{(i,\,j)}$ and $\Delta l_{(k,\,l)}$, respectively.

In the case of *c* clusters, there are $(n \times m)^c$ combinations of *c* co-existing clusters and a method similar to the one shown in Figure 10 can be used to find the total power variation. The difference is that the intersection of every *k* clusters, $2 \le k \le c$ must be found. The time complexity of this technique is $O((n \times m)^c)$, where $n \times m$ is the grid size and *c* is the number of clusters. Hence the time complexity of the method is exponential with the number of clusters. That makes the method inapt for a large number of clusters. The following subsection introduces a quadratic-time approximation method for estimating the correlated leakage power variation.



```
For every possible combination of (i,j) and (k,l): $1 \le i,k \le m$ and $1 \le j,l \le n$
    Find gates in A((i,j), r)
    Find gates in A((k,l),r)
    Find gates in $A_\cap$, the intersection of A((i,j), r) and A((k,l),r)
    Find $\Delta p_1$ and $\Delta p_2$, the total leakage power variations observed by gates in A((i,j), r)-$A_\cap$ and A((k,l),r)-$A_\cap$
    Find $\Delta p_\cap$ (the leakage power variation of gates in $A_\cap$)
    $\Delta p$((i,j), (k,l)) = $\Delta p_1 + \Delta p_2 + \Delta p_\cap$
$\Delta p$((i,j), (k,l)) holds the distribution of correlated leakage power variation
```

Figure 10. Finding correlated leakage power distribution for 2-cluster case

3.3.2 A Quadratic-Time Approximation of Correlated Intra-die Leakage Power Variation

This section explains an approximation method to find the distribution of leakage power variation in quadratic time. Before presenting the method, a few simplifications utilized in the quadratic time estimation are described below.

When there is more than one cluster affecting a circuit, their effects in the overlapped region of clusters is considered individually. I.e. the channel length of gates that fall in the intersection of two or more clusters is not considered to be the superposition of the channel length variation caused by each of the intersecting clusters. Instead, the variation caused by each cluster is considered separately and the corresponding leakage power variation is measured. For instance, if a gate, *g,* falls in the intersection of cluster 1 and cluster 2 with the channel length variation of Δl_1 and Δl_2, respectively, then the channel length variation observed by the gate *g* is $\Delta l_g = f(\Delta l_1, d_1) + f(\Delta l_2, d_2)$. The leakage power variation observed by *g* corresponding to the channel length variation of Δl_g must be used. Using this simplification, the sum of leakage power variations corresponding to the channel length variation $f(\Delta l_1, d_1)$ and $f(\Delta l_2, d_2)$ is used (the function *f* is explained in Section 3.1). This will introduce some error in the method, which depends on the relative size of the clusters and the grid. If the cluster size is small compared to the size of the grid, the probability of two clusters intersecting is small.

The second simplification is to categorize gates based on their leakage power consumptions. An average gate can represent gates falling in the same leakage category. For instance, in the case of dual-V_t circuits, low-V_t gates can be represented by an average low-V_t gate. Similarly, all high-V_t gates can be represented by an average high-V_t gate. Using this approximation, the layout information does not need to provide the exact gate located at any grid point. It is enough to know the leakage category of each gate. This simplification frees the method from finding the exact gates falling into any cluster. It is sufficient to find how many gates from each category fall into a cluster.

Lastly, square-shaped clusters with sides of *2r*×*2r* are assumed. Next, the approximation method is explained in the context of a dual-V_t circuit. The method is general and can be applied to any circuit, and the gates can be categorized into more than two categories.

Let *L* be a matrix corresponding to the layout of the circuit, with ones in the position of low-V_t gates and zeros elsewhere. Conversely, let *H* be a matrix with ones in the position of high-V_t gates and zeros elsewhere. Let *h* be the correlation kernel, a *2r*× *2r* matrix of ones, which represents the cluster of size *r* (based on the assumption of square clusters and equal effect within the cluster). Let $h_{i_o j_0}$ be an $m \times n$ matrix, which has *h* centered at (i_0,j_0) and zeros everywhere else. C_L is defined as the correlation matrix of *L* and *h* as follows:

$$C_L = [C_L(i_0, j_0)] = [\sum_{i=1,\, j=1}^{i=m,\, j=n} h_{i_0 j_0}(i,j) \times L(i,j)] \qquad (5)$$

Similarly, C_H is defined as the correlation matrix of *H* and *h*. The $(i,j)^{\text{th}}$ element of C_L (C_H) is the number of low-V_t (high-V_t) gates within the cluster whose center is the gate at the $(i,j)^{\text{th}}$ location. This idea is explained in Figure 11.

$$L=\begin{bmatrix}1&1&0&1\\0&0&1&0\\0&1&0&1\\0&0&1&0\end{bmatrix}\quad h=\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$$h_{3,2}=\begin{bmatrix}0&0&0&0\\1&1&1&0\\1&1&1&0\\1&1&1&0\end{bmatrix}\quad C_L=\begin{bmatrix}2&3&3&2\\3&4&5&3\\1&\boxed{3}&4&3\\1&2&3&2\end{bmatrix}$$

Figure 11. An example of correlation matrix (C_L) of two matrices L and h

Let ΔL be an $m \times n$ random matrix, where ΔL (i,j) represents the variation in the channel length when the gate in the $(i,j)^{\text{th}}$ location is the center of a cluster. Let ΔP_{low} (ΔP_{high}) be an $m \times n$ matrix, where $\Delta P_{low}(i,j)$ ($\Delta P_{high}(i,j)$) has the leakage power variation of the average low-V_t gate corresponding to the channel length variation ΔL (i,j). The matrix ΔP holds samples of the correlated leakage power distribution. The step of generating the random matrix of ΔL and finding its corresponding random matrix ΔP can be repeated to give a better approximation of the leakage power variation. Below, it is explained how the method can be extended for the general case of c clusters.

$$\Delta P = [\Delta P_{low}(i,j)\times C_L(i,j) + \Delta P_{high}(i,j)\times C_H(i,j)], 1 \le i, j \le m, n \qquad (6)$$

Let $\Delta L_1, \Delta L_2, \ldots, \Delta L_c$ be c independent random matrices, where $\Delta L_l(i,j)$ is the channel length variation of the l^{th} cluster when the $(i,j)^{\text{th}}$ gate is the center of the l^{th} cluster. Also, let $\Delta P_1, \Delta P_2, \ldots, \Delta P_c$ be the correlated leakage power variations corresponding to $\Delta L_1, \Delta L_2, \ldots, \Delta L_c$ respectively. For example, ΔP_l is the variation in leakage power caused by the l^{th} cluster and $\Delta P_l(i,j)$ is the variation in the leakage power when the $(i,j)^{\text{th}}$ gate is the center of the l^{th} cluster. In the case of c clusters, the ΔP matrix, which holds the distribution of correlated leakage power variation, is the sum of c random variables ΔP_1… ΔP_c.

On the other hand, *the probability distribution function of the sum of independent random variables is equal to the convolution of the probability distribution function of each random variable*. Therefore, to compute ΔP, it is sufficient to compute the convolution of the distribution of random variables represented by ΔP_1… $\Delta P_{c.}$ The convolution operation is a quadratic-time operation. Therefore a quadratic-time method for estimating the correlated leakage power variation is obtained.

3.4 EVALUATION

To evaluate the effectiveness of the proposed methods, ISCAS 85 benchmark circuits were synthesized for speed using Synopsys Design Compiler with a library of 2-input to 4-input NAND and NOR gates and inverters. The synthesized circuits were used with SPICE 70 nm models [42] to compute the delay and the leakage power of the circuits for the 70 nm technology using a look-up table method similar to [43]. All gates had a transistor channel length of 70nm and V_{dd} of 1 V. a high V_t of 0.3 V and low V_t of 0.1 are used for high-V_t and low-V_t gates, respectively. The transistor channel length can vary within 15% of its expected. The variation in the threshold voltage caused by the channel length variation and its effect on the leakage power consumption and the delay are considered implicitly through the SPICE look-up table. It should be noted that only correlated within-die process variation is considered, since this is the part that is sensitive to the placement method. The random within-die variation affects individual gates and is not responsive to the placement strategy. Also, the leakage power measurement includes all components of leakage [44], although the reduction in variability of leakage is obtained by reducing the variability of sub-threshold leakage.

The dual-V_t assignment is obtained using a modified version of the method in [39]. In the modified dual-V_t assignment method, initially all the gates have low V_t. Then, the timing constraint is relaxed so that only a certain percentage of gates remain at the low V_t. In the results presented in this section, the percentage of low-V_t gates is 20% of the total number of gates.

To study the effectiveness of the three placement methods, circuits are first placed using one of the methods on an $m \times n$ grid, where the number of gates in the circuit is less than or equal to $m \times n$ and the grid is as close to being as square as possible. Then, the Monte Carlo simulation is used to inject clusters in different parts of the circuit. The combination of the number of clusters, *C,* and the clusters size, *r,* is referred to as a configuration (*C, r*). For small- and medium-size circuits, 2000 Monte Carlo iterations are used, while for larger circuits the simulation stopped after 500 iterations.

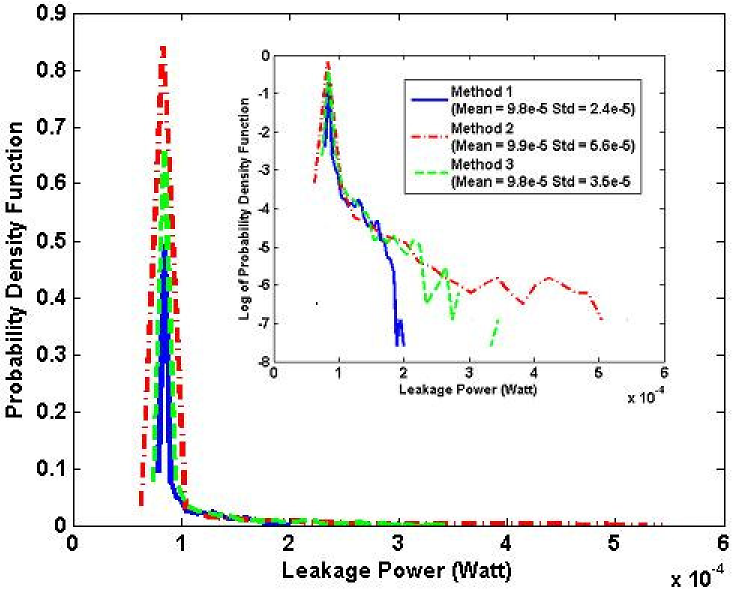


Figure 12. Distribution of the leakage power for the three placement methods

The probability distribution function (PDF) of the leakage power consumption of c1908 for the configuration (1, 4) is shown in Figure 12. The inset plot shows the PDF in log scale. The mean values of the leakage power are almost the same for all three methods, while their variances are significantly different, with method 1 having the least variance and method 2 having the largest variance. Method 1 has reduced the variance by 31% compared to method 3. Furthermore, the PDF plots do not appear to be normal. The Jarque-Bera test for normality has rejected the hypothesis of these PDFs being normal with a very high significant level. The non-normality of the leakage power distribution is expected. Because of the correlation of channel length variation among gates falling into the same cluster, the central limit theorem does not hold anymore. These observations were true for different circuits and different configurations, as presented in Table 2.

Table 2. Mean and standard deviation of leakage power using different methods

Circuit	Mean			Standard deviation		
	Method$_1$	Method$_2$	Method$_3$	Method$_1$	Method$_2$	Method$_3$
C432	4.1e-5	4.3e-5	4.1e-5	1.8e-5	4.0e-5	2.3e-5
C499	1.4e-4	1.4e-4	1.4e-4	2.8e-5	5.4e-5	3.8e-5
C1908	9.8e-5	9.9e-5	9.8e-5	2.4e-5	5.6e-5	3.5e-5
C2670	1.8e-4	1.8e-4	1.8e-4	2.9e-5	5.7e-5	3.3e-5
C3540	1.7e-4	1.8e-4	1.7e-4	2.8e-5	6.7e-5	3.0e-5
C5315	2.8e-4	2.8e-4	2.8e-4	2.8e-5	5.5e-5	3.2e-5
C7552	3.2e-4	3.1e-4	3.2e-4	2.7e-5	5.8e-5	3.0e-5

Table 2 shows that a maximum of 31% reduction in standard deviation and an average of 17% improvement were obtained using method 1 compared to method 3. The table also shows that for all circuits, method 1 has the least standard deviation in the leakage power and method 2 has the largest standard deviation. Let us explain why we are even considering method 2 with such a large standard deviation. Figure 13 shows cumulative distribution function of leakage power using the three different placements. The figure shows that for power yield up to lower 90%, method 2 has the least leakage consumption. But for higher leakage yield, method 2 observes a huge increase in the leakage power. Method 1 results in the least leakage power variation and outperforms other methods at high values of yield.

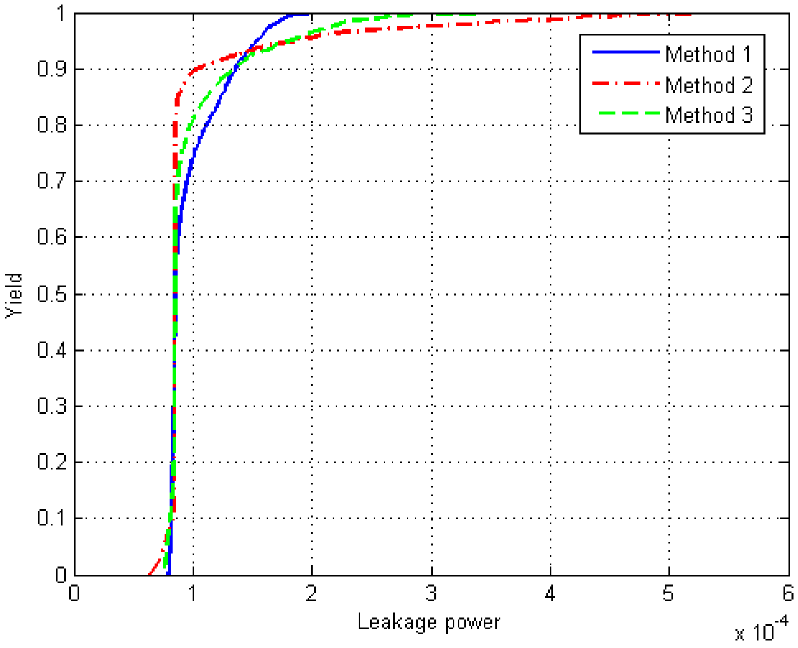


Figure 13. Cumulative distribution function

Method 1 and method 3 are compared at different percentile points in Table 3. In all but five of the 56 cases, method 1 worked better than method 3. A maximum improvement of 31% and an average improvement of 11.9% in power were observed over different configurations. Also, it can be seen that the advantage of method 1 is more at 99% yield.

To show that effect of proposed placement on the delay, we first measure the increase in wire length for the proposed methods as shown in Table 4. In Table 4, method 3 is used as the base of comparison. For each gate, the wire going out of that gate was measured by always finding the most common sub-wire which can be shared for routing different fan out gates. The total wire length is the sum of wires going out of all the gates in the circuit. It can be seen from the table that method 1 causes an average increase of

8% and a maximum increase of 12% in the wire length. For all circuits but C7552, method 2 decreases the wire length. The average decrease in wire length is 1.4%. Table 4 also shows the delay increase in circuits employing method 1 and method 2 for their placement strategies. To measure the contribution of interconnect to the delay, we use lumped wire model similar to [45]. The delay is measured by the assumption that the average wire length for the original placement (using method 3) has a capacitance effect equal to one inverter. The capacitance of other wires is scaled up up or down as a linear function of their relative length to the average wire length in method 3. The average delay increase of method 1 and method 2 are 1% and 1.6% respectively. The data shows that the negative impact of our proposed placement strategies on the delay is negligible. Assuming parasitic capacitance equal to 2 inverters, the average increase in delay is 2% and 2.6% for method 1 and method 2 respectively.

Table 3. Leakage power improvement using mehod1 at 95 and 99 percentile points

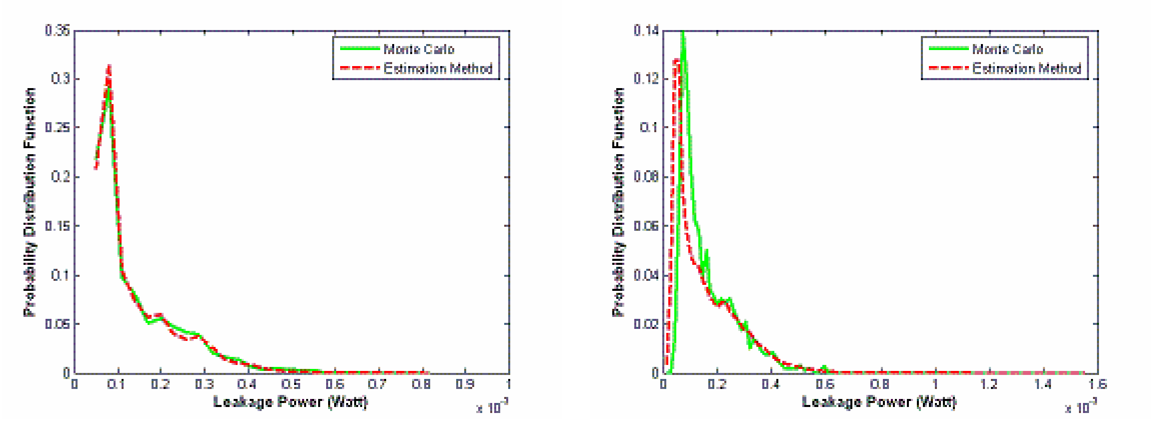
Circuit	Leakage saving (%) at 95 percentile point				Leakage saving (%) at 99 percentile point			
	Configurations (C,r)				Configurations (C,r)			
	(1,4)	(1,8)	(2,4)	(2,8)	(1,4)	(1,8)	(2,4)	(2,8)
C432	-0.08%	14.8%	12.9%	10.9%	31.4%	22.5%	29.7%	14.2%
C499	11.7%	11.7%	14.5%	12.1%	25.0%	25.2%	18.5%	11.5%
C1908	15.0%	22.2%	18.8%	19.1%	30.7%	30.1%	25.2%	26.1%
C2670	3.4%	-5.7%	10.1%	2.2%	12.1%	10.9%	15.0%	-0.34%
C3540	4.2%	11.4%	6.2%	16.4%	2.8%	20.7%	5.6%	14.1%
C5315	1.1%	-3.2%	5.3%	5.0%	8.2%	20.0%	6.4%	-11.4%
C7512	3.3%	1.4%	4.0%	8.7%	8.7%	11.5%	7.1%	7.8%

Table 4. Overhead on the wire length and delay using method 1 and method 2 (the base of comparison is method 3)

Circuit	Method$_1$		Method$_2$	
	Wire length increase (%)	Delay increase (%)	Wire length increase (%)	Delay increase (%)
C432	9.7	<1	-2.3	-1.0
C499	8.3	-2.6	-3.6	-3.2
C1908	3.4	1.3	-2.7	2.1
C2670	6.9	3.7	-1.3	2.2
C3540	6.2	<1	-1.9	4.8
C5315	12.0	1.1	-1.5	1.7
C7552	9.6	2.8	3.6	4.4

Next, the accuracy of the proposed quadratic-time-complexity technique for estimating the correlated within-die leakage power variation is evaluated by comparing it with the Monte Carlo results. Figure 14 shows the probability distribution function of the leakage power of C1908 when only the correlated channel length variation is considered. The cluster size is 4.5. The proposed estimation technique for leakage is almost identical to the Monte Carlo result in the case of two clusters. In the case of three or more clusters, the proposed method is underestimating the leakage power. The difference is because of simplifications used, and in particular because the variations of the gates falling into the intersection of two or more clusters are not considered to be the superposition of the variation of each cluster.

The error in estimating the mean and standard deviation of power are shown in Figure 15 and Figure 16 respectively. The estimation errors of these properties are quantified by comparing the results with the Monte Carlo results. The figures show that except in the case of small circuits such as c432, the estimation error is small. The larger errors for smaller circuits are due to the fewer number of samples from which the distribution of leakage power is estimated (the number of samples of the distribution being a function of function of the grid size). For an $m \times n$ grid, the number of samples is $(m \times n)^c$ for the case of c clusters. In multiple-cluster cases, the error is also due to the simplification in dealing with gates in the intersection of multiple clusters. For larger circuits, the estimation errors always stay below 10% for all statistical properties.



(a) 2 cluster (b) 3 cluster

Figure 14. Comparison of the estimation technique with Monte Carlo

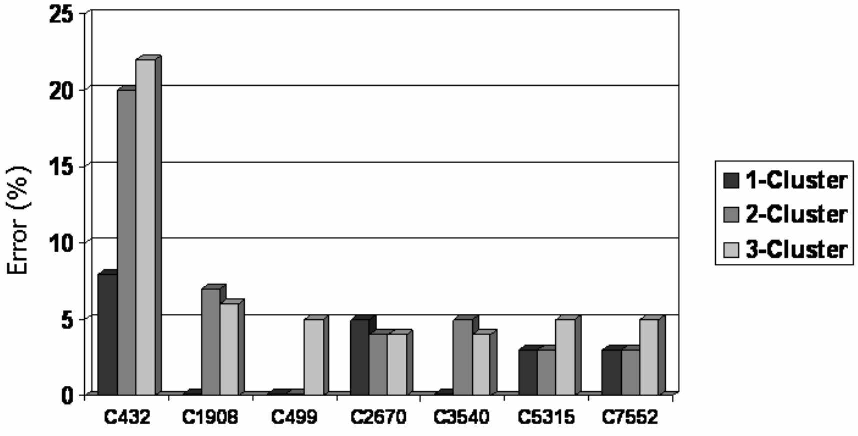


Figure 15. Error in estimating the mean

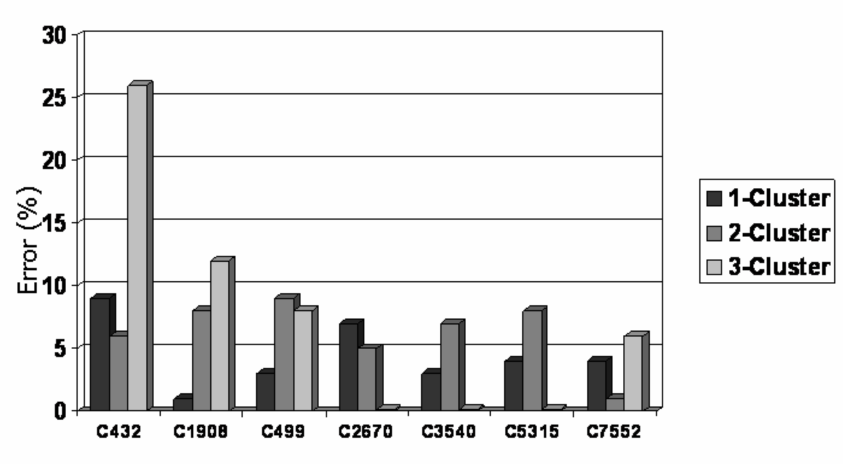


Figure 16. Error in estimating the standard deviation

The leakage estimation technique can be easily extended to the general case in which the $(i,j)^{th}$ gate within a cluster l observes the variation $f\ (d(i,j)(l),\ s_l)$ (a function of its distance from the cluster center). The required modification is in constructing the correlation kernel matrix, h. In this case, the correlation matrix contains $g(f(d(i,j)(l),\ s_l))$, where g is the function that defines the relationship between the leakage variation and the channel length variation, $\Delta p = g(\Delta(l))$. The function g is defined separately for the low-V_t and high-Vt categories through a curve fitting of the K recorded channel length variations and the corresponding leakage power variations.

3.5 CONCLUDING REMARKS

In this chapter, new approaches to the placement of standard cells in dual-V_t circuits were studied with the objective of minimizing the effect of correlated within-die variation on leakage power. A placement method, which distributes the low-V_t gates across the layout as evenly as possible results in the least variance in leakage power consumption and outperforms other methods at high yield with respect to leakage power. A method which places all low-V_t gates next to one another results in a majority of dies (about 90% of them) having low leakage power, but the leakage power of the remaining dies (about 10% of them) is very high. Leakage optimal layouts were obtained with a minimal impact on circuit delay compared to layouts that place gates on the die to minimize wire length and delay. Although we presented our work at gate level placement in the context of dual-V_t design, a similar concept can be applied at the module level with leaky modules to be placed as far from each other as possible (method1) or clustered in one location on the chip (method 2). In designs with multiple clock domains, different modules have different speed requirements and to achieve the speed limits, some fast modules might be leakier than other slower modules.

We also presented a new method for estimating distribution of leakage power variation, caused by correlated intra-die process variation. The method can accurately and efficiently estimate the distribution of leakage power. The accuracy of the proposed technique was validated on ISCAS benchmarks and by comparing it with the Monte Carlo simulation. It was shown that the method is essentially as accurate as Monte Carlo for large circuits. The new estimator can be used as a tool to guide the placement of low-V_t gates in a dual-V_t circuit to obtain the desired yield and the acceptable leakage limit.

CHAPTER 4

POST-MANUFACTURE TUNING FOR NANO-CMOS USING RECONFIGURABLE LOGIC

In this chapter, an architectural framework for post-silicon testing and tuning of circuits is developed to bring their power and performance characteristics within the acceptable ranges. Here, the objective is to have a mean of tuning circuit performance after fabrication to adjust for the likely performance variation, caused by ever-increasing process parameter variations. For circuits that meeting the timing constraints, tuning is used to reduce the power consumption. In the proposed architecture, specific hardware tuning “knobs” (control mechanisms) such as tunable gates, supply voltage, or body bias can be employed to deal with the delay or leakage variation. These control mechanisms are actuated by an approximate test that implicitly measures the delay of embedded logic circuits. A hardware framework that can support such self-adaptation is developed and algorithms are designed for optimizing the various enabling design parameters. Any available post-silicon tuning knobs can be used in conjunction with the proposed self-adaptation framework. We also propose one of such tuning knobs, called tunable gates. A tunable gate is a modified form of CMOS gate that can be programmed to work in a low-speed/low-power mode or a high-speed/high-power mode. The effectiveness of tunable gates is compared with previously introduced tunable knobs, i.e. supply voltage and body bias. Also the maximum yield improvement using all three tuning knobs combined is evaluated in the context of self-adapting framework. Simulation results show that using the proposed tunable gates on close-to-critical paths combined with V_{dd} and body bias tuning can improve the delay yield by an average of 40%. The area overheads of the proposed technique are also analyzed.

The rest of the chapter is organized as follows. The next section outlines the self-adaptation architecture. Then, the concept of tunable gates is introduced. After that the implicit delay prediction procedure is sketched. Next, experimental results are presented, followed by conclusions and future work.

4.1 SELF-ADAPTATION FRAMEWORK

In the following, we discuss the key architectural elements of the proposed self-adaptation approach. These core elements and their interactions to form a self-adaptation framework are shown in Figure 17:

<u>*Tunable gates*</u> that can be switched from a low-speed/low-power mode (OFF mode) to a high-speed/high-power mode (ON mode) using a control signal (Section 4.2.1 explains the tunable gates in details). A set of gates with a common control signal (called a *bank* of gates) can be simultaneously switched from OFF to ON. A typical reconfigurable design may contain *M* banks of gates with a total of *M* control signals, one for each bank. The tunable gates must be inserted into the “right” nodes of a design as presented in Section 4.2.2. In addition, there may be other mechanisms for modulating circuit performance and power dissipation such as reverse or forward body bias. We will analyze the yield recovery achievable using each of such mechanism individually and together in Section 4.5.

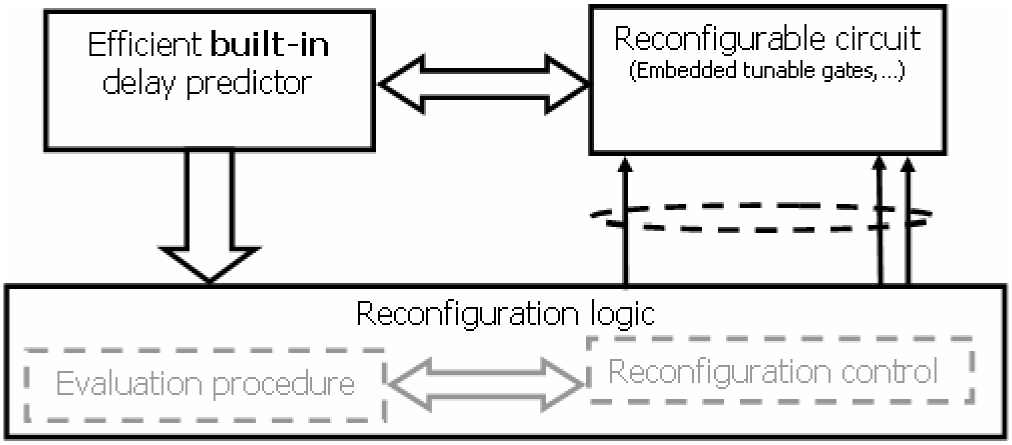


Figure 17. Self-adaptation components

<u>*Reconfiguration logic*</u> that is implemented using on-chip digital logic assigns the control signals to reconfigurable components of the circuit under test/reconfiguration in such a way that the circuit timing constraint is met with minimal impact on power dissipation. This is done by an iterative implicit delay prediction (IDP) – reconfiguration procedure. For each assignment of control signals to the tunable gates, an IDP procedure is run to predict with a *high probability of correctness* whether the circuit meets its timing constraint. The evaluation procedure of the reconfiguration logic determines how the control signals are set at each step of this iterative procedure to achieve the minimal-power solution while meeting the circuit timing constraint in a given maximum number of iterations. In fact, the reconfiguration logic is at the heart of the self-adaptation framework. It evaluates the circuit under test using the IDP and assigns the control signal to achieve a new configuration. The detail of this module is explained in Section 4.4.

<u>*An implicit delay prediction*</u> that is used to assess *with a high accuracy*, whether a circuit meets its timing constraints is implemented for on-chip prediction. Such a test is

used because it is not possible to run extensive two-pattern delay tests such as [46]-[50] at each step of the reconfiguration process for two reasons: (a) such tests are very time consuming and require large numbers of test sequences and (b) it is very difficult to guarantee full coverage in a delay-fault built-in self test (BIST) setting (we desire an on-chip delay-fault BIST capability to enable tester-independent self-adaptation). Another key factor is that the *circuit critical paths change from one step of the iterative process to another* as the tunable gates are turned ON or OFF. Hence, it is impossible for a given delay-fault test set to have a fixed "coverage" for any two random assignments of the M control signal values. In practice, the IDP consists of random two-pattern delay-fault tests of limited size such that the self-adaptation can be performed within a specified time bound. For the IDP, we assume that tests are generated by an LFSR and the results are compressed by a MISR [51]. It is assumed that each time the IDP is run, the same random test set is applied to the CUT.

4.2 TUNABLE GATES

In this section, the concept of tunable CMOS gates is introduced. The tunable gates have the capability to operate in two modes: a low- low-speed/low-power mode (called the "OFF" mode) and a high-speed/ high-power mode (called the "ON" mode). In the following, the problems of design, insertion, and control of tunable gates are discussed.

4.2.1 Tunable Gates: Design

The general architecture of a tunable gate is as follows. It uses the asymmetry in the rise time and the fall time of CMOS gates and brings down the one that is dominating the gate delay (either rise time or fall time) by placing a parallel path for charging or discharging the gate output. If the rise time is exceeding the fall time, a p network identical to the p network of the original gate is placed in parallel with the original p network, between the supply voltage and the gate output. If the fall time is larger, an n network, identical to the n network of the original gate is put in parallel with it, between the primary output and the ground. This parallel network is controlled through a footer transistor, which is off in the OFF mode and on in the ON mode. The footer is sized up to be four times the size of a regular FET so that its effect on the delay of the OFF mode case is less pronounced (Figure 18).

In the ON mode, the footer transistor is on, which makes the two (p or n) networks in parallel and therefore reduces the network resistivity and contributes to the reduction of rise or fall time. At the same time, having the parallel network increases the gate output junction capacitance and negatively impacts both the rise and the fall time. Furthermore, the input capacitance of the tunable gate is more than the input capacitance of the original gate, because now each input has to drive at least an extra transistor in the parallel network in addition to the one(s) it drives in the original gate. The increase in the input capacitance makes the driving gate slower. If the reduction in delay resulting from decrease in resistivity of charge or discharge path surpasses the increase in the delay because of the increase in the gate junction capacitance and the increase in the input capacitance, then in the ON mode the tunable gate will make a circuit operate faster than the original circuit. The achievable speed-up depends on several conditions, including the external load capacitance at the gate output and the disparity between the rise time and the fall time (which is correlated to the gate fan-in in standard cells).

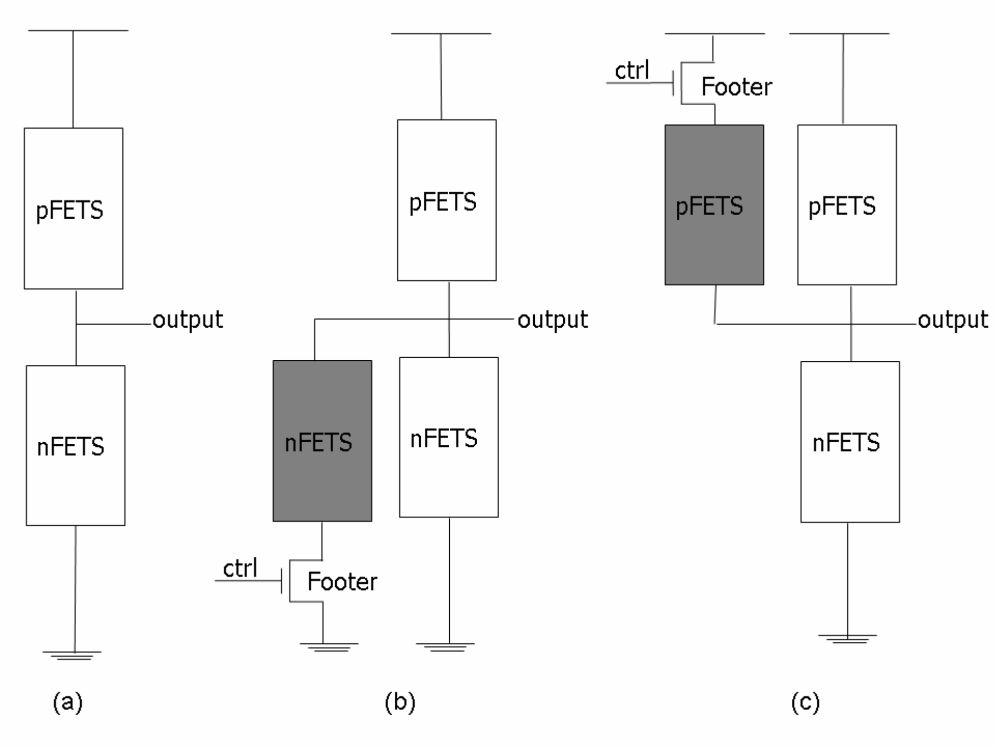


Figure 18. Structure of tunable gate a) Static CMOS , b) CMOS with reduced fall-time, c) CMOS with reduced rise-time

To analyze the effectiveness of tunable gates, regular NAND2 and NAND4 and their corresponding tunable version were implemented using SPICE 70 nm models [42]. Table 5 shows the delay characteristics of tunable gates for both NAND2 and NAND4 (with an nFET parallel branch). The delay is measured over a chain of four NAND gates, when there were all regular NAND, as the base of our comparison, and when they were converted to tunable gates with the parallel branch on/off. Positive numbers show a delay increase and negative numbers show a delay reduction. The table shows that the improvement in the fast mode is greater for NAND4. Also our data shows that larger external output capacitance makes the delay improvement more in the ON mode and results in less delay penalty in the OFF mode.

Table 5. Tunable NAND gate delay change in the ON/OFF mode

Delay changes (%)			
NAND2 → tunable NAND2 (OFF mode)	NAND2 → tunable NAND2 (ON mode)	NAND4 → tunable NAND4 (OFF mode)	NAND4 → tunable NAND4 (ON mode)
+10	-20	+2.5	-29

The extra parallel network will impose extra leakage and dynamic energy. Table 6 shows the leakage overhead of tunable NAND2 in the ON and OFF mode. The columns labeled "Output = 0" are when the output is at stable 0. The table shows that the leakage power does not increase when the output is at stable "0", and increases by 42% and 100% when the output is "1" for the case of OFF mode and ON mode respectively.

Since the dynamic power is a function of operating frequency, similar to the case of delay measurement, earlier in this section, we measure the dynamic power over a chain of four NAND2 gates and a chain of four tunable NAND2 in the ON and OFF mode. The dynamic power increases by 2% and 40% for the case of tunable gates OFF and ON respectively. It should be noted that dynamic power is a function of frequency, and that is one of the reason for the huge increase in the dynamic power when we switch from static NAND2 to tunable NAND2 in ON mode (another reason is increase in junction capacitance). The increase in dynamic energy (not a function of frequency) is only about 12% and is independent of the mode of operation of the tunable gate.

Table 6. Leakage overhead of tunable gates

Leakage increase (%)			
NAND2 →Tunable NAND2 (OFF mode)		NAND2 →Tunable NAND2 (ON mode)	
Output = 0	Output = 1	Output = 0	Output = 1
≈0	42	≈0	100

4.2.2 Tunable Gates: Insertion

To insert the tunable gates, paths whose delay is at least 90% of the longest path delay are assumed to be critical. A final output, whose arrival time is greater than 90% of the longest path delay, is considered to be a critical output. Let O_1... O_m be the critical outputs. Also, assume that there is a limited number of external control signals (*M*) to turn the tunable gates on and off. Below is a simple procedure to choose gates to become tunable and divide them into *M* banks such that they can be controlled via the *M* available control signals. There is a limit on the number of gates allowed to be tunable to keep low the area and power overhead. Let *N* be the maximum number of gates that can be made tunable. Intuitively, gates on critical paths with higher fan-in or fan-out are good candidates to become tunable.

Procedure

```
1. Mark  gates on critical paths
2. For each marked gate $g$:
        Find $S_{gj}$, the increase in the delay of critical output $O_j$ when g is tunable and goes from
    ON to OFF.
        If there is an output $O_j$ such that $S_{gj}$> threshold
          $g$  is candidate for tunability
          $S_g = \sum_{j = 1..m} S_{gj}$
```

3. Sort gates by their S_g value
4. Keep the top *N* gates with the largest S_g
5. Divide the *N* gates into *M* banks such that the sum of S_g of gates in a bank is almost equal for different banks.

In the above procedure, S_g is an indicator of achievable speedup if the gate *g* is tunable and operates in the ON mode. Tunable gates are switched from ON mode to OFF mode by the reconfigurable control module as shown in Figure 17.

4.3 IMPLICIT DELAY PREDICTION

The overall self-adaptation framework is shown in Figure 17. As described in Section 4.1, the adaptation is done by the "reconfiguration logic". It is an iterative procedure and consists of two main steps: a delay-evaluation step (to verify whether the circuit is meeting the timing constraint or not) and the assignment of control signals (reconfiguration). It is not practical to run an accurate delay test procedure with high test coverage here, because of the large test time. Also, our objective is to achieve true self-adaptation and accurate delay testing requires some level of external tester support. To tackle this issue, we propose the IDP. In practice, the IDP is a two-phase procedure as described below.

The first phase is a calibration phase, a one-time procedure during which we pick *N* sample ICs (called the calibration set) and find the actual delay of each sample (*T*) and the maximum measurable delay (*T'*) of the sample in the calibration set using a small set of test vectors, *S* (practical for self-adaptation). We choose a threshold on *T'*, called T_h, as a decision making criteria such that we can achieve a good classification of the calibration set if we categorize all dies with $T' < T_h$ as good and all dies with $T' > T_h$ as bad.

After such a T_h is found, during the second phase, where the actual test (prediction) is performed on the circuit under test (CUT), the test set *S* (the same used during the calibration phase) is applied to the circuit at the clock rate T_h and the responses are recorded in a multiple inputs shift register (MISR) and are compared with the output of a good circuit. The comparison results in a go/no-go answer. During this phase, we are predicting if the circuit is meeting the speed requirement by testing if the paths that are exercised by the test set S have delays less or equal to T_h. The test set *S* can be generated using an LFSR seeded with the *same seed* for the pre-processing calibration phase and the IDP- reconfiguration phase.

During the calibration procedure, *T'* can be found using a procedure similar to the F_{max} procedure as described below. We run *S* on a sample at a given clock period and if the response is correct, the clock period is reduced. *S* is run again on the sample circuit at the reduced clock period and the procedure continues until the circuit fails (i.e. the output response is incorrect). The smallest clock period for which the response to test set S is correct defines *T'*. Likewise, the actual circuit delay, *T,* can be found using the same procedure and an exhaustive delay test set. Note that $T' < T$ due because the test set *S* does not necessarily exercises all the circuit critical paths.

Figure 19 shows the probability distribution of maximum measured delays for circuit C1908 of ISCAS suits using a randomly generated test *S* with thousand vectors. Samples used in Figure 19 suffer equally from correlated and random variations

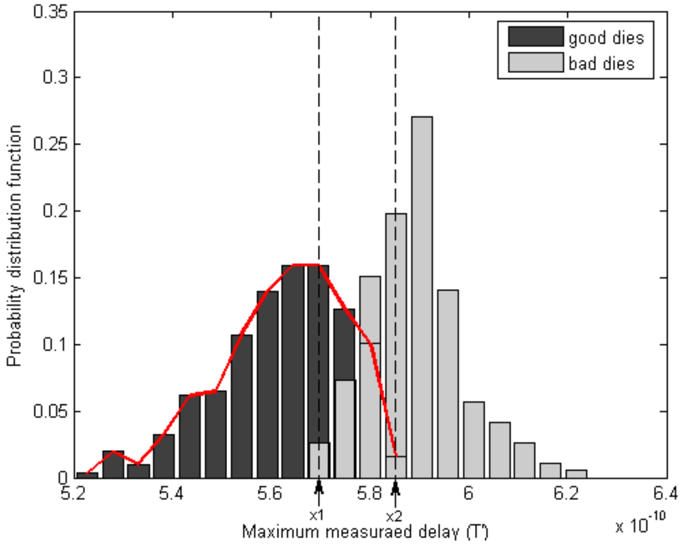


Figure 19: Distribution of maximum measured delays of good dies and bad dies

In Figure 19, the darker histogram corresponds to ICs that meet the original timing specifications (under an exhaustive test) and the lighter histogram corresponds to ICs that do not. The histograms may overlap. The problem is to find the value of T_h (to be used during the IDP procedure) such that as many good dies are classified as good and as many bad dies are classified as bad and misclassification is minimized. Next, we explain the issues related to finding T_h. Then, we show how the pattern of process variation affects the correlation between the actual delay of the circuit and the maximum measured delay.

Let x_1 be the minimum *T'* of bad dies and x_2 be the maximum T' of the good die as marked in Figure 19. If $x_2<x_1$, the two histograms do not overlap. In this case, we set T_h to be x_1. When $x_2 > x_1$, then there is an overlapping region. In this case, the choice of T_h is a trade off between the test escape probability (the probability of a bad die passing as good) and the amount of unnecessary reconfiguration performed (the probability of a good die passing as bad). For example, consider an extreme case when T_h is chosen to be x_1. This forces good dies with maximum measured delay in the range of [x_1…x_2] to be reconfigured (unnecessary reconfiguration). Another extreme case is to pick T_h to be x_2. This will make bad dies whose maximum measured delay falls in the range of [x_1…x_2] to escape the test and be considered as good die. T_h can be chosen to be x_1 or x_2 or anything in between depending on the designer's primary objective. Below, it is shown how the choice of T_h can affect yield improvement or result in over compensation. Four different T_h values are chosen and their corresponding yields are compared with the case of perfect delay test (Figure 20) for circuit C1908 with embedded tunable gates that can be turned ON by the reconfiguration logic. Figure 20 (a) and (d) uses x_1 and x_2 as T_h respectively. Figure 20 (b) and (c) use values between x_1 and x_2 as T_h. Figure 20 shows that the yield loss increases with an increase in T_h. It should be mentioned that unnecessary reconfigurations (i.e. when the circuit meets the timing requirement but still being reconfigured because of the imperfection of IDP) are 37%, 33%, 27%, and 5% for subplots (a), (b), (c), and (d) respectively.

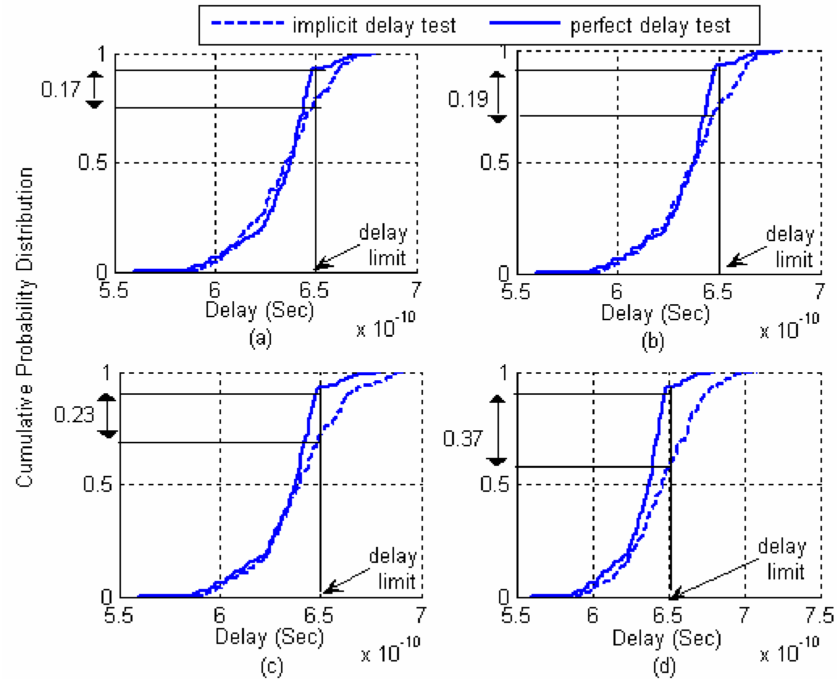


Figure 20: Yield loses as a result of applying IDP instead of a perfect delay test for different values of T_h increasing from (a) to (d)

Figure 21 shows the actual circuit delays as a function of maximum measured delay using a small test set *S* for three different process variation patterns, namely only die-to-die variation, only random gate-to-gate variation, and both random gate-to-gate variation and correlated cross die variation, respectively. It can be seen that when there is only die-to-die variation, one could predict the actual delay of the circuit based on the measured delay with 100% accuracy. There is no real correlation between the actual delay of the circuit and the maximum measured delay when there is only random gate-to-gate variation. However, when there are both random gate-to-gate and the correlated intra- and inter-die process variations, one could predict with a high degree of confidence the actual delay of the circuit from the measured delay. The focus of this chapter is on the last case where there are both correlated and random within-die process variations.

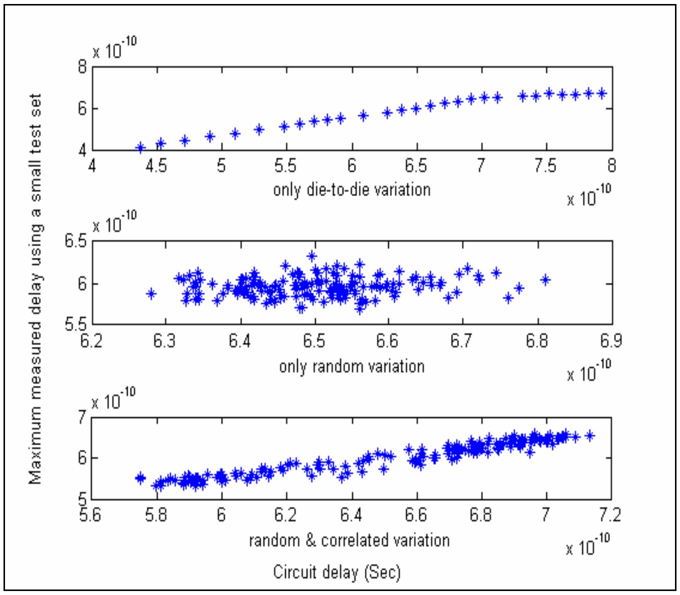


Figure 21: Circuit delay as a function maximum measured delay

4.4 TUNING STRATEGY

The proposed tunable gates are useful in reducing the delay and therefore improving the timing yield. In practice, there are cases, where due to process variations, the power becomes prohibitive and the power limit is violated, resulting in yield loss. In this chapter, we use tunable gates along with two previously-known tuning knobs, i.e. the supply voltage and body bias to improve the circuit delay and power.

Here, we first outline our general strategy for adaptation to process variation using tunable gates, power supply, and body bias. Then, we provide the concrete procedure for doing the adaptation. The tuning procedure can be simply implemented in hardware using finite state machines. In Section 4.5, we provide an overhead of implementing such a circuitry on-chip.

The general guideline for our adaptation procedure is as follows:

- We only reduce supply voltage from its nominal value (no supply voltage increase) in cases when the timing is met to reduce power consumption. In another word, we do not increase the supply voltage as a tuning knob to bring the timing in the acceptable range. The reason is that to date circuits generally operate in the maximum supply voltage allowable by reliability measure. Therefore, it is impractical for reliability reason to increase the supply voltage.
- We do not use reverse body bias to reduce the leakage consumption because its ineffectiveness diminishes in smaller technologies as studied in [10].
- We use forward body bias and turn ON tunable gates to bring the timing in the acceptable level.
- We do not use tunable gates as a way of bringing the supply voltage below the nominal values, i.e. the supply voltage only will be brought below the nominal values if the manufactured circuit, with no tunable gate ON, meets the timing requirement.

In the experimental result section, when doing the comparison between tunable gates, AVS, and ABB, we let the supply voltage go above the nominal operating supply voltage and we use reverse body bias.

The procedure used by the *reconfiguration control* module is shown in Figure 22. Just for the ease of explanation, assume that the first k_1 bits of the (reconfiguration) control signals represent the value of supply voltage, the next k_2 bits represents the bias values, and the last k_3 bits represent the control values of tunable banks. Let us call the first k_1 bits, *group*$_1$ control signals, the next k_2 bits, *group*$_2$ control signals, and the next k_3 bits *group*$_3$ control signals.

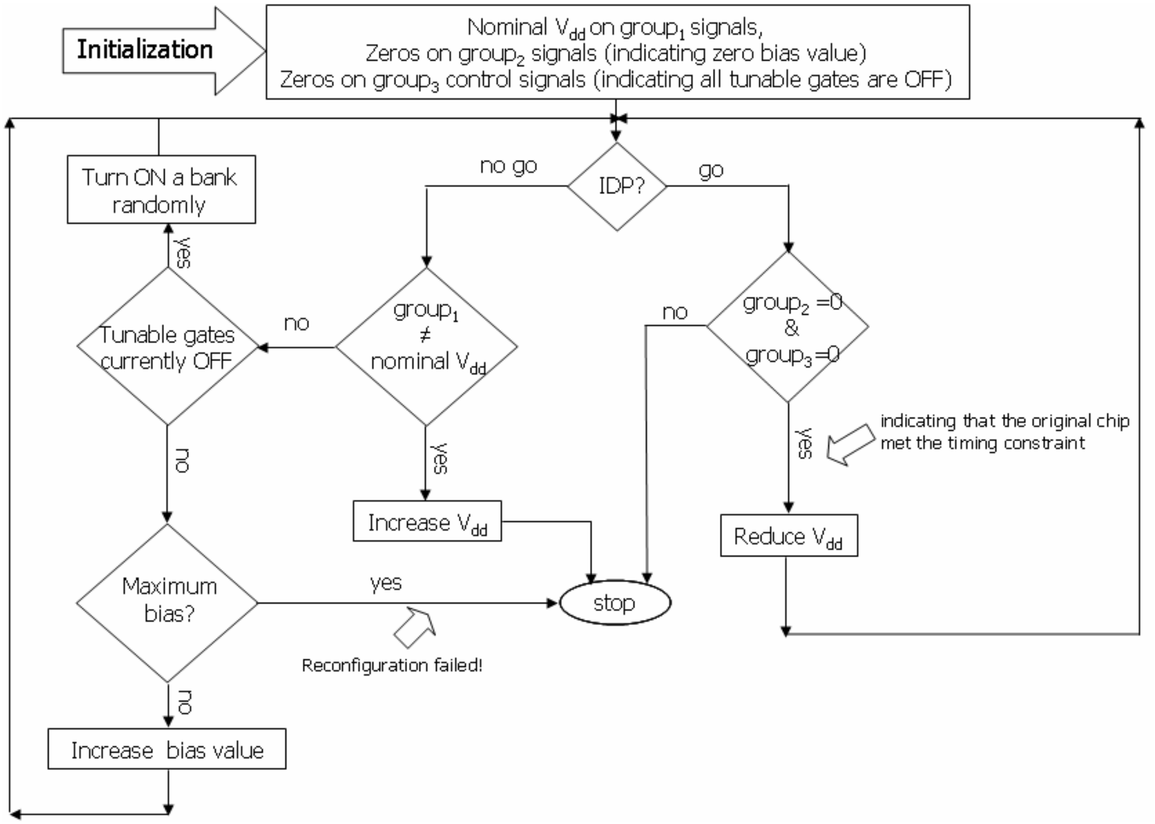

Initialization
Nominal V_{dd} on $group_1$ signals,
Zeros on $group_2$ signals (indicating zero bias value)
Zeros on $group_3$ control signals (indicating all tunable gates are OFF)
Turn ON a bank randomly
no go
IDP?
go
yes
Tunable gates currently OFF
no
$group_1$ ≠ nominal V_{dd}
no
$group_2$ =0 & $group_3$=0
indicating that the original chip met the timing constraint
no
yes
yes
Increase V_{dd}
Reduce V_{dd}
Maximum bias?
yes
stop
Reconfiguration failed!
no
Increase bias value

Figure 22. Reconfiguration procedure

In the experimental result section, we also study the effectiveness of AVS, ABB, and tunable gates individually. For the case of V_{dd} as the tuning knob, the technique in [13] is used. For the case of body bias and tunable gates the techniques in [9] and [52] are used respectively. The technique proposed in [52] is sketched below and depicted in Figure 23. In step 1 (box labeled 1), all the tunable gates are set initially to the low-speed/low-power mode. In step 2, the IDP is run and a decision is made on whether the circuit meets its timing specification or not. If it meets the timing, then the current configuration of control signals is "accepted" (step 4). This configuration is held for future consideration. If not, then the circuit needs to be "speeded-up" by *randomly* turning "ON" a bank of tunable gates that is currently in the "OFF" mode. The IDP is repeated, iteratively turning "ON" tunable gates until the circuit timing constraint is met. It is possible that such a process might not converge (e.g. if the circuit is excessively slow). In such a case, a viable solution is not possible (this is not shown in the simplified graph of Figure 23). The combined loop of steps 2, 3 and 4 is called a *reconfiguration sequence* and results in a sequence of specific banks of tunable gates being turned "ON" until the timing specifications are met. The objective of steps 5, 6 and 7 is to see if there are other reconfiguration sequences that meet the timing constraints with *fewer banks of tunable gates turned "ON"* from the initial starting configuration (step 1). If such sequences exist, then the circuit timing constraints can be met with *lower power dissipation cost*. Hence, in step 5, a "reconfiguration sequence of lower weight" is defined to be one that results in fewer banks of tunable gates turned "ON". This is easily determined by *counting the number of 1's in the M control signals*. Step 6 keeps track of the "best" solution obtained at any step of the procedure and step 7 determines if the maximum number of allowed iterations (this determines the time needed to do adaptation) has been exceeded. In step 3, the next bank of gates to be turned "ON" is picked *randomly* to make any pair of *reconfiguration sequences be as different from each other as possible*. In this way, in the limited number of adaptation iterations, the solution space is sampled uniformly and possible process variations in different layout regions of the CUT are covered evenly.

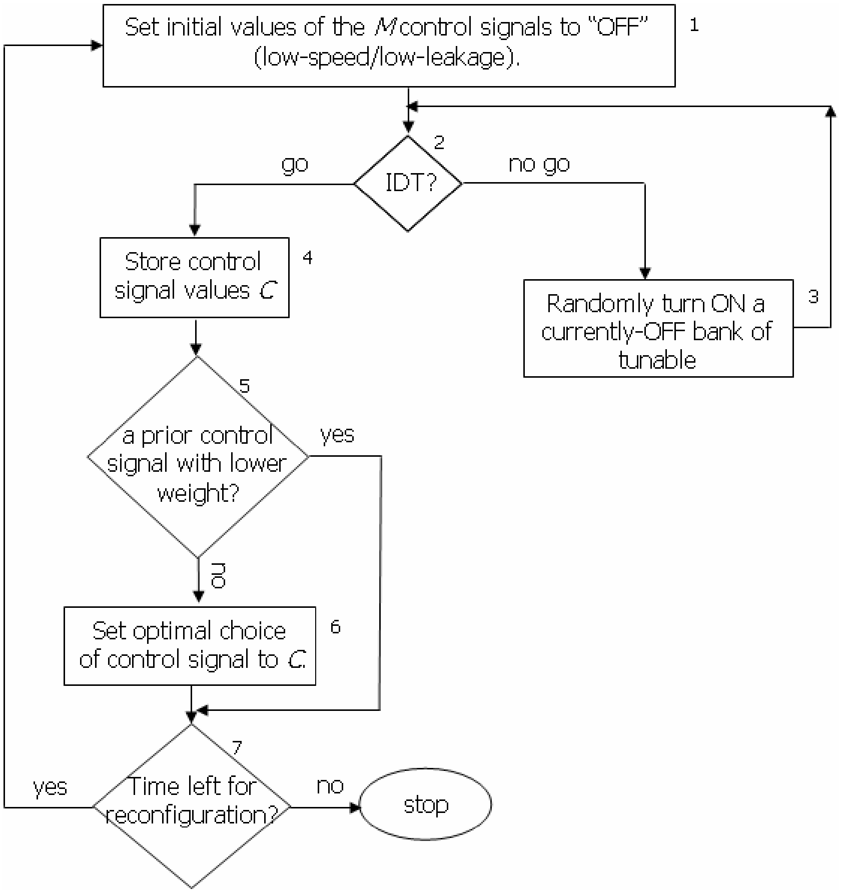

Set initial values of the M control signals to "OFF" (low-speed/low-leakage).
1
2
IDT?
go
no go
Store control signal values C
4
Randomly turn ON a currently-OFF bank of tunable
3
5
a prior control signal with lower weight?
yes
no
Set optimal choice of control signal to C.
6
7
Time left for reconfiguration?
yes
no
stop

Figure 23. Procedure for assigning control signals of tunable gates

4.5 EVALUATION

To evaluate the yield improvement achievable by the proposed reconfiguration architecture, circuits from the ISCAS'85 benchmark circuits was synthesized using Synopsys Design Compiler with a library of 2-input to 4-input NAND and NOR gates and inverters. The synthesized circuits were used with SPICE 70 nm models [42] to compute the propagation delay, dynamic, and leakage power using a look-up table. All gates had a transistor channel length of 70 nm, VDD of 1V, Vt of 0.2 V. The transistor channel length is assumed to vary within 15% of its expected value with equal contribution from random and correlated variations. Because of the delay and area overhead of tuneable gates, the percentage of gates made tunable is kept between 5-10% of gates in a circuit. Candidate gates for tunability are the ones on paths with delays about 90%-100% of the longest path delay ($T_{critical}$) chosen by the heuristic presented in Section 4.2.2. The circuit were placed using a simulated annealing procedure with the objective of minimizing the total wire length. To model random correlated within-die process variation, the grid model similar to [32] is used. A 3×3 grid for modeling within-die variation is assumed. While using supply voltage as a tuning knobs, the value of supply voltage changes within ±20% of its nominal value with step size of 50mv similar to assumption used in [13]. When using the bias value as a tuning knob, we use bias values in the range of ±20% of V_{dd} [13]. In this research, only tunable NAND gates are considered because of their high usage in CMOS circuits and the larger area penalty associated with a parallel p network required in NOR gates.

In the first experiment of this section, we evaluate the effectiveness of each of the three available tuning knobs individually, both in terms of recovering timing yield and their impacts on power (leakage and dynamic). Figure 24 shows the delay distribution using “no tuning”, ABB, tunable gates, and AVS. In terms of timing yield recovery, ABB, tunable gates, and AVS increase the time yield by 13%, 36%, and 40%, respectively. Our study shows that if we allow the bias value to be within ±50% of V_{dd}, ABB will be almost as effective as tuneable gates and indeed increases the timing yield by 33%. AVS is the most effective of the techniques for timing yield recovery. The impact of the technique on power is summarized in Table 7. In terms of impact on leakage power, AVS and tuneable gates increase the mean leakage the same way while

ABB does not increase the mean leakage. If tunable gates were always on, the leakage power would increase by another 2-3% for this circuit. This number will increase as number of tunable gate increases, which justifies why tunable gates should be turn ON only if they are needed. In terms of impact on dynamic power, as expected AVS has the largest impact. Using AVS, the average dynamic energy increases by 6.5% respectively.

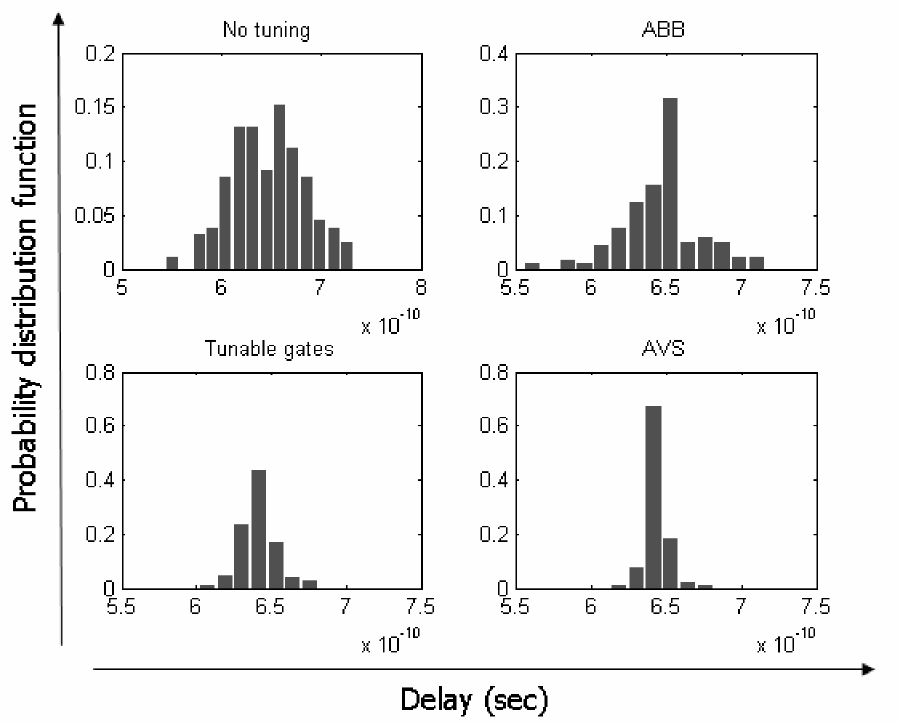


Figure 24. Delay distributions using different tuning knobs individually

Table 7. Impact of tuning techniques on power

Tuning technique	Average leakage power increase (%)	Average dynamic power increase (%)
ABB	-2.7	-0.3
Tunable gates	3.2	3.9
AVS	3.5	6.5

Below, we analyze the effectiveness of the technique proposed in Section 4.4, where supply voltage, body bias, and tunable gates are all used as post-manufacture tuning knobs. In Figure 25, we show the delay distribution for four ISCAS circuits for the case of no tuning and when the tuning procedure in Section 4.4 is applied. The leakage and dynamic power plot are shown in Figure 26 and Figure 27 respectively. Similar results were obtained for other ISCAS circuits but we don't show them here to keep the plots readable. Our experimental data on ISCAS benchmark circuits show that using the tuning procedure the standard deviation of delay improves by an average of 80%. The mean of delay does not change dramatically. Tuning provides 1.3% mean delay reduction. The mean and standard deviation of leakage improves by 7.1% and 51% respectively. The mean of dynamic power improves by 4.8% and the standard deviation increases by an average of 71%. In these figure a perfect delay test is assumed. Table 8 shows the delay yield improvement when the proposed tuning procedure is used along with a perfect delay technique and along with IDP procedure. The average delay yield improvement goes down from 45% to 42% when the perfect delay test is replaced by the IDP procedure. The table also shows the power improvement achievable through tuning. The power improvement was computed using the maximum power consumption of the dies that meet the timing requirement before and after tuning. An average leakage power improvement of 33% and 24% was obtained when the reconfiguration module uses a perfect delay test and the IDP procedure respectively. The dynamic power improvement of 12% and 6% was obtained when tuned using the perfect delay test and the IDP respectively. These results show that IDP could be used instead of an expensive delay test method while tuning with little impact on yield improvement or power consumption.

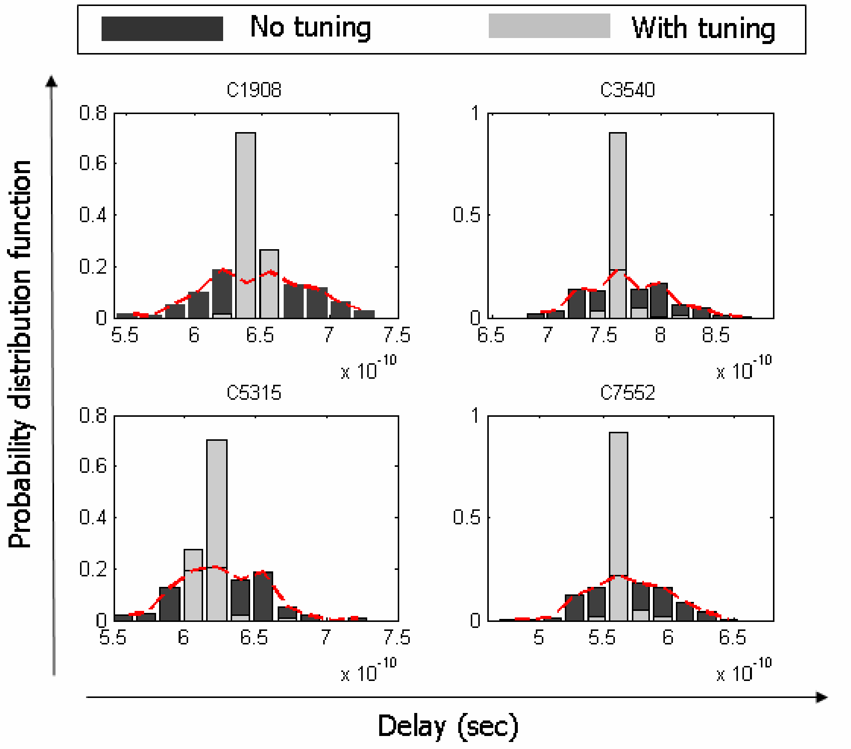


Figure 25. Delay distribution using the proposed tuning procedure

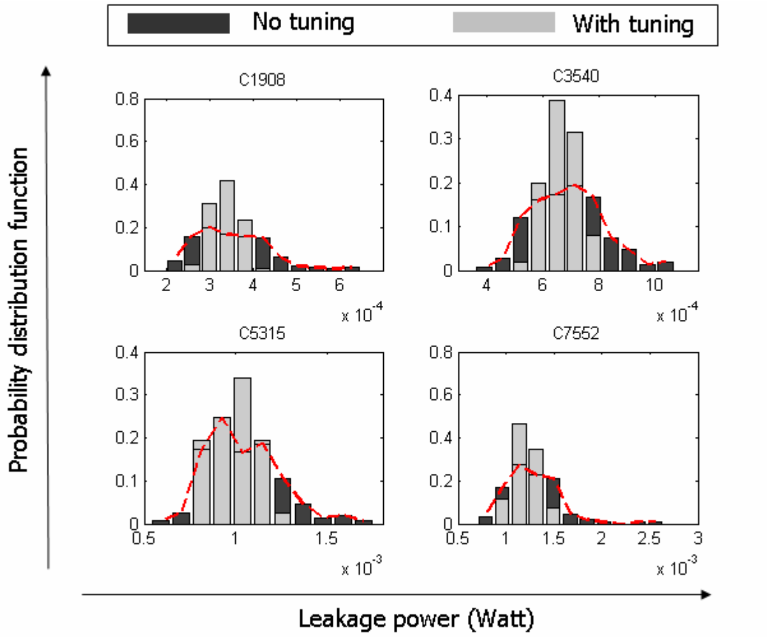


Figure 26. Leakage distribution using the proposed tuning procedure

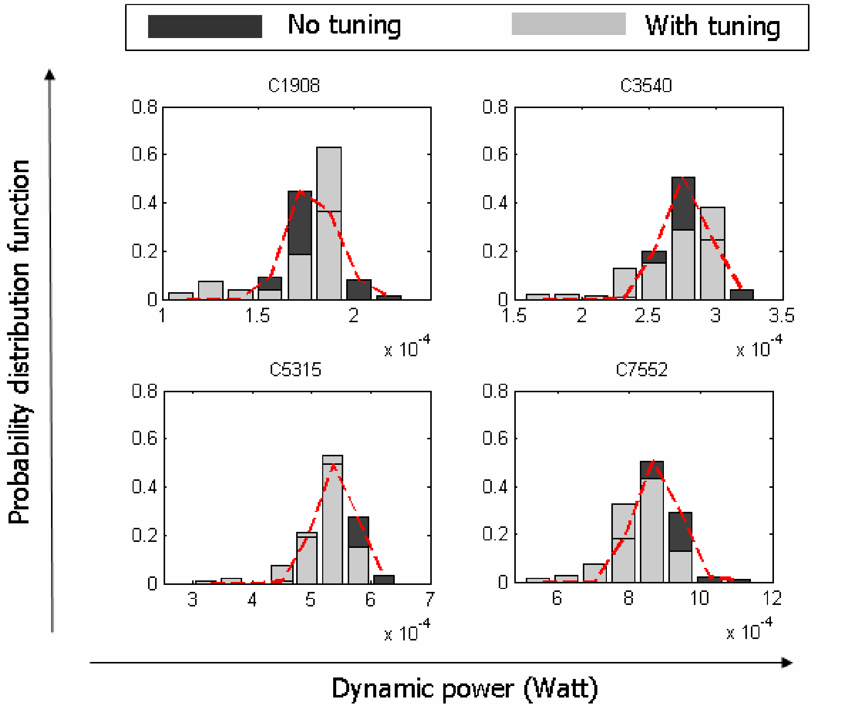


Figure 27. Dynamic power distribution using the proposed technique

Table 8. Delay and power improvement using tuning methodology with perfect delay test and with IDP

Circuit	Tuned using perfect delay test			Tuned using IDP		
	Delay yield improvement (%)	Leakage improvement (%)	Dynamic improvement (%)	Delay yield improvement (%)	Leakage improvement (%)	Dynamic improvement (%)
C432	49	42	13	45	38	9
C499	49	33	10	41	25	5
C1908	43	33	14	38	22	9
C2670	56	32	12	54	27	7
C3540	35	22	9	35	8	3
C5312	45	29	8	41	18	2
C7552	41	38	15	41	31	9

To evaluate the area overhead of the proposed tuning methodology, the reconfiguration module (including the IDP procedure) was implemented in Verilog and synthesized using *Synopsys Design Compiler* and 0.25 μm standard cell library. The ISCAS benchmark circuits were also synthesized using *Design Compiler*. The area overhead of the reconfiguration module is shown in Table 9. For small circuit such as C432 the area overhead is considerable. But for larger circuit the area overhead is small and is negligible for the largest circuit, C7552. The area overhead of tunable gates is also shown in Table 9. This overhead includes the overhead of the large footer transistor of tunable gates for tuning them ON or OFF. The area overhead of tunable gates depends on the percentage of gates made tunable. This information is also presented in Table 9. For implementing the routing of control signals of footer transistors in tunable gates, an attractive option is the poly-silicon layer in the layout. Note that there are no switching performance constraints on the control signals. These are not signals that switch during normal operation. Thus the reconfiguration control grid can be laid out in the poly-silicon layer, which is too slow to be used for the functional interconnect. *Such a strategy allows* *the reconfiguration control signals to be implemented with virtually no adverse impact on the availability of the metal layers for use for the functional interconnect.*

Table 9. Area overhead of reconfiguration module and tunable gates

Circuit	Area overhead of reconfiguration module (%)	Percentage of tunable gates	Area overhead of tunable gates (%)	Total area overhead (%)
C432	33.5	5.5	8.4	41.9
C499	10.4	10	15.1	25.5
C1908	13.1	10	17.5	30.6
C2670	9.9	4.6	6.5	16.4
C3540	6.9	5	7.2	14.1
C7552	0.3	5	7.4	7.7

4.6 CONCLUDING REMARKS

In this chapter, an architectural framework for post-silicon performance testing and tuning to bring the delay and power consumption of a die within the acceptable range was developed. Also, a modified form of CMOS gate that can be programmed to work in a low-speed or a high-speed mode is presented.

In the proposed architecture, specific hardware tuning “knobs” (control mechanisms) such as tunable gates, supply voltage, or body bias can be employed to deal with the delay and power variation. These control mechanisms are actuated by a proposed efficient delay test method that *implicitly* measures the delay of embedded logic circuits. A hardware framework that can support such self-test/self-adaptation is developed and algorithms are designed for optimizing the various enabling design parameters. This work covers different area from delay testing to low-level CMOS gate design of tunable gates. Simulation results show that using the proposed tunable gates on close-to-critical paths combined V_{dd} and body bias tuning can improve the delay yield by 42%.

CHAPTER 5

TRANSIENT ERRORS: TRENDS AND SOLUTIONS

Due to technology scaling and the increased susceptibility of deep sub-micron (DSM) circuitry to uncertainties originating from noise (reduced noise margins) and soft errors (induced by atmospheric neutrons), it will become necessary to design error detection and correction capability into future logic designs for reliable computation. In the past, data encoding techniques have mostly been used for error detection and correction in wired and wireless communications channels. In order to enable reliable computing in nanoscale technologies of the future, not only data but also computation performed on the data will need to be encoded for real-time error detection and correction (i.e. redundant computations will need to be performed).

The technology scaling increases the vulnerability of a circuit to transient errors for several reasons. First, because of the feature size reduction, which reduces the average node capacitance, the voltage fluctuation at a node is larger. Second, the supply voltage reduction in every technology generation reduces the noise margins and aggravates the transient error problem. Third, the increase in the clock frequency raises the chance of an error being latched and propagated to a primary output. Moreover, because of shorter pipeline stages, the number of gates through which an error propagates (and hence attenuates) is smaller. Therefore, the probability of an error being masked in a modern high-performance digital system is becoming increasingly small compared to earlier technologies ([53]and [54]).

Early on in the silicon era, it became known that the dense memory elements such as DRAM or SRAM are susceptible to errors [55]. That is why coding techniques such as parity bits are used to design reliable memory banks and to enable reliable memory access, but their use in on-chip signal processing has been limited. In the scaled digital circuits, the transient errors in flip flops demand an immediate attention. Furthermore, it is expected that the error rate of combinational logics in the scaled technologies to escalate by 9 orders of magnitude from 1992 to 2011, when it will equal the error rate of unprotected memory elements [53]. Hence, further down the technology road map handling combinational logic errors is also essential for the future logic design [55].

One of the key barriers to widespread use of coding techniques for reliable on-chip computing is the cost of data and circuit redundancy necessary to implement a coding technique with logic error detection and correction. Theoretically, a code of distance $t+1$ is necessary for detecting up to t errors and a code of distance $2t+1$ is necessary for correcting up to t errors. Various linear codes such as those based on real-number checksums ([55]-[62]) have been used in the past for real-time error detection in applications such as digital filtering, matrix multiplication, convolution, and FFT computation. While error detection is accomplished relatively easily across the majority of known algorithm-based ([56]-[64]) and communication system coding techniques ([65]-[67]), error correction is a harder problem and can require significant computation for exact error correction. This renders real-time correction without loss of significant throughput difficult, if not impossible, to achieve and is especially true for the majority of

DSP applications that involve matrix-vector multiplications and are the core subject of the error detection and compensation technique presented in Chapter 6. In this context, it is important to point out that in future scaled technologies with high error rates, rapid error correction with least impact on throughput will be a critical technology enabling factor. Without this capability, technology scaling itself may grind to a halt due to gross loss of circuit and system level performance.

The next section reviews previous work on fault/error-tolerant techniques.

5.1 FAULT-TOLERANT TECHNIQUES

Traditionally, fault tolerant techniques were used in applications where an error could result in an irreversible consequence. Different forms of redundancy such as the hardware, software, time, and/or information redundancy, are the main components of a fault tolerant system [68]. The hardware redundancy techniques, utilized in the triple module redundancy (TMR), have a high hardware overhead but less effect on system performance. The high area and power cost associated with these techniques makes them impractical for more general applications. At the same time such a high price is not necessary in most cases especially in non real-time systems. Therefore techniques such as the time redundancy or partial duplication or software redundancy have been proposed. Such techniques have far less hardware overhead but severely impact the system performance. The algorithm-based fault-tolerant methods with focus on matrix operations and FFT have been proposed in [55] - [60], [63], and [64]. The algorithm-based fault-tolerant methods aim at minimizing both the hardware and the performance penalty. In all techniques mentioned so far, the focus was on eliminating the chance of an error being observed at the final output of the system, either by masking the error or by detecting it and preventing it from being observed at the final output.

In recent years, with the increase in soft-error rate in the scaled technologies, a great deal of research is dedicated to protecting against soft errors. This is done by using hardware-level solutions with the aim of reducing the probability of a transient (soft) error being observed at the circuit output with minimal impact on the circuit delay, power and area. Soft-error resilient techniques could be divided into the ones for flip flops and the ones for combinational logics. Techniques in [43] and [69] are two examples of recent techniques proposed for making the combinational logics soft-error resilient. In [43], a capacitive loading is added to the primary outputs of a circuit. Then, the size, supply voltage, and threshold voltage of internal gates are optimized to minimize the energy and delay overhead of the added capacitive loads. The technique in [69] dynamically controls the soft-error tolerance. When the particle flux in the environment is increased, the technique increases the soft-error tolerance using an adaptive supply voltage and threshold voltage modulation and variable capacitance bank. Techniques in [70], [71], and [72] protect system flip flops. In [70], the author proposes to replace each latch with two or three latches that are clocked with a fixed phase-delay or their inputs arrive after a fixed phase-delay. A simple voting circuitry is needed to pick the correct latch value. In [71], the redundancy that exists in the scan flip flops is used along with the Muller C-element to protect the flip flops against transient errors. The Muller C-element is a two-input one and one-output component that keeps its output value if its two inputs do not match. In [72], the authors propose a soft-error immune latch. The new latch design keeps its state on three different nodes. When the value is destroyed in one of these nodes, the other two nodes still hold the right value. This design is not protected against the transient errors generated in the combinational blocks.

Algorithmic noise tolerant (ANT) techniques were proposed in [73]-[76] to compensate for the error (noise) introduced into a digital signal processing (DSP) system. The errors are assumed to occur because of the supply voltage over-scaling or soft errors. They propose a few techniques for making the error less dominant in the output of the DSP using less hardware than what is required in more classical fault-tolerant techniques. The techniques have a common denominator. They all use a less complex module compared to the DSP block to find an estimate of the DSP output. The techniques are as follows:

- Prediction-based ANT [73] : Here, the assumption is that if the DSP filter is sufficiently narrowband, then the filter outputs at consecutive time points are highly correlated. This technique uses a linear predictor to predict the filter output using previous values of the output.
- Reduced precision redundancy-based ANT [74]: This technique exploits the fact that the most significant bits of the output are more critical and must be protected from the noise. The technique uses a low precision replica of the original DSP filter. If an error occurs in the original high precision filter, the output of low precision module will be directed to the output.
- Error cancellation-based ANT [75]: This technique uses the fact that in a filter with the voltage over-scaling, there is a high correlation between the input of the filter and the error at the output. In this technique, an adaptive filter is first trained to predict the error in the filter output. After the training phase, the adaptive filter

estimates the error in the filter output and adjusts the output accordingly. This technique cannot be used for random transient errors such as soft errors.

In Chapter 6, we present an error detection and compensation technique, which can handle errors in both the combinational logic and the storage elements of a digital circuit for a class of linear filtering DSP applications where it is not necessary to maintain cycle-to-cycle accurate computation as long as system level signal-to-noise ratio (SNR) metrics are satisfied or degraded within acceptable levels. Such an SNR metric may impact end-product video or audio quality. The focus is on real-time error compensation in on-chip linear DSP computations, formulated as linear state variable systems. Such computations can be implemented as ASICs using behavioral synthesis tools (also called silicon compilers ([77]-[79]) or via software running on programmable DSPs. In the former, the DSP algorithm is implemented via dedicated circuitry, whereas in the latter, the same is implemented using software.

5.2 CONCLUDING REMARKS

This chapter provides an overview of techniques proposed in the literature both for fault-tolerance in mission-critical applications and recent techniques to handle high rate of soft/transient errors.

CHAPTER 6

EFFICIENT PROBABILISTIC ERROR CORRECTION

In this chapter, a probabilistic compensation technique for minimizing the effect of transient errors is proposed. Here, the focus is to develop a transient error (soft error) compensation technique for digital signal processing (DSP) applications in which exact error compensation is not always necessary and end-to-end system level performance is degraded minimally as long as the impact of the "noise" injected into the system by the onset of transient errors is minimized. The proposed technique, referred to as *checksum-based probabilistic compensation*, uses real number checksum codes for error detection and partial compensation. Traditional coding techniques need a code of distance three and relatively complex back-end calculations for perfect error correction. In this chapter, it is shown that a distance-two code can be used to perform probabilistic error compensation in linear systems with the objective of improving the signal-to-noise ratio (SNR) in the presence of random transient errors. The goal is to have a technique with small power and area overheads and to be able to perform the compensation in real time with negligible latency of compensation. The proposed technique is comprehensive in the sense that it can handle errors in the combinational circuits as well as the storage elements. Comparison against a system with no error correction shows that up to 13 dB SNR improvement is possible. The area, power, and timing overheads of the proposed checksum-based probabilistic compensation technique are analyzed.

6.1 REAL NUMBER CHECKSUM CODES FOR ERROR DETECTION AND CORRECTION

In this work, we focus on transient upset induced error (noise) reduction in linear digital filters using a checksum-based probabilistic compensation mechanism. A digital filter, either FIR or IIR, can be implemented using three kinds of modules: adders, multipliers and registers. A filter represented by its transfer function can have many different physical realizations. It can be shown that there are infinite realizations of a given transfer function. Irrespective of the realization of a filter, linear digital filters can be represented in state variable form [80]. In the following, we first discuss this representation and then show how checksum codes can be designed, based on this representation, to detect and compensate for errors in real-time.

6.1.1 Linear Digital State Variable Systems

Linear time-invariant systems can be represented as *state variable* systems. The general form of a state variable system contains a computational block, which takes the primary inputs, ($u_1...u_m$), and the current latched states as input and generates the next states as well as the primary outputs, ($y_1…y_w$) (Figure 28). The computational block is a network of adders and multipliers and feeds the primary outputs and system latches. The processing is purely arithmetic and therefore inputs, outputs, and states represent numerical values. If $s(t) = [s_1(t), s_2(t), …, s_n(t)]^T$ is the state vector and $u(t) = [u_1(t), u_2(t), …, u_n(t)]^T$ is the input vector at time t, then the system can be represented by the following equations:

$$\begin{aligned} s(t+1) &= A \cdot s(t) + B \cdot u(t+1) \\ y(t+1) &= C \cdot s(t) + D \cdot u(t+1) \end{aligned} \qquad (7)$$

where the *A, B, C,* and *D* matrices represent the arithmetic operations performed on the *n* states, *m* primary inputs, and *w* primary outputs in the computational block. In general, a computational module (i.e. an operator) in the computational block can feed more than one state or one output of the system. Such an implementation of the system where the computation trees of different states or outputs are not disjoint is referred to as a *shared implementation* [61]. In a shared implementation, an error in an operator may result in multiple erroneous states. We discuss the challenges posed by the shared implementation of the computational block later in the chapter.

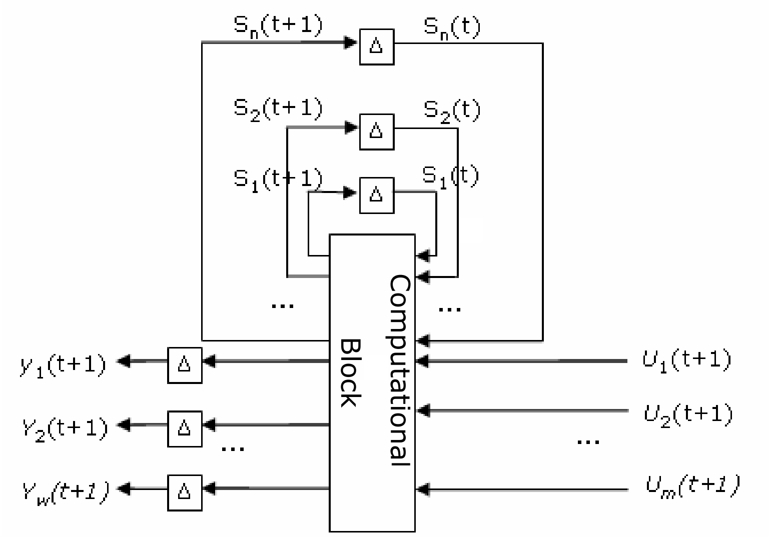


Figure 28. Structure of a state variable system

An error can affect the operators (adders, multipliers), used in the computational block, the system states, or the system outputs. An error in the computational block, responsible for computing the state, or an error in the system states will stay in the system for multiple clock cycles and can propagate to other states and the primary outputs of the system. An error restricted to a primary output does not propagate to other outputs or states and disappears after one clock cycle. For this reason, this chapter focuses on detecting and correcting errors either in the operators of the computational block that are involved in computing the system states or in errors affecting the system data registers (flip-flops) directly. A unified approach that can handle both cases is developed.

6.1.2 Concurrent Error Detection

In a linear state variable system, real number codes [59] can be implemented to encode the state vector, $s(t)$, using one or more check variables. These check variables can be used for the error detection and correction [61]. Each row i of the A and B matrices is scaled using a real-value weight α_i and summed to give the vectors X and Y below, respectively. Let the coding vector be the vector having the relevant weighting factors, i.e. $CV= [\alpha_1, \alpha_2,\ldots, \alpha_n]$. X is defined as $X=CV.A$. Similarly, Y is defined as $Y=CV.B$. A check variable c, corresponding to each coding vector is computed as: $c(t+1) = X.s(t)+Y.u(t)$. It is trivial to see that if there is no error in the system, then $c(t+1) = CV.s(t+1)$. Hence, an error signal, e, can be computed as: $e(t+1) = CV.s(t+1)\text{-}c(t+1)$ and is zero in the absence of any error. Figure 29 shows a state variable system with error detection capability.

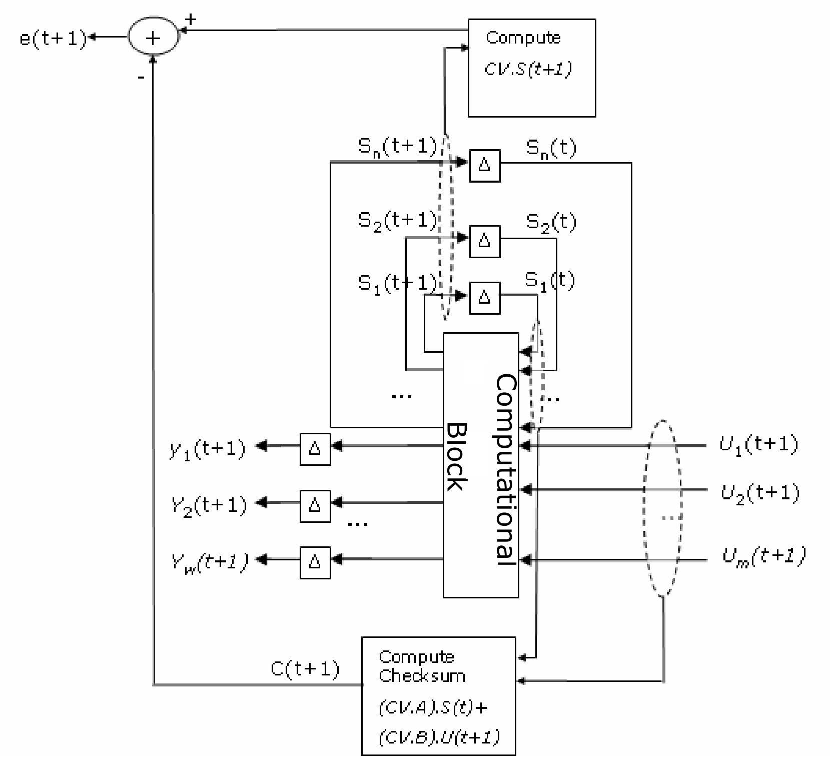


Figure 29. A state variable system with checksum-based error detection

It should be noted that in the presence of multiple erroneous states (caused by an erroneous operator shared by two or more states), the error signal might be zero. In [61], conditions on the coding vector for preventing such *error aliasing* based on the concept of gain matrix, is presented. A review of this is presented below.

Gain of an Operator:

The gain of an operator quantifies how an error in the operator affects different system states. To find out how an error in an operator O_j affects the i^{th} state, s_i, first we find all the paths, p_i, from the output of O_j to s_i. For each such path, we define the gain, Θ_i, to be the product of the gains of all the operators on that path. The gain of an adder (subractor) is +1 from its "+" input and -1 from its "-" input. The gain of multiplier is the multiplication constant. Let $g_{i,j}$, the total gain from O_j to s_i be $\sum_{i=1:P} \Theta_{\mathrm{i}}$, where P is the number of paths existing from the output of O_j to s_i. $g_{i,j}$ effectively represents the amount by which an error ε_j at the output of operator O_j is scaled before being added to the value of state s_i. In other words, an error ε_j in O_j causes an error $g_{i,j} \times \varepsilon_j$ in s_i. For example for the system shown in Figure 30, the gain of path $\underline{8, 9, 5, 3, S_3}$ from O_8 to S_3 is (1/3)(+1)(-1) = -1/3 and $g_{3,8}$ = (1/3)(+1)(-1)+(1/3)(1)(1)(-1) = -2/3 [61].

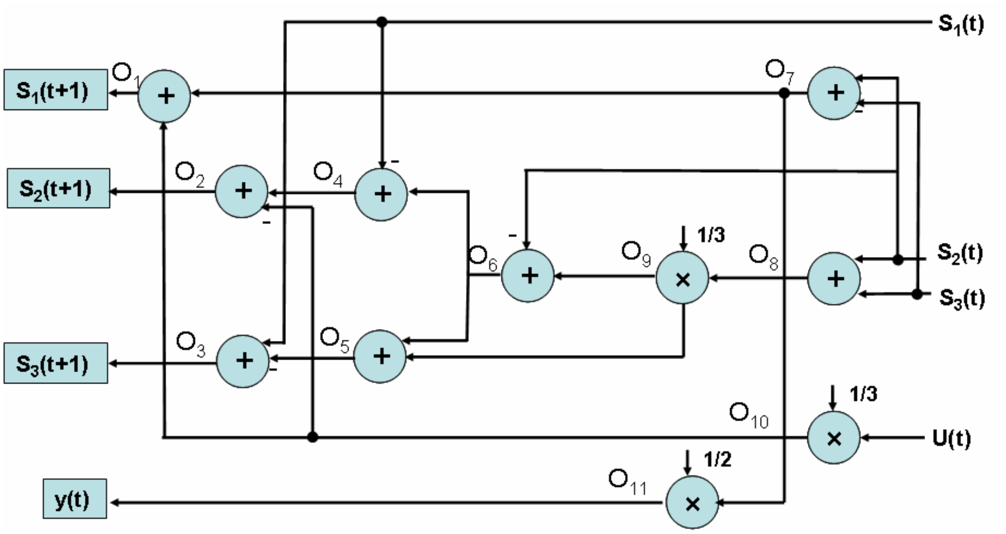


Figure 30. Structure of a linear State variable system with shared operators [courtesy of [61]].

Gain of a state:

Similar to the gain of an operator, one can define the gain of a state as follows.

$$\text{Gain of } s_i \text{ on } s_j = g_{ij} = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}.$$

In other words, the effect of an error in a state, s_i, is zero for all other states except itself. The gain matrix GM is an $n \times (o+n)$ matrix where n is the number of states and o is the number of operators involved in computing the system states. Let $N= o+n$. Each column of the gain matrix represents the gain of an operator or a state. Without loss of generality, we construct the gain matrix such that the first o columns represent the operators gain and the last n columns represent the gain of states. From here on, we refer to a potential source of error (either an operator involved or a state) as a *module,* where $module_i$ $i=1\ldots o$ are the operators and $module_i$ $i=o+1\ldots o+n$ represents the states. Figure 31 shows the gain matrix corresponding to the example system of Figure 30. It should be noted that since the operator O_{11} is not involved in computing any of the three states (i.e., only involved in computing the output), we do not include it in the gain matrix.

$$GM = \left(\overbrace{\underbrace{\begin{matrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & \frac{1}{3} & -1 \\ 0 & 0 & 1 & 0 & -1 & -1 & 0 & -2 & -\frac{2}{3} & 0 \end{matrix}}_{\text{Operators}}}^{\longleftarrow O_1 \ldots O_{10} \longrightarrow} \;\vdots\; \underbrace{\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}}_{\text{States}} \right)$$

Figure 31. Gain Matrix corresponding to the system in Figure 30

In [61], it is shown that to guarantee an error being observed on the error signal, the coding vector must be chosen such that all elements of the product $CV \times GM$ are non-zero.

A non-zero value of the error signal indicates an error either in states, $s(t+1)$, in the check variable, $c(t+1)$, or in the error signal, $e(t+1)$. One can use two coding vectors and their corresponding error signals to determine whether an error is in the system states or in the error (checksum) computation circuitry (false alarm) [61]. If two check variables are used, one can conclude that the error is restricted to the system states iff both error signals corresponding to the two check variables are non-zero. In this work, we do not use two check variables. Hence there is no way to determine if the error is in the original or the checking circuitry. Given that the state computation is more complex and involves more arithmetic operations than the check variable or the error signal computation, and the fact that in digital systems most soft errors occur in flip flops, the resulting probability of a false alarm is low. However, one can always use two check variables (as described above and in [61]) to identify false alarms, but at a higher hardware cost. Next, we describe the proposed probabilistic checksum-based compensation scheme.

6.1.3 Proposed Probabilistic Error Compensation

A single check variable detects an error in the system, but fails to identify the erroneous state. In [61], the authors provide a method that can identify the faulty state by using two check variables and carefully selecting the coding vector for each check variable. This is done in such a way that the ratio of the magnitude of error signals corresponding to the two checksums identifies the erroneous state. The amount of computation needed to find the faulty state is high enough that it makes the correction technique not suitable for real-time application. Furthermore, the technique becomes very complicated and difficult to implement for systems with shared hardware implementation. Here, we propose a probabilistic checksum-based error compensation to compensate for errors in both combinational and the sequential parts of a linear digital system to improve the system output quality (noise power or SNR).

6.1.3.1 Probabilistic Compensation: Overview

From the discussion in the prior section, if an error occurs in the time frame (*t*, *t*+1), then the vector *s*(*t*+1) has the wrong value for some of the system states. Therefore the error value *e*(*t*+1) computed in the time frame (*t*+1,*t*+2) is non-zero. The error signal is non-zero only for a single time interval and returns to zero in the next interval. Next, it is described how the error signal value is used to *probabilistically compensate* the system states within the time step (*t*+1,*t*+2) such that the overall system SNR is improved. Before proceeding to describing the technique, a few notations and definitions are introduced next.

Let y_{good} be the output signal when there is no error and y_{err} be the outputs when there is an error. The output noise is $noise = y_{good} - y_{err}$. The output noise power and the output SNR are defined as follows:

$$NoisePower = \sum_{i=1}^{T} noise(i)^2 \qquad (8)$$

$$SNR = 10\log_{10}\left(\frac{\mathrm{var}(y_{good})}{\mathrm{var}(noise)}\right) \qquad (9)$$

where *T* is the duration of the output signal and $noise(i)^2$ is the noise power component at time *i*. In the following, the output noise power is used as a metric to find the best compensation vector for the checksum-based probabilistic error correction technique.

An error occurring during the time interval (*t, t+1*) results in a deviation $EV = [es_1\ es_2 \ldots es_n]^T$ in the state values (i.e. the state s_i has an error es_i), from their correct values as given by Equation (10).

$$s_{err}(t+1) = s_{good}(t+1) + EV \qquad (10)$$

If there is no error correction, the error in the system states at time *t+k+1*, *k* cycles after its occurrence, assuming no other errors happen in between, can be computed as follows:

$$s_{err}(t+k+1) = A^{k} EV + s_{good}(t+k+1) \quad (11)$$

The error vector *k* cycles after the error occurrence is $A^{k}EV$. If the system is stable, the errors in the states disappear after *m* cycles, where $A^{m} \rightarrow 0$. The error in the output is $CA^{k}EV$, *k* cycle after an error occurs.

The goal of probabilistic checksum-based compensation is to compensate for the error in the system states in a *probabilistic sense*, such that the average output noise power is minimized. As opposed to deterministic error correction [61], no error diagnosis is performed after the error is detected (error diagnosis involves finding the single operator with the erroneous output in shared hardware systems). In deterministic error correction, after determining the source of the error (using a complex procedure involving the use of a look-up table), the error is compensated *exactly* by feeding back $e(t+1)$ with appropriate weights back into the system states in the time frame $(t+1, t+2)$. The weights with which $e(t+1)$ is fed back depends on the operator that is determined to be erroneous. In probabilistic checksum based compensation, since the erroneous operator is never diagnosed, the weights with which $e(t+1)$ is fed back to the system states are *independent of the operator that is erroneous* and determined in such a way that the overall system SNR is improved as long as the operator failure statistics is known. This significantly reduces the latency of error compensation and makes near real-time error compensation possible with significant improvement of SNR (described later).

It should be mentioned that other objective functions such as based on minimizing the worst case noise power are possible. The time-frame expansion of the linear state variable system with the times frames in which error detection (ED) and error compensation (EC) are performed is shown in Figure 32. Because of the delay overhead associated to the scheme, the clock period must be stretched to accommodate the delays associated to ED and EC modules. If the delay penalty is not acceptable, the ED could be done in parallel with actual computation ("Compute States & Outputs") in the time frame (t+1, t+2) to hide its delay. The EC, if needed, can be postponed to the cycle after the error detection. Although the approach imposes less delay overhead (because the ED runs in parallel with the state and output computation), it propagates the incorrect output for one cycle, hence increases the output noise.

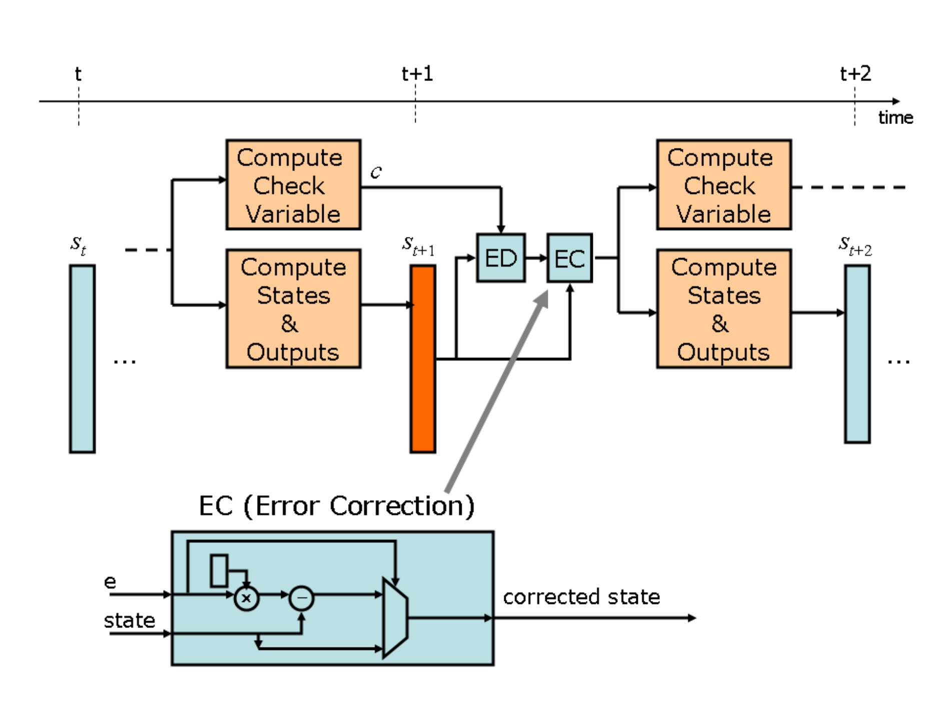


Figure 32. Checksum-based probabilistic state correction

For an error, ε, affecting the output of the j^{th} module, the error in the state s_i is $\varepsilon \times g_{i,j}$ by definition. It can be easily seen that the value of the error signal is as shown in Equation (12), where α_i is the i^{th} element of the coding vector as described earlier in this section.

$$e(t+1) = (\sum_{i=1}^{n} \alpha_i g_{i,j}) \times \varepsilon \qquad (12)$$

where n is the number of states in the system.

After detecting an error on the error signal, $e(t+1)$, if it was known that the j^{th} module is the erroneous one, then we can directly compute the error in each state as shown in Equation (13), where we substitute ε, the error in the j^{th} module, as a function of $e(t+1)$, derived from Equation (12).

$$e_{s\,i}\,|_{\mathrm{mod}\,ule_j} = g_{i,j} \times \varepsilon = \frac{g_{i,j} \times e(t+1)}{(\sum_{i=1}^{n} \alpha_i g_{i,j})} \qquad (13)$$

We can rewrite Equation (13) as follows, where the filter structure dependent parts ($g_{i,j}$) and coding vector dependant parts ($\alpha_1 \ldots \alpha_n$) are clumped together and represented as $\gamma_{i,j}$ (i.e. $\gamma_{i,j} = \frac{g_{i,j}}{(\sum_{i=1}^{n} \alpha_i g_{i,j})}$).

$$e_{s\,i}\,|_{\mathrm{mod}\,ule_j} = \gamma_{i,j} \times e(t+1) \qquad (14)$$

Let Δ_j be the error in the state vector when the j^{th} module is erroneous. The expanded form Δ_j is shown by Equation (15).

$$\Delta_{j=}\begin{bmatrix}\gamma_{1j}\\ \gamma_{2j}\\ .\\ .\\ .\\ \gamma_{nj}\end{bmatrix}\times e(t+1)=\gamma\times e(t+1) \qquad (15)$$

The goal is to find a compensation vector, *V,* an (n×1) vector, to compensate the error in the states (*EV*) when an error is detected. After compensation*,* the error in the states is *EV-V.* The error in the states and the output *k* cycles after the correction are $A^k(EV\text{-}V)$ and $CA^k(EV\text{-}V)$ respectively. The goal is to find the compensation vector, *V,* such that the average output noise power (shown in Equation (16)) is minimized.

$$AverageNoise=\sum_{i=1}^{N}\sum_{k=0}^{m}w_i\,(CA^k(\Delta_i-V))^2 \qquad (16)$$

where w_i is the probability of the i^{th} module being erroneous. A solution to the minimization problem, assuming $\sum_{k=0..m}CA^k\neq 0$, is given by Equation (17)(See Appendix I for more details).

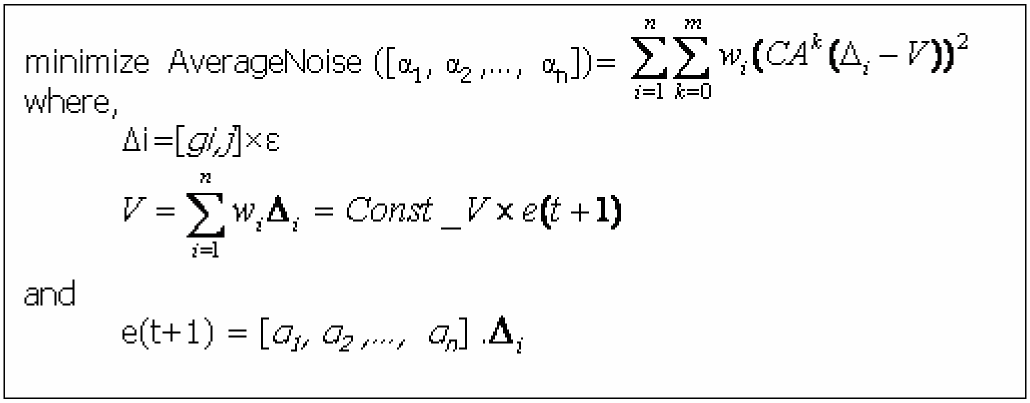
$$V=\sum_{i=1}^{N}w_i\Delta_i \qquad (17)$$

Using Equation (15), we can write the compensation vector, *V*, shown in Equation (17), as a function of the error signal as given below. It can be seen from Equation (18) that the compensation vector is written as a constant vector (*Const_V*=[$const_1 \ldots const_n$]) multiplied by the value of the error signal, $e(t+1)$. The vector *Const_V* is a constant (n×1) vector and known prior to system implementation, while $e(t+1)$ is known during the system execution. When an error is detected, we simply multiply the value of error signal by *Const_V* and then subtract it from the state value to do compensation.

$$
\begin{aligned}
V &= \sum_{i=1}^{N} w_i [\gamma_{1,i} \ldots \gamma_{n,i}]^T e(t+1) \\
&= (\sum_{i=1}^{N} w_i [\gamma_{1,i} \ldots \gamma_{n,i}]^T) \times e(t+1) \\
&= [const_1 \ldots const_n] \times e(t+1) \\
where \;\; const_j &= \sum_{i=1}^{N} w_i \times \gamma_{j,i}
\end{aligned}
\tag{18}
$$

In subsection 6.1.2, we provided the condition for the coding vector in order to detect all errors (i.e. to prevent errors from being concealed). The choice of coding vector is also important in the context of error compensation. As shown in Equation (18), the compensation vector depends on a *Const_V* and *e*(*t*+1), which both are functions of coding vector elements (α_i). Therefore the choice of coding vector affects the noise power reduction obtainable using the probabilistic error compensation and hence the quality of our compensation technique. Below, we find the coding vector, which results in the minimum average noise power when probabilistic compensation is applied.

The objective is to find the coding vector $CV = [\alpha_1, \alpha_2 ,..., \alpha_n]$ such that the average noise power after compensation is minimized. Recall that the average noise power is $AverageNoise = \sum_{i=1}^{n+o} \sum_{k=0}^{m} w_i (CA^k (\Delta_i - V))^2$, where *V* is given in Equation (17) and $\Delta_i=[g_{i,j}]\times\varepsilon$. It should be noted that Δ_i is independent of the coding vector values and only depends on the filter implementation, which manifests itself in the gain values. On the other hand, the compensation vector is a function of α_i. We would like to find α_i such that our objective function of minimizing the average noise power is satisfied. To find α_i values, we apply a simplex search method on Equation (16), using MATLAB fminsearch function ([81] and [82]). The technique is summarized in Figure 33. Since the fminsearch procedure is a local search, to avoid a local minimum, we repeat the procedure in Figure 33 with different initial conditions of *CV*.



minimize AverageNoise ($[\alpha_1, \alpha_2, \dots, \alpha_n]$)= $\sum_{i=1}^{n}\sum_{k=0}^{m} w_i(CA^k(\Delta_i - V))^2$

where,

$\Delta i = [gi,j] \times \varepsilon$

$$V = \sum_{i=1}^{n} w_i \mathbf{\Delta}_i = Const\_V \times e(t+1)$$

and

$e(t+1) = [a_1, a_2, \dots, a_n] . \mathbf{\Delta}_i$

Figure 33. To find the optimum coding vector (MATLAB `fminsearch` is used to solve the optimization)

6.1.4 State Partitioning and Checksum Design

By minimizing the average noise power using Equation (17), it is possible that in specific instances where errors occur in only a *subset* of the total number of operators, the injected noise power in the system with compensation is *larger* than the injected noise power in the system without compensation. Ideally, we desire Equation (19) to be satisfied individually for all states and operators (adders and multipliers) in the system.

$$\sum_{k=0}^{m}(CA^k(\Delta_i - V))^2 \le \sum_{k=0}^{m}(CA^k\Delta_i)^2 \qquad i = 1 \dots N \quad (19)$$

However, it may not possible to satisfy Equation (19) for all operators in the system with a single check variable. In other words, the noise power reduction possible with the use of a single check variable is limited and can be increased if multiple check variables (checksums) are used. Clearly, such multiple checksums must be computed across carefully selected subsets of the overall set of system states. To resolve the above issues, we provide a heuristic algorithm that partitions the system states, where each subset of states, corresponding to a partition, is monitored for errors by a separate check variable. The partitioning heuristic aims to satisfy Equation (19) for as many states and operators as possible while using Equation (17) to perform error compensation to minimize the noise power corresponding to the subset of states in each individual state partition. For instance if s_1 and s_3 are the two states being monitored for error detection/correction in a system with four states ($s_1 \ldots s_4$), then in calculating $c(t+1) = X.s(t)+Y.u(t)$, the vectors X and Y have zeros as their second and fourth elements (i.e. in positions corresponding to states not being monitored). In this case, the error signal is computed as $e(t+1) = [\alpha_1\ 0\ \alpha_3\ 0].s(t+1)\text{-}c(t+1)$.

In the following, we explain a heuristic procedure for partitioning the system states into different subsets, each of which is monitored by a single check variable. The use of a maximum of k check-variables (state subsets) is allowed. The heuristic is described in Figure 34. The heuristic procedure first finds k states that generate the highest noise power at the output if being erroneous and assigns them to the k check variables. The remaining states are assigned to different check variables such that the maximal noise power reduction is obtained. In Figure 34, $Monitor_i$ is the set of all states being monitored by the i^{th} checksum. After state partitioning, the checksum of all the states in each partition is computed separately and probabilistic compensation is performed for each subset. The compensation vector is found based on the subset of chosen states, using Equation (17). It should be noted that only operators that are involved in computing at least one of the states in the subset “monitored” and only states that are monitored are considered while computing the compensation vector. Furthermore, while using Equation (17), Δ_i has zeros in locations corresponding to states not being monitored. The time complexity of the heuristic algorithm is $O\ (k.n^2)$.

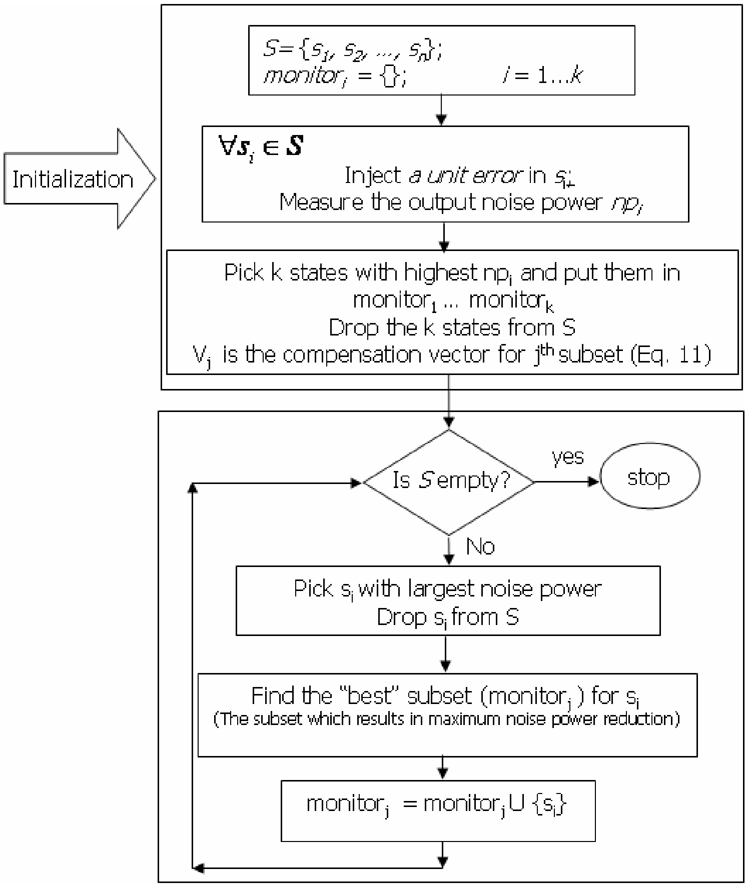

S={s1, s2, ..., sn};
monitori = {}; i = 1...k
Initialization
∀si ∈ S
Inject a unit error in si;
Measure the output noise power npi
Pick k states with highest npi and put them in monitor1 ... monitork
Drop the k states from S
Vj is the compensation vector for jth subset (Eq. 11)
Is S empty?
yes
stop
No
Pick si with largest noise power
Drop si from S
Find the "best" subset (monitorj) for si
(The subset which results in maximum noise power reduction)
monitorj = monitorj U {si}

Figure 34. A heuristic to find the best subsets of states to be monitor using different check variables

The next section presents the evaluation results of the checksum-based probabilistic error correction. The technique is compared with another approach called *state restoration*. The state restoration approach simply sets the state latches to their previous values whenever an error is detected. State restoration uses the real checksum code for error detection. The technique requires extra set of registers. State restoration is in fact a Prediction-based ANT [73] technique in its simplest form. The only difference is that in the prediction-based ANT, the prediction is done on the output (rather than the state) and that a comparator is used for error detection (rather than the checksum code).

6.2 EVALUATION

The experimental results of this section are generated using the 3rd order linear state variable system, shown in Table 10 unless specified otherwise. The linear system, the error detection, and the error correction modules were implemented in MATLAB. Also, the errors in different operators were emulated by modifying the magnitude of states proportional to the gain of the faulty operator for each state. For the checksum-based probabilistic error compensation, the compensation vector in Equation (17) is used. The optimum coding vector was obtained using the optimization shown in Figure 33. The input is a sinusoid with a maximum amplitude of 1 and with a frequency of 10 KHz sampled at 10 times the Nyquist rate, unless specified otherwise. The simulation time is assumed to contain 20 periods of the sine wave. It is assumed that the erroneous probability of all modules is the same, i.e.$w_i=1/N$, where N is the number of modules (i.e. the total number of states and operators).

Table 10. A 3rd order linear state system

$A = \begin{bmatrix} 0.3 & 0 & 0 \\ -0.3 & -0.3 & -0.9 \\ 0.8 & 0.3 & 0.1 \end{bmatrix}$	$B = [2.6\ \ 1.2\ 1.5]^T$
C= [0.11 0.06 0.08]	D = [0.2]

An implementation of the system and its corresponding gain matrix are shown in Figure 35 and Figure 36.

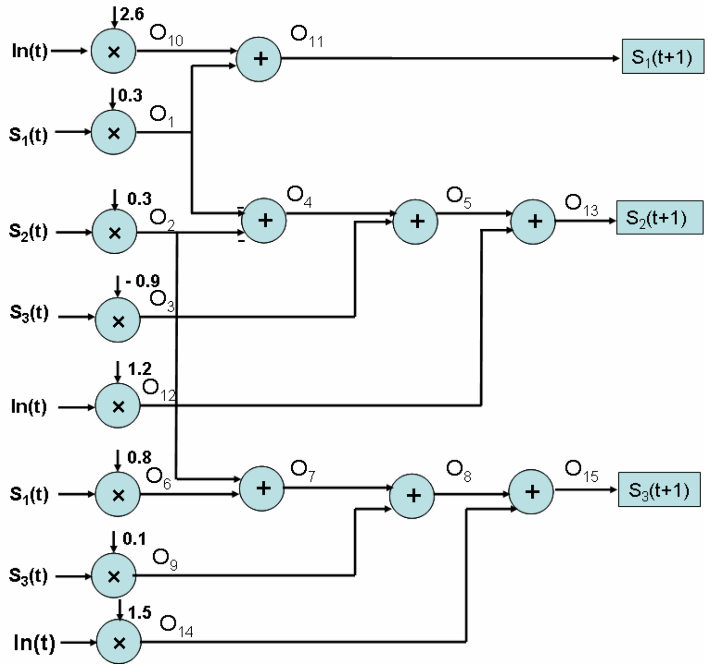


Figure 35. An implementation of the system in Table 10 with shared operators

$$GM = \left(\begin{array}{c|c} O_1 \dots O_{15} & O_{16}\ O_{17}\ O_{18} \\ \underbrace{\begin{matrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{matrix}}_{\text{Operators}} & \underbrace{\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}}_{\text{States}} \end{array}\right)$$

Figure 36. The gain matrix corresponding to the implementation in Figure 35

The injected error is modeled as a function of four random variables, defined as follows: 1) Error magnitude (EM), 2) Burst length (BL), the number of errors in a burst, 3) Burst-to-burst time (BBT), i.e. the time interval between two bursts of errors, and 4) Error-to-error time (EET), the time interval between two consecutive errors in a burst. Additionally, the time of occurrence of the first burst is another random variable, called *error position* (see Figure 37). Except when burst errors are being studied, a single error (i.e. BL = 1 and BBT = ∞), with EM = 1 is considered.

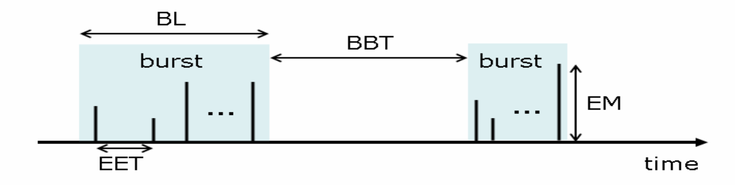


Figure 37. Error model

As mentioned in Section 6.1.3, the choice of coding vector affects the SNR improvement obtained by using the probabilistic error compensation. In [59], it is shown that the choice of coding vector is in fact a trade off between the round-off error, caused by increasing the values of coding vector elements, and the code reflectivity reduction, caused by smaller coding vector values. Therefore, in this work we assume that there is a limited range [$x_{\min}$, $x_{\max}$] of acceptable values for coding vector elements. The optimum coding vector for the system shown in Table 10, using the optimization formula in Figure 33 and MATLAB fminsearch simplex-based optimizer, is [5 3 4]. It should be noted that if CV_{opt} is an optimum coding vector, then any scaled version of CV_{opt} is still an optimum coding vector and will result in the same noise power reduction when used for probabilistic compensation. Using this characteristic of optimum coding vector, one can scale it so that the optimum coding vector values fall in the acceptable range of [$x_{\min}$, $x_{\max}$].

The noise power is shown in Figure 38 for different erroneous modules. The Figure also shows the noise power when no error compensation is present. The results of SNR improvement for different erroneous modules are also shown in Figure 38. For this system, the average noise power is reduced by more than 60% over the system with no correction and an average SNR improvement of 7.7 dB was obtained. In the remaining of this section, we analyze how the checksum-based probabilistic compensation and state restoration technique perform under different error statistics, including error position, error magnitude, and burst errors.

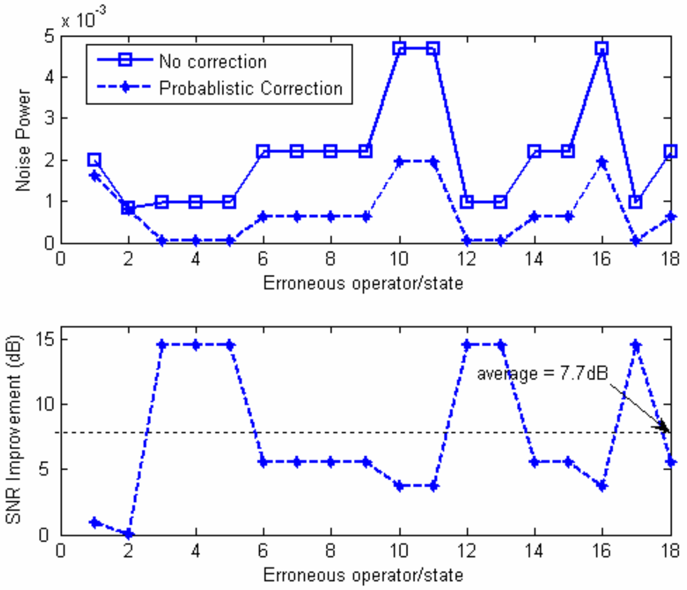


Figure 38. Noise Power reduction and SNR improvement using the optimal coding vector

The effect of error position:

To analyze the effect of error position, a single error was introduced at all possible positions within a single period of the input. All errors were injected in s_3, but the results hold for other erroneous operators and states. The results are shown in Figure 39. The checksum-based probabilistic compensation SNR does not depend on the error position and provides a constant 6.9 dB improvement over the no correction case. The state restoration SNR strongly depends on the error position. At those positions, where the states have the least derivative, the state restoration shows its best performance. The state restoration method results in 12.1 dB, -0.2 dB, and -5.6 dB SNR improvements (over the no correction case) for the best case, average case and worse case respectively.

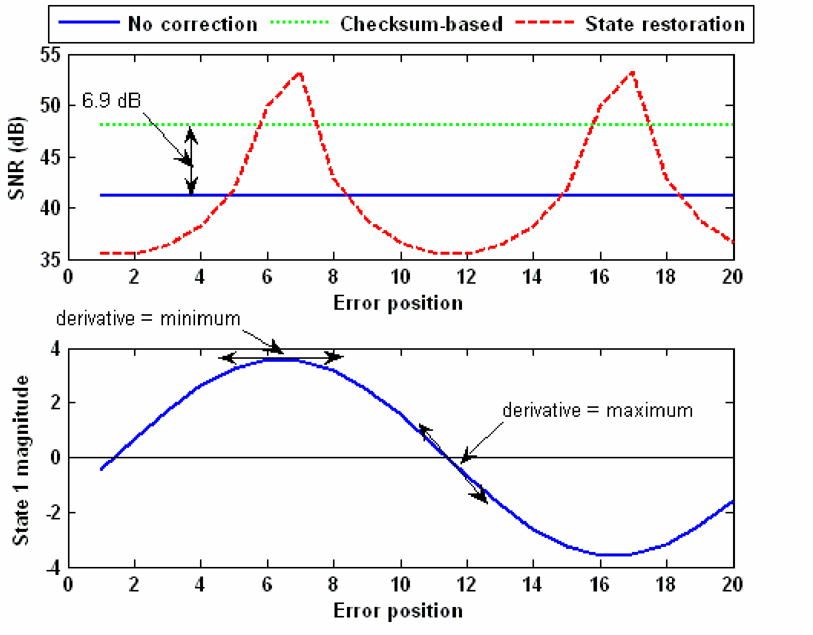


Figure 39. SNR as a function of error position

The effect of error magnitude:

The SNR values for two different error magnitudes, EM = 1 and EM = 0.5, are shown in Figure 40. The plots for the SNR of the state restoration technique are identical and overlapping. The figure shows that the improvement of the checksum-based probabilistic compensation over the no correction case stays constant regardless of the error magnitude. Therefore, one can conclude that for a single error, regardless of the error position and error magnitude, checksum-based probabilistic correction results in a constant SNR improvement. Appendix II proves that the SNR improvement using checksum-based probabilistic compensation is independent of the error magnitude.

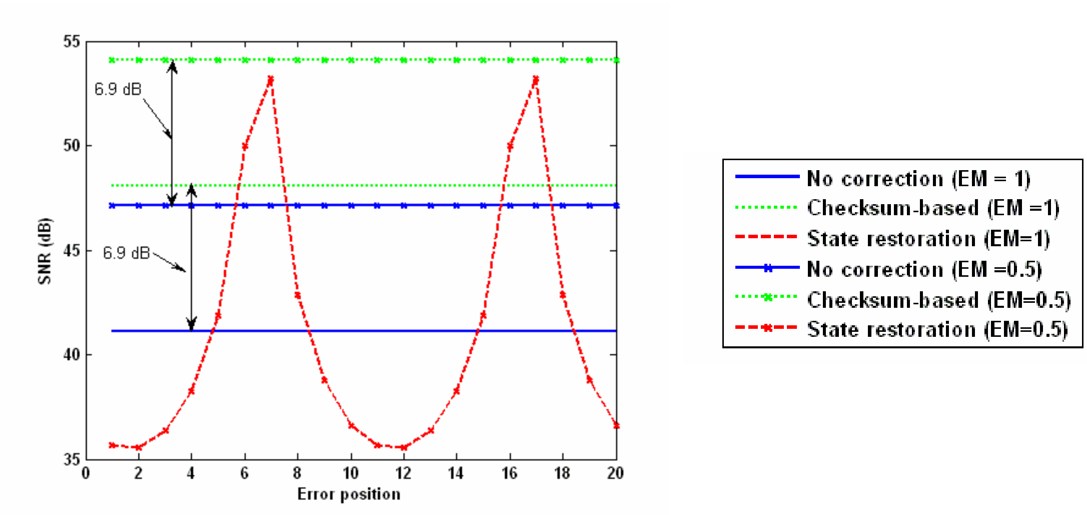


Figure 40. Effect of error position on SNR for various error magnitudes

The effect of sampling frequency:

The effect of the sampling frequency on the SNR and SNR improvement of different techniques can be seen in Table 11. For this result, the error of magnitude 1 was injected in s_3. As expected the SNR reduces as the sampling frequency decreases. The probabilistic correction technique always maintains a fixed SNR improvement of 6.9 dB over the no correction case. The SNR improvement of this technique is not affected by the sampling frequency. The SNR improvement of the state restoration technique strongly depends on the sampling frequency and for lower sampling rates, which are more practical, on average it performs worse than the checksum-based probabilistic compensation or even worse than no correction.

Table 11. Effect of sampling frequency on SNR

Sampling frequency	No correction SNR (dB)	Probabilistic correction SNR (dB)	State restoration (dB)		
			Max	Min	Average
4×input frequency	31.9	38.8	17.1	14.1	15.6
8×input frequency	36.6	43.5	32.5	23.8	27.7
16×input frequency	40.1	47.0	48.3	32.7	37.8
32×input frequency	43.3	50.2	63.6	41.6	47.3
64×input frequency	46.3	53.2	78.7	50.7	56.5

The effect of burst error:

Figure 41 shows the SNR of different schemes as a function of burst length. A single burst is assumed, i.e. BBT = ∞. Also errors occur in consecutive cycles, i.e. EET=1 and EM=1. For the case of state restoration, only the average SNR is shown. In the top graph of Figure 41, all errors are injected in s_1. In the bottom graph of Figure 41, all errors are injected in s_3. The figure shows that the SNR of the state restoration reduces drastically as the burst length increases. Although the SNR of the checksum-based probabilistic error correction also reduces with the burst length, the reduction is not as drastic as in the state restoration and always achieves a positive SNR improvement over the no correction case.

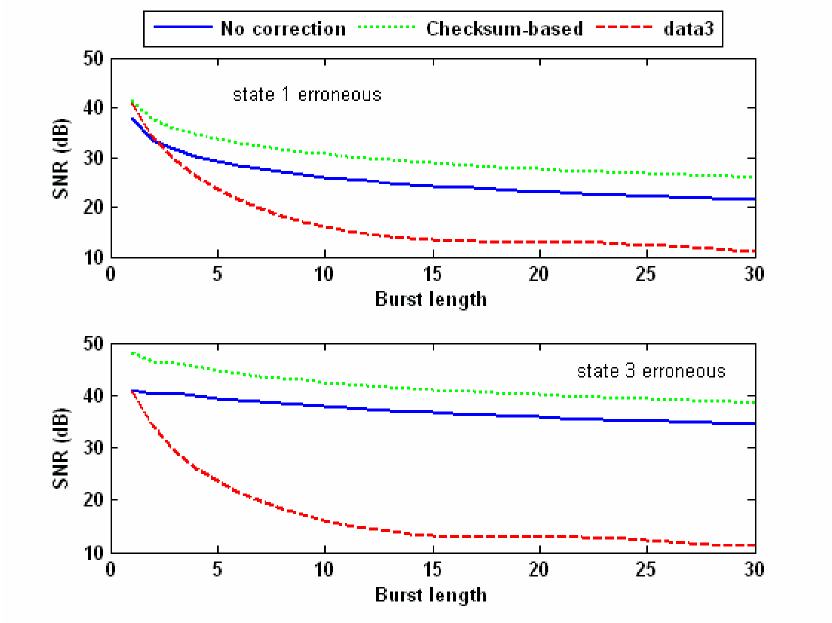


Figure 41. SNR as a function of burst length

Figure 42 shows the SNR improvement of checksum-based probabilistic correction over no correction. For the case of errors in $s_{1,}$ having a longer burst improves the SNR obtained using checksum-based probabilistic compensation. In this case, the SNR improvement initially increases with burst length increase, but stays almost constant at 4.6 dB for burst lengths greater than 20. In the case where all errors are in $s_{3,}$ the SNR improvement reduces with burst length.

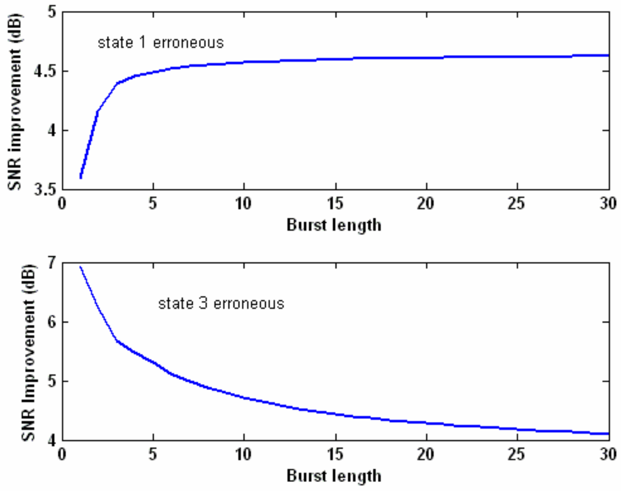


Figure 42. SNR improvement as a function of burst length (using probabilistic checksum-based compensation)

Whether a larger burst length improves the SNR achievable (i.e. reduces the output noise power) using the checksum-based probabilistic compensation depends on the filter structure and is a complex phenomenon. Here, we explain why there are cases where larger burst lengths improve the SNR of our probabilistic technique using BL = 2. Note that if a single error stays in the system, for *m* cycles before dying out, the noise at the output for the no correction case and the probabilistic checksum-based compensation are as follows (subscripts *nc* and *pc* represent the no correction and checksum-based probabilistic correction respectively):

BL = 1 (single error):

$$noise_{nc} = \{0,...,0, CA^0 E, CA^1 E, CA^2 E,..., CA^m E,0,...0\}$$

$$noise_{pc} = \{0,...,0, CA^0 (E-V), CA^1 (E-V), CA^2 (E-V),..., CA^m (E-V),0,...0\}$$

where |*noise*$_{nc}$ |= |*noise*$_{pc}$| = T, the duration of simulation and *noise*$_{nc}$ has *m* non-zero elements.

For the case of BL = 2:

$$\begin{aligned} noise_{nc} = \{&0,...,0, A^0 E, C(A + A^0)E, C(A^2 + A^1)E, C(A^3 + A^2)E,...,\\ &C(A^m + A^{m-1})E, CA^m E, 0,...0\} \end{aligned}$$

$$\begin{aligned} noise_{pc} = \{&0,...,0, A^0 (E - V), C(A + A^0)(E - V), C(A^2 + A^1)(E - V), C(A^3 + A^2)(E - V),...,\\ &C(A^m + A^{m-1})(E - V), CA^m (E - V), 0,...0\} \end{aligned}$$

One can see that depending on the filter (*A*, *C*) and the source of error (*E*), the noise power may or may not decrease with the increase in burst length. For instance, it may or may not be the case that $C(A^2 + A)\,E$ is greater than $C(A^2 + A)\,(E\text{-}V)$.

In the analysis presented up to this point, the effects of error magnitude, burst length, and error position were studied individually, while the rest of parameters, which define the error characteristics, were constants. For the next experiment, a distribution is assumed for each of the random variables on which the error statistic depends, i.e. EM, BL, BBT, EET, and error position. For these error statistics (Table 12), a distribution for SNR of each scheme was obtained (Figure 43). For this experiment, we used longer simulation time so that multiple burst would be possible. The simulation time is assumed to contain 100 periods of the sine wave and the error could be in any module with the equal probability. The histogram has 1000 data points. The plot shows that on average, the SNR improvement is 4.7 dB. We did not include the state restoration plot since the technique performs poorly in the presence of burst errors.

Table 12: Different error parameter distributions (*T* is the duration of the output)

EM	BL	BBT	EET	Error Position
Uniform[0.5,1]	*Uniform*[1,10]	*Uniform*[1,400]	*Uniform*[1,3]	*Uniform*[10,200]

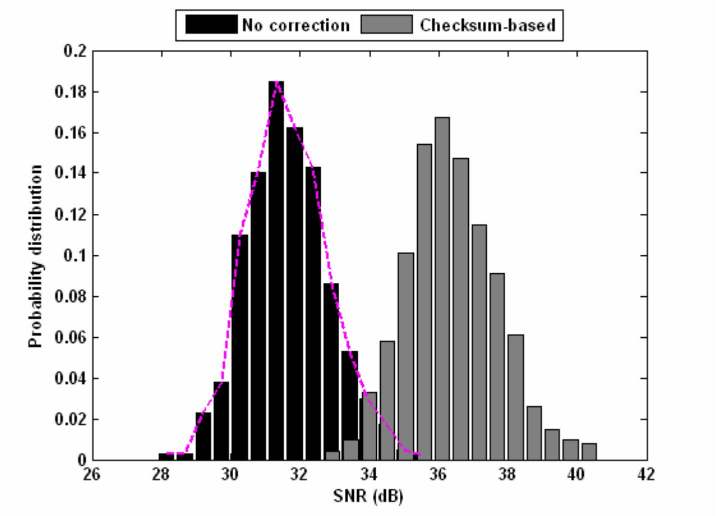


Figure 43. SNR distribution of three different techniques

Larger systems:

So far while analyzing the checksum-based probabilistic scheme, the checksum variable is used to detect errors on all states. However, it is possible to use the checksum variable to detect errors only in a subset of states. This becomes an important issue when the system is large and we have a limited budget in terms of area and power to spend on the error detection and error compensation circuitry. In this case, it is beneficial to monitor the states which are more responsible in bringing down the SNR and monitor those states as explained in Section 6.1.3. Here, for a 10th-order filter, we use the partitioning heuristic to monitor different subsets for different check variables. The results are compared against the case where for *k* available check variable, the first *n/k* states are monitored by the first check variable, the second *n/k* states are monitored by the second check variable, and so on (Table 13). In this example, for the case of 1 checksum, the heuristic slightly underperforms compared to the case with no partitioning, but for the case of 2 and 3 check variables, it outperforms the partitioning done based on state number identification. Regardless of what partitioning method is used, Table 13 shows that the significant SNR improvement can be achieved even for large systems.

Table 13. SNR improvement for an order 10 system

# of check variables	Partition using heuristic				Partition based on state number (id)			
	Partitions	SNR improvement (dB)			Partitions	SNR improvement (dB)		
		Max	Min	Mean		Max	Min	Mean
1	{2 3 4 5 6 8 9}	10.3	0	6.5	{1,2,3,…, 10}	15.0	1.6	6.9

2	{1,2, 6,7,8,9,10}
	{3,4,5}
		15.9	2.9	8.7	{1,2,…,5}
					{6,7,…,10}
						11.2	2.0	7.0

3	{4,5,6,7,8,9}
	{ 3}
	{1 2 10}
		18.1	5.3	9.1	{1,2,3}
					{4,5,6}
					{7,8,9,10}
						12.7	2.6	7.8

<u>Hardware and Power overhead:</u>

In order to evaluate the power and area overhead of the proposed probabilistic checksum based error compensation, we implemented the linear system,and the error detection and compensation circuitry in Verilog. We obtained the power and area estimation using *Synopsys Design Compiler* and the 0.25 μm standard cell library. The timing, area, and dynamic power consumption of each technique is shown in Table 14. The table shows that the checksum-based probabilistic compensation has the least overhead compared to the state restoration and TMR. Only 11% delay overhead and less than 13% area overhead was imposed in the case of the proposed probabilistic compensation compared with no correction case. State restoration has 15% and 1X delay and area overhead respectively. It should be noted that while the area overhead of TMR is the most (as expected), it has very little delay overhead, 1.8%.

Table 14. Delay, area, and power associated with each technique

Technique	Timing (*ns*)	Area	Dynamic Power (*mW*)
No correction	5.5	185,002	56.0
Checksum-based probabilistic compensation	6.1	208,641	55.0
State restoration	6.3	376,717	105.8
TMR	5.6	519600	161.1

6.3 CONCLUDING REMARKS

In this chapter, we showed how the checksum-based probabilistic error compensation can be used not only to mitigate the error occurring in flip flops but also errors in the combinational parts. This is an important problem because down the technology road map, transient errors in the combinational part of a system will become as important as errors in the sequential part. The proposed technique results in a large SNR improvement in a linear digital system. For the 3rd order example system presented here up to 13 dB improvements was achieved. More recently, the technique was extended to cover the transient errors in non-linear systems using a technique, called time-freeze linearization. The details of this technique can be found at [83]. It shows that the probabilistic checksum-based compensation is a powerful technique, which can cover both linear and non-linear systems and can handle errors in both combinational and sequential components.

CHAPTER 7

CONCLUSIONS

This chapter summarizes the main contributions of this thesis and provides possible directions for future research in this domain. The objective of the performed research was to develop circuit-level techniques to address process variations and transient errors in scaled CMOS circuits. The proposed techniques can be divided into two parts. The first part addresses the issues related to process variations and proposes techniques to reduce the variation effects on power and performance variations. The second part deals with the transient errors and techniques to reduce the effect of transient errors with minimum hardware or computational overhead instead of eliminating them, which would require an excessive amount of redundancies.

7.1 VARIATION TOLERANT DESIGN

With the increase in process variations in the CMOS technologies, power and performance variations become major concerns of circuit designers. Techniques such as the use of forward/reverse body bias and voltage scaling are commonly used to bring down the delay and power consumption specifications in the acceptable range. Variation-aware circuit sizing is another technique used in the design stage to have a more variation-tolerant design.

The key goal of this research was to provide techniques for designing more variation-tolerant circuits. We proposed to attack the problem both at the design stage and at the post-fabrication stage. The latter requires the feasibility of having ways of specification tuning and a fast and efficient framework that makes the post-silicon tuning attractive. The summary of the proposed techniques is as follows:

- Addressing the huge leakage variation by looking at the effect that the gate placement has in leakage distribution. The main idea of this work is the subject of Chapter 3.
- Developing an architectural framework for post-silicon testing and tuning to bring the performance of the circuit within the acceptable range. Also, a tunability feature was studied. A modified form of CMOS gate that can be programmed to work in a low-speed or high-speed modes is presented. The ideas are discussed in Chapter 4.

7.2 TRANSIENT ERROR TOLERANT DESIGN

Reliability is another major concern for the CMOS technologies beyond 90 nm. According to ITRS 2003, “Relaxing the requirement of 100% correctness for devices and interconnects may dramatically reduce costs of manufacturing, verification and test.” In other words, it is hard to achieve 100% correctness because of an increase in transient error rate. Such an increase is assumed to be driven by the aggressive technology scaling and is associated with the reduced noise margin, power/ground bounce and radiation-induced effect or because of permanent failures on internal signal lines that are excited intermittently by real-time stimulus. At the same time, the classical fault-tolerant techniques are all proven to be too costly to be used in non-critical applications. Following this trend and the observation that for many DSP applications, it is not necessary to maintain a cycle-to-cycle accurate computation as long as the system level quality of service metrics are satisfied or degraded within acceptable levels, a real-time probabilistic compensation technique for DSP applications was proposed. The objective of technique is to improve the quality of service of the DSP application, using very little hardware overhead. The work is discussed in Chapter 6.

7.3 FUTURE WORK

Several possible future research direction based on this work are summarized below.

- In this research a modified version of CMOS gates was proposed. Such modified version can be used to effectively tune the circuit performance after manufacturing to compensate for process variation and to increase parametric yields. Innovative techniques that enable post-manufacturing tuning will be of great interest as yield may be unacceptably low due to process variation. One of such novel technique is proposed in [84], which can effectively recover from manufacturing defects and process variation.
- A probabilistic compensation technique was proposed for mitigating the effect of transient errors. By using such compensation technique in the baseband system, the baseband circuitry can operate at lower supply voltage to allow some level of transient errors for certain inputs which exercise the longer paths of the circuit while saving power.

APPENDIX I

BEST COMPENSATION VECTOR FOR PROBABLISTIC COMPENSATION

We want to find the compensation vector, *V*, such that $AverageNoise = \sum_{i=1}^{n+o}\sum_{k=0}^{m} w_i (CA^k(\Delta_i - V))^2$ is minimized. Let $b_k = CA^k$.

$$f(V) = AverageNoise = \sum_{i=1}^{n+o}\sum_{k=0}^{m} w_i (b_k.(\Delta_i - V))^2$$

$$f(V) = \sum_{i=1}^{n+o}\sum_{k=0}^{m} w_i \sum_{j=1}^{n} \ (b_k(j).(\Delta_i(j) - V(j)))^2$$

$$\frac{\partial f}{\partial V(j_0)} = \sum_{i=1}^{n+o}\sum_{k=0}^{m} w_i \sum_{j=1}^{n} (-2 \times b_k(j_0))(b_k(j).(\Delta_i(j) - V(j))) = 0$$

$$\sum_{i=1}^{n+o}\sum_{k=0}^{m} w_i \sum_{j=1}^{n} (-2 \times b_k(j_0))(b_k(j).(\Delta_i(j) - V(j))) = 0$$

$$\sum_{i=1}^{n+o}\sum_{k=0}^{m} w_i b_k(j_0)(b_k.(\Delta_i - V)) = 0$$

$$\sum_{i=1}^{n+o}\sum_{k=0}^{m} w_i b_k(j_0) b_k.\Delta_i - \sum_{i=1}^{n+o}\sum_{k=0}^{m} w_i b_k(j_0) b_k.V = 0$$

$$\sum_{k=0}^{m} b_k(j_0) b_k [\ \sum_{i=1}^{n+o} w_i.\Delta_i - \sum_{i=1}^{n+o} w_i V \ ] = 0$$

$$\sum_{k=0}^{m} b_k(j_0) b_k [\ \sum_{i=1}^{n+o} w_i.\Delta_i - V \ ] = 0$$

$$V = \sum_{i=1}^{n+o} w_i.\Delta_i$$

Taking the second-order partial derivative shows that Equation (1) is in fact a minimum point.

$$\frac{\partial^2 f}{\partial V(j_0)^2} = \sum_{i=1}^{n+o} \sum_{k=0}^{m} w_i \sum_{j=1}^{n} (+2 \times b_k{}^2 (j_0)) \geq 0$$

APPENDIX II

RELATIONSHIP BETWEEN SNR IMPROVEMENT OF THE PROBABLISTIC COMPENSATION AND ERROR MAGNITUDE

Below, it is shown mathematically that the SNR improvement of the checksum-based probabilistic correction over the no correction case is independent of error magnitude. In the argument below subscripts *nc* and *pc* represent the no correction and checksum-based probabilistic correction respectively.

$$SNR_{nc} = 10\log10(\frac{\mathrm{var}(y_{good})}{\mathrm{var}(noise_{nc})})$$

$$SNR_{pc} = 10\log10(\frac{\mathrm{var}(y_{good})}{\mathrm{var}(noise_{nc})})$$

$$SNR_{improve} = SNR_{pc} - SNR_{nc} = 10\log10(\frac{\mathrm{var}(noise_{nc})}{\mathrm{var}(noise_{pc})} \qquad (1)$$

It can be easily verified that:

$$noise_{nc} = \{0,...,0,CE,CA^1E,CA^2E,...,CA^mE,0,...0\}$$

$$noise_{pc} = \{0,...,0,C(E-V),CA^1(E-V),CA^2(E-V),...,CA^m(E-V),0,...0\}$$

where |noisenc| = |noisepc| = T, the duration of simulation.

When the noise magnitude changes from *em_1* to *em_2*, the error vector $E_2 = em_2/em_1 \times E_1$. In other words, E_2 = *constant*×E_1. Therefore, using the fact that *var* $(ax) = a^2 \times var(x)$, we have:

$$\mathrm{var}(noise_{nc})\,|_{em_2} = (cons\tan t)^2 \times \mathrm{var}(noise_{nc})\,|_{em_1} \qquad (2)$$

Since $V = [w_i] \times E,$

$$\mathrm{var}(noise_{pc})\,|_{em_2} = (cons\tan t)^2 \times \mathrm{var}(noise_{pc})\,|_{em_1} \qquad (3)$$

From (1), (2), and (3) it can be concluded that the SNR_{improv} is independent of error magnitude.

REFERENCES

[1]. S. Nassif, “Within-Chip Variability Analysis", Proc. IEDM 1998, pp. 283 – 286.

[2]. S. Nassif, "Design for Variability in DSM Technologies", Proc. ISQED 2000, pp. 451 – 455.

[3]. B. E. Stine, et al., “Analysis and Decomposition of Spatial Variation in Integrated Circuit Processes and Devices”, IEEE Trans. on Semiconductor Manufacturing, Vol. 10, No. 1, Feb. 1997, pp. 24 – 41.

[4]. M. Orshansky, et. al, “Impact of Systematic Spatial Intra-Chip Gate Length Variability on Performance of High Speed Digital Circuits”, Proc. ICCAD 2000, pp. 62 – 67.

[5]. A. Chandrakasan, et al., Design of High-Performance Microprocessor Circuits, Wiley-IEEE Press 2000.

[6]. S. Borkar, et al., “Parameter Variations and Impact on Circuits and Microarchitecture”, Proc. DAC 2003, pp. 338 – 342.

[7]. J. W. Tschanze, et al., “Dynamic Sleep Transistor and Body Bias for Active Leakage Power Control of Microprocessors”, IEEE Journal of Solid-State Circuits, Vol. 38, No. 11, Nov. 2003, pp. 1838 – 1845.

[8]. J. P. Uyemura, “Introduction to VLSI Circuits and Systems”, John Wiley and Sons 2002.

[9]. J. Tschanz, et al., “Adaptive Body Bias for Reducing Impacts of Die-to-Die and Within-Die Parameter Variations on Microprocessor Frequency and Leakage”, IEEE Journal of Solid-State Circuits, Vol. 37, No. 11, Nov. 2002, pp. 1396 – 1402.

[10]. A. Keshavarzi, et al., “Technology Scaling of Optimum Reverse Body Bias for Standby Leakage Power Reduction in CMOS IC’s”, Proc. ISLPED 1999, pp. 252 – 254.

[11]. T. Kuroda and M. Hamada, “Low-Power CMOS Digital Design with Dual Embedded Adaptive Power Supplies”, IEEE Journal of Solid-State Circuits, Vol. 35, April 2000, pp. 652 – 655.

[12]. C. Chen, et al., “On Gate Level Power Optimization Using Dual-Supply Voltages”, IEEE Trans. VLSI, Vol. 9, Issue 5, Oct. 2001, pp. 616 – 629.

[13]. T. Chen and S. Naffziger, “Comparison of Adaptive Body Bias (ABB) and Adaptive Supply Voltage (ASV) for Improving Delay and Leakage under the Presence of Process Variation”, IEEE Trans. on VLSI, Vol. 11, No. 5, Oct. 2003, pp. 888 – 899.

[14]. J. Tschanz, et al., “Effectiveness of Adaptive Supply Voltage and Body Bias for Reducing Impact of Parameter Variations in Low Power and High Performance Microprocessors”, Symp. On VLSI Circuits 2002, pp. 826 – 829.

[15]. C. H. Kim, et al., “A Process Variation Compensating Technique for Sub-90-nm Dynamic Circuits”, Symp. on VLSI Circuits 2003, pp. 205 – 206.

[16]. O. Coudert, “Gate Sizing for Constrained Delay/Power/Area Optimization”, IEEE Transactions on VLSI Systems, Vol. 5, Issue 4, Dec. 1997, pp. 465 – 472.

[17]. J.P. Fishburn, “LATTIS: An Iterative Speedup Heuristic for Mapped Logic”, Proc. DAC 1992, pp. 488 – 491.

[18]. D. S. Chen and M. Sarafzadeh, “An Exact Algorithm for Low Power Library-Specific Gate Re-Sizing”, Proc. DAC 1996, pp. 783 – 788.

[19]. P. Pant, et al., “Dual-Threshold Voltage Assignment with Transistor Sizing for Low Power CMOS Circuits”, IEEE Trans. on VLSI Systems, Vol. 9, Issue 2, April 2001, pp.390 – 394.

[20]. Y.S. Dhillon, et al., “Algorithm for Achieving Minimum Energy Consumption in CMOS Circuits Using Multiple Supply and Threshold Voltages at the Module Level”, Proc. ICCAD 2003, pp. 693 – 700.

[21]. S. H. Choi, et al., “Novel Sizing Algorithm for Yield Improvement under Process Variation in Nanometer Technology”, Proc. DATE 2004, pp. 454 – 459.

[22]. A. Srivastava, et al., “Statistical Optimization of Leakage Power Considering Process Variations Using Dual-V_{th} and Sizing”, Proc. DAC 2004, pp. 773 – 778.

[23]. O. Neiroukh and X. Song, “Improving the Process-Variation Tolerance of Digital Circuits Using Gate Sizing and Statistical Techniques”, Proc. DATE 2005, pp. 294 – 299.

[24]. M. R. Guthaus, et al., “Gate Sizing Using Incremental Parameterized Statistical Timing Analysis”, Proc. ICCAD 2005, pp. 1029 – 1036.

[25]. K. Ghopra, et al., “Parametric Yield Maximization Using Gate Sizing Based on Efficient Statistical Power and Delay Gradient Computation”, Proc. ICCAD 2005, pp. 1023 – 1028.

[26]. S. Narenda, et al., “Full-Chip Sub-threshold Leakage Power Prediction and Reduction Techniques for Sub-0.18-µm CMOS, IEEE Journal of Solid-State Circuits, Volume 39, Feb. 2004, pp. 501 – 510.

[27]. R. Rao, et al., “Statistical Estimation of Leakage Current Considering Inter- and Intra-Die Process Variation”, Proc. ISLPED 2003, pp. 84 – 89.

[28]. R. Rao, et al., “Parametric Yield Estimation Considering Leakage Variability”, Proc. DAC 2004, pp. 442 – 447.

[29]. A. Srivastava, et al., “Accurate and Efficient Gate-Level Parametric Yield eEstimation Considering Correlated Variations in Leakage Power and Performance”, Proc. DAC 2005, pp. 535 – 540.

[30]. H. Chang and S. S. Sapatnekar, “Full-Chip Analysis of Leakage Power Under Process Variations, Including Spatial Correlations”, Proc. DAC 2005, pp. 523 – 528.

[31]. S. Zhang, et al., “A Probabilistic Framework to Estimate Full-Chip Sub-threshold Leakage Power Distribution Considering Within-Die and Die-to-Die P-T-V Variations”, Proc. ISPLED 2004, pp. 156 – 161.

[32]. H. Chang and S. S. Sapatnekar, “Statistical Timing Analysis Considering Partial Correlations Using a Single Pert-Like Traversal”, ICCAD 2003, pp. 621 – 625.

[33]. A. Agarwal, et al., “Statistical Timing Analysis for Intra-Die Process Variations with Spatial Correlations”, ICCAD 2003, pp. 900 – 907.

[34]. A. Agrawal, et al., “Statistical Timing Analysis Using Bounds and Selective Enumeration”, ICCAD 2003, pp. 1243 – 1260.

[35]. A. Devgan and C. Kashyap, “Block-based Static Timing Analysis with Uncertainty”, ICCAD 2003, pp. 607 – 614.

[36]. S. Bhardwaj, et. al, “τAU: Timing Analysis under Uncertainty”, ICCAD 2003, pp. 615 – 620.

[37]. K. Okada, et. al., “A Statistical Gate-Delay Model Considering Intra-Gate Variability”, ICCAD 2003, pp. 908 – 913

[38]. N. Tripathi, et. al., “Optimal Assignment of High Threshold Voltage for Synthesizing Dual Threshold CMOS Circuits”, VLSI Design 2001. pp. 227 – 32.

[39]. L. Wei, et. al., “Design and Optimization of Dual-Threshold Circuits for Low-Voltage Low-Power Applications”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol.. 7, 1999, pp. 16 – 24.

[40]. . Burns, et. al., “Design Optimization of a High-Performance Microprocessor Using Combination of Dual-Vt Allocation and Transistor Sizing”, VLSI Circuits Dig. Tech. Papers, 2002, pp. 218 – 219.

[41]. N. Sherwani, Algorithms for VLSI Physical Design Automation, 3rd edition, Kluwer Academic Publishers 1999.

[42]. Y. Cao, et.al., “New Paradigm of Predictive MOSFET and Interconnect Modeling for Early Circuit Simulation”, Proc. *CICC*, 2000*,* pp. 201 – 204.

[43]. Y. S. Dhillon, et. al., "Soft-Error Tolerance Analysis and Optimization of Nanometer Circuits", Proc. DATE 2005, pp. 288 – 293.

[44]. K. Roy and S. Parasad, Low Power CMOS VLSI: Circuit Design, 1st edition, Wiley-Interscience 2000.

[45]. J. Rabaey, et. al., Digital Integrated Circuits, 2nd edition, Prentice Hall 2002.

[46]. K. T. Cheng, et. al, "Robust Delay-Fault Test Generation and Synthesis for Testability under a Standard Scan Design Methodology", DAC 1991, pp. 80 – 86.

[47]. P. Agrawal, et. al, "Generating Tests for Delay Faults in Non-Scan", Design and Test of Computes, March 1993, pp. 20 – 28.

[48]. L. C. Chen, et. al, "High Quality Robust Tests for Path Delay Faults", VTS 1997, pp. 88.

[49]. A. K. Majhi and V. D. Agrawal, "Delay Fault Models and Coverage", VLSI Design Conference 1998, pp. 364 – 369.

[50]. J. Savir, "Broad-Side Delay Test", IEEE Tran. on CAD, Aug. 1994, pp. 284 – 290.

[51]. M. Abramovici, et. al, "Digital Systems Testing and Testable Design", AT&T 1990.

[52]. M. Ashouei, et. al., "Probabilistic Self-Adaptation of Nanoscale CMOS Circuits: Yield Maximization under Increased Intra-Die Variations", VLSI Design 2007, pp. 711 – 716.

[53]. P. E. Dodd and L. W. Massengill, "Basic Mechanisms and Modeling of Single-Event Upset in Digital Microelectronics", IEEE Trans. on Nuclear Science, Vol. 50, Issue 3, June 2003, pp. 583 – 602.

[54]. P. Shivakumar, et. al., "Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic", Proceeding International Conference on Dependable System and Networks 2002, pp. 389 – 398.

[55]. R. K. Iyer, et. al., "Recent Advances and New Avenues in Hardware-Level Reliability Support", IEEE MICRO, Vol. 25, No. 9, November 2005, pp. 18 – 29.

[56]. K. H. Huang and J. A. Abraham, "Algorithm-Based Fault Tolerance for Matrix Operations", IEEE Transactions on Computers", Vol. C-33, Issue 6, June 1984, pp. 518 – 528.

[57]. J. Y. Jou and J. A. Abraham, "Fault-Tolerant Matrix Arithmetic and Signal Processing on Highly Concurrent Computing Structures", Proceedings of the IEEE, vol. 74, No. 5, May 1986, pp. 732 – 741.

[58]. J. Y. Jou and J. A. Abraham, "Fault Tolerant FFT Networks", IEEE Transactions on Computers, Vol. 37, No. 5, May 1988, pp. 548 – 561.

[59]. V. S. Nair and J. A. Abraham, “Real-Number Codes for Fault-Tolerant Matrix Operations on Processor Arrays”, IEEE Trans. on Computer, Vol.39, Issue 4, April 1990, pp. 426 – 435.

[60]. L. N. Reddy and P. Banerjee, “Algorithm-Based Fault Detection for Signal Processing Applications”, IEEE Transactions on Computers, Vol. 39, No. 10, October 1990, pp. 1304 – 1308.

[61]. A. Chatterjee and M. A. d’Abreu, “The Design of Fault Tolerant Linear Digital State Variable System: Theory and Technique”, IEEE Trans. on Computers, Vol. 42, No. 7, July 1993, pp. 794 – 808.

[62]. A. Chatterjee and R. K. Roy, “Concurrent Error Detection in Nonlinear Digital Circuits Using Time-Freeze Linearization”, IEEE Transactions on Computers, Vol. 46, No. 11, December 1997, pp. 1208 – 1218.

[63]. F. T. Luk and H. Park, “An Analysis of Algorithm-Based Fault Tolerance Techniques”, Journal of Parallel and Distributed Computing, Vol. 5, Issue 2, April 1988, pp. 172 – 184.

[64]. G. M. Megson and D. J. Davis, “Algorithmic Fault Tolerance for Matrix Operations on Triangular Arrays”, Journal of Parallel Computing, Vol. 10, No. 2, April 1989, pp. 207 – 219.

[65]. W. W. Peterson and E. J. Weldon, “Error Correcting Codes”, MIT Press, Cambridge. Mass, 1972.

[66]. D. G. Hoffman, et al., Coding Theory: The Essentials, Marcel Dekker Inc, N.Y. 1991.

[67]. R. Blahut, Algebraic Codes for Data Transmission, MIT Press, Cambridge, Mass, 2003.

[68]. B. W. Johnson, “Design and Analysis of Fault Tolerant Digital System”, Addison-Wesley 1989.

[69]. A. U. Diril, et. al., "Design of Adaptive Nanometer Digital Systems for Effective Control of Soft Error Tolerance", Proceeding of VTS 2005, pp. 298 – 303.

[70]. M. Nicolaidis, “Time Redundancy Based Soft Error Tolerance to Rescue Nanometer Technologies”, Proceeding of VTS 1999, pp. 86 – 94.

[71]. S. Mitra, et. al., “Robust System Design Built-In Soft-Error Resilience”, Computer, Vol. 38, No. 2, Feb. 2005, pp. 43 – 52.

[72]. Y. Arima, et. al, “Cosmic-Ray Immune Latch Circuit for 90nm Technology and Beyond”, Proc. SSCC 2004, pp. 492 – 493.

[73]. R. Hedge and N. R. Shanbhag, “Soft Digital Signal Processing”, IEEE Tran. On VLSI, Vol. 9, Issue 6, Dec 2001, pp. 813 – 823.

[74]. B. Shim and N. R. Shanbhag, “Reduced Precision Redundancy for Low-Power Digital Filtering”, In Proc. of Asilmar Conerence 2001, pp. 148 – 152.

[75]. N. R. Shanbhag, “Reliable and Energy-Efficient Digital Signal Processing”, DAC 2002, pp. 830 – 835.

[76]. B. Shim and N. R. Shanbhag, “Energy-Efficient Soft Error-Tolerant Digital Signal Processing”, IEEE Tran. On VLSI, Vol. 14, No. 4, April 2006, pp. 336 – 348.

[77]. P. Denyer and D. Renshaw, “VLSI Signal Processing: A Bit-Serial Approach”, Addison-Wesley, 1985.

[78]. R. I. Hartley and J. R. Jasica, “Behavioral to structural transformation in a bit-serial silicon compiler”, IEEE Transactions on Computer-Aided Design, Vol. 7, No. 8, August 1988, pp. 877-886.

[79]. N. Park and A. Parker, “Sehwa: A Software Package for Synthesis of Pipeline from Behavioral Specifications”, IEEE Transactions on Computer-Aided Design, Vol. 7, Issue 3, March 1988, pp. 356 – 370.

[80]. G. Zelniker and F. J. Taylor, “Advanced Digital Signal Processing: Theory and Applications”, Marcel Dekker, Inc., 1994.

[81]. J. C. Lagarias, et al., "Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions", SIAM Journal of Optimization, Vol. 9, No. 1, 1998, pp. 112 – 147.

[82]. http://www.mathworks.com/access/helpdesk/help/techdoc/index.html, fminsearch help (August 2007)

[83]. M. Nissar, et al., “Probabilistic Concurrent Error Compensation in Nonlinear Digital Filters Using Linearized Checksums”, to appear in IOLTS 2007, pp. 173 – 182.

[84]. M. Ashouei, et al., “Reconfiguring CMOS as Pseudo N/PMOS for Defect Tolerance in Nano-Scale CMOS”, VLSI Design 2008, *submitted.*