National Technical University of Athens
School of Electrical & Computer Engineering
Department of Computer Science

KU LEUVEN

Katholieke Universiteit Leuven
Faculty of Engineering Science
Arenberg Doctoral School

Performance Variation in Digital Systems: Workload Dependent Modeling and Mitigation

Michail Noltsis

Supervisors:
Prof. Dimitrios Soudris (NTUA)
Prof. Francky Catthoor (KUL)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

September 2020

Performance Variation in Digital Systems: Workload Dependent Modeling and Mitigation

Michail NOLTSIS

Examination committee:

Prof. Dimitrios Soudris (NTUA), supervisor
Prof. Francky Catthoor (KUL), supervisor
Prof. Kiamal Pekmestzi (NTUA)
Dr. Alessio Spessot (imec)
Prof. Georges Gielen (KUL)
Prof. Yanos Sazeides (UCY)
Prof. Dimitrios Gizopoulos (UOA)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

September 2020

This work is partially supported by the European Commission FP7 612069-HARPA project.

Michail Noltsis, 25 Kalavriton Street, Zografou, 15773 Athens, Greece
mnoltsis@microlab.ntua.gr

ΕΘΝΙΚΌ ΜΕΤΣΌΒΙΟ ΠΟΛΥΤΕΧΝΕΊΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ
ΥΠΟΛΟΓΙΣΤΩΝ

Performance Variation in Digital Systems: Workload Dependent Modeling and Mitigation

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Μιχαήλ Α. Νόλτσης

Συμβουλευτική Επιτροπή : **Δημήτριος Σούντρης**
Francky Catthoor
Κιαμάλ Πεκμεστζή

Εγκρίθηκε από την επταμελή εξεταστική επιτροπή την 31η Ιούνιου 2020

.....	.....	.....
Δημήτριος Σούντρης	Francky Catthoor	Μιχαήλ Ψαράκης
Καθηγητής ΕΜΠ	Professor KUL	Επ. Καθηγητής ΠΑΠΕΙ
.....	.....	.....
Georges Gielen	Alessio Spessot	Δημήτριος Γκιζόπουλος
Professor KUL	Program Manager imec	Καθηγητής ΕΚΠΑ
	.....	
	Γιαννάκης Σαζείδης	
	Professor UCY	

Αθήνα, Σεπτέμβριος 2020

.....

Μιχαήλ, Α. Νόλτσης

Διδάκτωρ Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Acknowledgments

The current text is the result of my PhD studies. The academic process has taken place in two universities: the National Technical University of Athens, Greece and Katholieke Universiteit Leuven, Belgium. This attempt, while time-consuming and difficult has also been very interesting. I hereby would like to acknowledge the names of certain people whose help allowed the successful completion of my studies and the writing of this dissertation.

Firstly, I would like to express my sincere gratitude to my supervisor Prof. Dimitrios Soudris. His guidance, support and related research have been of major importance during the course of my PhD. Furthermore, with tons of patience and his assistance and understanding, I have managed to navigate through the unfriendly environment of the Greek university. I would also like to thank my co-supervisor Prof. Francky Catthoor from the Katholieke Universiteit Leuven. His mentoring and immense knowledge have been key enablers for the fulfillment of my PhD work.

Besides my supervisors, I would like to thank the rest of my thesis committee: Prof. Kiamal Pekmestzi and Alessio Spessot for their insightful comments and encouragement, but also for the hard questions which provided me with an incentive to widen my research from various perspectives.

I thank my fellow lab mates, Dimitris and Nikos, for enlightening me the first glance of research. Moreover, I would like to thank certain students I had the pleasure to work with, particularly Eleni and Nikos; I am very proud of their progress. Big ups also to Nikitas and Orestis who have been my friends since the undergraduate years.

Last but not the least, I would like to thank my family: my parents and my brother for supporting me spiritually throughout my academic studies and my life in general.

Extended Abstract

The computer industry is witnessing an unprecedented demand for more functionality and performance and is continuously using silicon components with smaller form factor and feature size. This aggressive downscaling of hardware components has brought about several failure mechanisms that degrade the system's operation, threatening its dependability. Such failure mechanisms can be the result of the natural occurring variation in the attributes of circuit elements during the fabrication procedure, or can be attributed to the aging and the gradual wearout of the hardware and other variability effects related to space particles and power/ground line voltage variation. The inherent stochastic nature of these failure phenomena contributes to the so-called performance variation of digital systems, in the sense that system behavior and response cannot be fully deterministic and have a dynamic component.

In the software layer, computational- and data-intensive applications, user interaction and quality of service conditions also generate persistently varying and unpredictable workloads, deteriorating this effect. While software applications are becoming even more complex and resource-hungry (especially due to the continued "virtualization" that leads to the ubiquitous use of run-time threads and dynamic memory allocation) and since the shrinking of transistor and interconnect dimensions is not expected to end in this decade, we can assume that we have already entered an era of inevitable, strongly dynamic performance variation.

Under this highly dynamic context of system operation, ensuring dependability and meeting timing constraints seems challenging. The goal of the current research is to study existing methodologies that mitigate performance variation and develop related schemes that can ultimately ensure and guarantee timing deadlines. For this reason, we first briefly discuss the dominant failure mechanisms which create defects is the silicon layer and deteriorate the reliability of the system. A thorough review of the prior art on the subject of reliability mitigation is also important in order to realize the current, state-of-the-art

mitigation approaches and methodologies. We specifically explore the research domain of parametric reliability, namely the mitigation techniques that protect the system from extreme fluctuations of operation parameters, especially regarding the timing aspects.

Then, the aforementioned reliability threats need to be captured and modeled while their impact on the system's performance should be described and estimated. Hence, we employ existing tools, and suggest new ones, in order to develop a complete framework that effectively evaluates the failure probability of electronic components, focusing especially on the SRAM buffers of NoC routers. While the reliability community widely uses the time-consuming Monte-Carlo experiments to asses this metric, our framework is founded on the analytics-based MPFP methodology, which we have studied and improved for the case of typical 6T memory cells.

We later present a realistic case study of a closed-loop PID controller that mitigates performance variation with a reactive DVFS response. This concept has been discussed but was only illustrated on small benchmarks; in particular, the integration of the approach on an embedded platform and the extension to manage dynamic workloads has not been shown earlier. This scheme is compared against the version of a Linux CPU frequency governor in terms of energy consumption and timing response. Moreover, we move forward and suggest another flavor of this scheme to perform thermal management. Again this controller is implemented on pure hardware and illustrated with a realistic case study.

Next, the aforementioned PID controller is improved to operate on finer granularity, at the thread node level. The concepts of performance and deadline vulnerability factor are introduced to support the formulation of a discrete time control problem while the basis of this new approach utilizes the system scenario methodology; this methodology, along with related terms and definitions, is studied in detail. In addition, a run-time adjustment on this methodology to adapt to performance variability norms is shown, creating an adaptive scenarios scheme and achieving notable energy gains. Still, however, performance variation is managed with a reactive response and no timing guarantees are yet delivered.

Finally, we propose proactive DVFS actuations on the thread node level using dynamic scenarios to guarantee timelines in a cost-efficient manner. By exploiting the partial predictability of the application behavior, we develop a dynamic scenario approach and enable cost-effective DVFS decisions. Simulation results present significant energy gains compared to previous frequency guardband methods while experimental results on the hardware platform substantiate the effectiveness of our scheme.

Εκτεταμένη Περίληψη

Η βιομηχανία υπολογιστών βιώνει ολοένα κι αυξανόμενη (ανευ προηγουμένου) ζήτηση για αυξημένη λειτουργικότητα και επίδοση, χρησιμοποιώντας συνεχώς πυριτίο με όλο και μικρότερα μεγέθη χαρακτηριστικών και συντελεστή μορφής. Η επιθετική σμίκρυνση των συνιστωσών υλικού οδήγησε αναπόφευκτα σε νέους μηχανισμούς σφαλμάτων, τα οποία αποτελούν απειλή για την αξιοπιστία και την εύρυθμη λειτουργία του συστήματος. Τα σφάλματα αυτά μπορεί να οφείλονται στις εύλογες διακυμάνσεις των χαρακτηριστικών και διαστάσεων των στοιχείων του κυκλώματος κατά την κατασκευή, ή να αποδίδονται στη γήρανση και τη σταδιακή φθορά του υλικού. Η εγγενώς στοχαστική φύση αυτών των μηχανισμών είναι η γενεσιουργός αιτία της αποκαλούμενης 'διακύμανσης επίδοσης' των ψηφιακών συστημάτων, υπό την έννοια ότι η συμπεριφορά κι απόκριση του συστήματος δεν μπορεί να είναι απόλυτα ντετερμινιστική αλλά ενέχει και μια δυναμική συνιστώσα.

Σε επίπεδο λογισμικού, η ύπαρξη εφαρμογών υπολογιστικής έντασης αλλά και έντασης δεδομένων, η επίδραση του χρήστη αλλά και η ποιότητα της υπηρεσίας συμβάλλουν εξίσου στη συνεχή ύπαρξη κυμαινόμενου και απρόβλεπτου υπολογιστικού φόρτου, επιδεινώνοντας περαιτέρω την επίδοση. Με τις εφαρμογές λογισμικού να γίνονται όλο και πιο πολύπλοκες και υπολογιστικά απαιτητικές σε επίπεδο πόρων, ειδικά λόγω της αυξανόμενης 'εικονικοποίησης' που οδηγεί σε απανταχού χρήση νημάτων εκτέλεσης και δυναμική εκχώρηση μνήμης και δεδομένου ότι η σμίκρυνση των τρανζίστορ και των διαστάσεων συνδεσμολογίας δεν αναμένεται να φτάσει στο τέλος της εντός της τρέχουσας δεκαετίας, μπορεί με ασφάλεια να υποτεθεί ότι θα βιώσουμε μια εποχή αναπόδραστης διακύμανσης επίδοσης έντονα δυναμικού χαρακτήρα.

Στα πλαίσια του δυναμικού αυτού περιβάλλοντος λειτουργίας του συστήματος, η διασφάλιση της αξιοπιστίας και η πλήρωση των χρονικών περιορισμών φαντάζει δύσκολη πρόκληση. Στόχος της παρούσας εργασίας είναι η μελέτη υπαρχουσών μεθοδολογιών για τη μείωση της διακύμανσης επίδοσης και η ανάπτυξη σχετικών μεθόδων που μπορούν να διασφαλίσουν και να εγγυηθούν τους χρονικούς περιορισμούς. Στα πλαίσια αυτά, αρχικά θα παρουσιαστούν εν συντομία οι

κυρίαρχοι μηχανισμοί σφαλμάτων που ευθύνονται για αστοχίες σε επίπεδο πυριτίου και επιδεινώνουν την αξιοπιστία του συστήματος. Η ενδελεχής μελέτη της υπάρχουσας βιβλιογραφίας στον τομέα της αξιοπιστίας συστημάτων είναι σημαντική και για την εμπέδωση των σύγχρονων προσεγγίσεων και μεθόδων ελέγχου της αξιοπιστίας. Πιο συγκεκριμένα, θα μελετηθεί το ερευνητικό πεδίο της παραμετρικής αξιοπιστίας, δηλαδή οι τεχνικές προστασίας του συστήματος από ακραίες διακυμάνσεις των παραμέτρων λειτουργίας, ειδικά σε σχέση με τις χρονικές απαιτήσεις.

Κατόπιν, οι προαναφερθείσες απειλές αξιοπιστίας πρέπει να εντοπισθούν και να μοντελοποιηθούν, ενώ η επίδρασή τους στην επίδοση του συστήματος πρέπει να εξηγηθεί και να εκτιμηθεί. Για το σκοπό αυτό θα χρησιμοποιηθούν υπάρχοντα εργαλεία αλλά και θα προταθούν καινούρια, ώστε να αναπτυχθεί ένα πλήρες πλαίσιο αποτελεσματικής αξιολόγησης της πιθανότητας αστοχίας ηλεκτρικών συνιστωσών, επικεντρώνοντας περαιτέρω στα κύταρρα μνήμης δρομολογητών σε ηλεκτρονικες πλατφορμες. Παρόλο που η ερευνητική κοινότητα χρησιμοποιεί ευρέως το χρονοβόρο μοντέλο Monte-Carlo για την αποτίμηση αυτής της πιθανότητας, το προτεινόμενο πλαίσιο βασίζεται στην αναλυτική μεθοδολογία MPFP που μελετήθηκε και βελτιώθηκε για την υπόθεση τυπικών 6T στοιχείων κυττάρων μνήμης.

Στη συνέχεια θα παρουσιαστεί ένα ρεαλιστικό σενάριο ενός PID ελεγκτή κλειστού βρόχου για τον περιορισμό της διακύμανσης επίδοσης μέσω μιας αντιδραστικής απόκρισης DVFS. Η ιδέα αυτή έχει μελετηθεί αλλά μόνο σε μικρή κλίμακα πιο συγκεκριμένα, η ενσωμάτωση της σε μια ολοκληρωμένη πλατφόρμα και η επέκτασή της για τη διαχείριση δυναμικού φόρτου εργασίας δεν έχει εξετασθεί. Το μοντέλο αυτό θα συγκριθεί ως προς κατανάλωση ενέργειας και χρονική απόκριση με το μοντέλο βάσης ενός ελεγκτή συχνότητας του λειτουργικού Linux. Επιπλέον θα προταθεί μια διαφοροποιημένη εκδοχή του μοντέλου για τη διαχείριση θερμότητας. Ο ελεγκτής αυτός υλοποιείται και πάλι με πραγματικό υλικό και παρουσιάζεται με ένα ρεαλιστικό σενάριο.

Εν συνεχεία, ο προαναφερθείς ελεγκτής βελτιώνεται ώστε να λειτουργήσει με αυξημένο βαθμό λεπτομέρειας σε επίπεδο thread node level. Εισάγονται οι έννοιες του συντελεστή ευπροσβλητότητας επίδοσης και ευπροσβλητότητας χρονικού ορίου ώστε να στοιχειοθετηθεί ένας πρόβλημα ελέγχου διακριτού χρόνου, ενώ η νέα αυτή προσέγγιση χρησιμοποιεί ως βάση τη μεθοδολογία σεναρίων συστήματος. Η μεθοδολογία αυτή, αλλά και οι σχετικοί όροι και ορισμοί, μελετώνται λεπτομερώς. Επιπρόσθετα, παρουσιάζεται μια προσαρμογή στο περιβάλλον εκτέλεσης της μεθοδολογίας ώστε να ληφθούν υπόψη μοτίβα διακύμανσης επίδοσης, δημιουργώντας ένα πλαίσιο ευπροσάρμοστων σεναρίων και σημαντικά ενεργειακά οφέλη. Σε αυτό το σημείο, η διακύμανση επίδοσης εξακολουθεί να αντιμετωπίζεται με αντιδραστική απόκριση, χωρίς να παρέχονται χρονικές εγγυήσεις.

Τελικώς, προτείνονται προληπτικοί χειρισμοί DVFS σε επίπεδο thread node, με χρήση δυναμικών σεναρίων για χρονικές εγγυήσεις με οικονομικά αποδοτικό τρόπο. Αξιοποιώντας τη μερική προβλεψιμότητα της συμπεριφοράς κάθε εφαρμογής, αναπτύσσεται μια προσέγγιση δυναμικών σεναρίων και καθίσταται εφικτή η λήψη οικονομικά αποδοτικών αποφάσεων DVFS. Τα αποτελέσματα της προσομοίωσης επιτυγχάνουν σημαντικά ενεργειακά οφέλη σε σχέση με προηγούμενες μεθόδους ελέγχου συχνότητας, ενώ τα πειραματικά αποτελέσματα της υλοποιηθείσας διάταξης καταδεικνύουν την αποτελεσματικότητα του προτεινόμενου μοντέλου.

Uitgebreide Samenvatting

De computerindustrie is getuige van een ongekende vraag naar meer functionaliteit en prestaties en gebruikt continu siliciumcomponenten met een kleinere vormfactor en dimensies. Deze agressieve verkleining van hardwarecomponenten heeft verschillende faalmechanismen veroorzaakt die de werking van het systeem verslechteren en de betrouwbaarheid ervan in gevaar brengen. Dergelijke faalmechanismen zijn ofwel het gevolg van de natuurlijk voorkomende variatie in de eigenschappen van circuitelementen tijdens de fabricageprocedure, of zij kunnen worden toegeschreven aan de geleidelijke veroudering van de hardware en andere variabiliteitseffecten die verband houden met de componenten zelf en de variatie in de spanning van de voeding of de grond. De inherente stochastische aard van deze faalmechanismes draagt bij aan de zogenaamde prestatievariatie van digitale systemen, in die zin dat systeemgedrag en -respons niet volledig deterministisch kunnen zijn en een significante dynamische component vertonen.

In de softwarelaag van de applicaties, genereren dynamische computationele en data-intensieve applicaties, gebruikersinteracties en wisselende quality of service-voorwaarden ook voortdurend een variërende en onvoorspelbare platformbelasting, waardoor dit effect nog verslechtert. Terwijl softwaretoepassingen nog complexer worden en meer middelen nodig hebben op het platform (vooral vanwege de voortdurende " virtualisatie " die leidt tot het alomtegenwoordige gebruik van event-gedreven taken en dynamische geheugentoewijzing) en sinds de inkrimping van transistor- en interconnectdimensies naar verwachting nog zal verdergaan in dit decennium, kunnen we aannemen dat we al een tijdperk van onvermijdelijke, sterk dynamische prestatievariaties zijn binnengegaan.

Onder deze zeer dynamische context van systeemuitvoering blijkt het een uitdaging om betrouwbaarheid te garanderen en te voldoen aan timingbeperkingen. Het doel van het huidige onderzoek is om bestaande methodologieën te bestuderen die prestatievariaties verminderen en gerelateerde schema's te ontwikkelen die uiteindelijk deze randvoorwaarden kunnen garanderen. Om deze reden bespreken we eerst kort de dominante faalmechanismen die

defecten veroorzaken in de siliciumlaag en dus een negatieve impact hebben op de betrouwbaarheid van het systeem. Een grondige herziening van de stand van de techniek op het gebied van beperking van betrouwbaarheid is ook belangrijk om de huidige mitigatiebenaderingen en -methodologieën te realiseren. We onderzoeken specifiek het onderzoeksdomein van parametrische betrouwbaarheid, namelijk de mitigatietechnieken die het systeem beschermen tegen extreme schommelingen van uitvoeringsparameters, vooral met betrekking tot de timingaspecten.

Vervolgens moeten de bovengenoemde betrouwbaarheidsbedreigingen worden vastgelegd en gemodelleerd, terwijl hun impact op de systeemprestaties nauwkeurig moet worden beschreven en geschat. Daarom gebruiken we bestaande CAD omgevingen als fundament en stellen nieuwe uitbreidingen voor om een compleet raamwerk te ontwikkelen dat de faalkans van elektronische componenten effectief evalueert, met speciale aandacht voor de SRAM-buffers van NoC-routers. De huidige betrouwbaarheidsgemeenschap maakt op grote schaal gebruik van tijdrovende Monte-Carlo-experimenten om deze statistiek te beoordelen. In tegenstelling daartoe, is ons raamwerk gebaseerd op de op analyse-gebaseerde MPFP-methodologie, die we hebben bestudeerd en verbeterd voor het geval van typische 6T-SRAM geheugencellen.

We presenteren daarna een realistische demonstratie van een gesloten-lus PID-controlesysteem dat de prestatievariatie vermindert door middel van een reactieve DVFS-sturing. Dit concept is besproken, maar is alleen geïllustreerd op kleine applicaties; met name de integratie van de aanpak op een iingebed platform en de uitbreiding om dynamische workloads te beheren is niet eerder getoond. Deze regeling wordt vergeleken met de versie van een Linux CPU frequentieregelaar in termen van energieverbruik en timingreactie. Bovendien gaan we verder en suggereren we een andere variant van dit schema om thermisch beheer uit te voeren. Wederom is deze controller geïmplementeerd op pure hardware en geïllustreerd met een realistische demonstratie.

Vervolgens is de eerder genoemde PID-regelaar verbeterd om te werken met een fijnere granulariteit, op het niveau van de zogenaamde “thread-node”. De prestaties en timing-kwetsbaarheidsfactor worden geïntroduceerd om de formulering van een discreet tijdcontroleprobleem te ondersteunen. De basis van deze nieuwe benadering maakt gebruik van de systeemscenario-methodologie. Deze methodologie, samen met gerelateerde termen en definities, wordt in detail bestudeerd. Daarnaast wordt een aanpassing op deze methodologie getoond om zich tijdens de uitvoering zelf nog verder aan te passen aan prestatievariabiliteitsnormen. Hierdoor wordt een adaptief scenarioschema gecreëerd en opmerkelijke resultaten worden behaald naar energiewinst toe. Toch worden prestatievariaties nog steeds beheerd met een reactieve respons en worden nog geen timinggaranties gegeven.

Ten slotte stellen we proactieve DVFS-activeringen voor met een tussenliggende granulariteit in de taken die uitgevoerd worden. Dit wordt bekomen met dynamische scenario's om de tijdsbeperkingen op een kostenefficiënte manier te garanderen. Door de gedeeltelijke voorspelbaarheid en de correlaties van het applicatiegedrag te benutten, ontwikkelen we een dynamische scenariobenadering en maken kosteneffectieve DVFS-beslissingen mogelijk. Simulatieresultaten tonen aan dat we aanzienlijke energiewinsten kunnen aanbieden in vergelijking met eerdere methoden, terwijl experimentele resultaten op een representatief hardwareplatform de effectiviteit van onze regeling verder ondersteunen.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this Chapter, we focus on the concept of performance variation in digital systems and discuss its sources, mainly the interdependence with certain phenomena manifesting in the hardware, like process variation and aging. First, we will briefly present the necessary terminology regarding system reliability. Next, we will explain how reliability threats, generated as defects and imperfections on the hardware layer can propagate through the system, architecture and software layers and cause binary errors and performance variation. Then, we will shortly discuss the current approaches adopted to mitigate performance variation. Finally, an overview of the contributions of our text will complete the Chapter.

1.1 Performance Variation in Digital Systems: Context and Motivation

System usage in modern electronic devices is not stable; on the contrary, digital systems now operate under a highly dynamic context as a result of uncertainties and numerous conditional elements in the hardware and software layers. In the latter, input data, user interaction and quality of service (QoS) requirements result in a constantly varying taskset of applications, algorithms and their workloads. In fact, counter to traditional applications such as the email and accounting software, today's applications are more dynamic and unpredictable. Prominent examples of modern workloads can be drawn from middleware and operating systems (OS) which are becoming increasingly complicated since numerous hardware features are being added to enable the use of runtime

libraries, interrupts, and parallel processing. The trend for more virtualization in relevant multimedia and communication applications, along with their complex stimuli and multiple QoS demands also generates series of workloads with notable dynamic components. In addition, certain real-time applications, like online gaming, video conferences and some e-commerce transactions, access databases and function within strict time frames that the user senses as immediate and provide real-time information that changes frequently. These nonstop changes in the workload of modern applications contribute to creating dynamic operation and thus, performance variation of digital systems.

In the logic devices of the hardware, primary sources of performance variation are reliability threats such as radiation striking, and aging. Process variability during the fabrication procedure, such as the fluctuations in the implanted impurity concentration or in the oxide thickness, alter transistors' characteristics, especially threshold voltage [250]. Certain device parameters can also be affected by temperature; carrier mobility and velocity saturation, for instance, are dropping with the rise of temperature. These parameters affect metal–oxide–semiconductor field-effect transistor (MOSFET) operation therefore, overall system performance is directly correlated to the system's thermal profile. On top of that, aging effects on the oxide of the MOSFET or the interconnects are exacerbated by temperature and can provoke defects in the physical layer that deteriorate system operation while alpha radiation and neutron strikes can damage electronic components (especially memories) and generate soft/transient errors. Finally, supply voltage variations are also a growing source of dynamic variations in deeply scaled nodes.

To ensure correct bit-wise operation and mitigate the aforementioned phenomena, computer engineers have introduced a diverse arsenal of countermeasures such as error-correcting codes, cyclic redundancy checks, shadow latches e.t.c. These routines usually come at a variety of costs, namely power (e.g. voltage binning [305, 248]), area (shadow latches [82]) or timing. Timing penalties in particular increase the execution time and introduce performance variation. Instruction-level rollback techniques [236, 43], for instance, ensure the correct functionality of the system by saving application data of several checkpoints into memory buffers. After the detection of an error, system is rolled-back to the last correct checkpoint and operation is repeated. This type of imposed performance variation threatens timely execution and is a bottleneck for applications with deadline constraints.

The hypothesis of eliminating reliability threats and eradicating aforementioned sources of dynamism is unrealistic to preserve not only for today's technology but also for future chips. First of all, process variation and hardware defects are expected to become more intense as the transistor miniaturization continues. As technology nodes shrink, the amount of process variability becomes notably

prominent since, in this case, variability affects a larger percentage of the device proportions (length or width) and feature sizes reach fundamental dimensions such as the light's wavelength in optical lithography [62, 145] and the size of atoms [145]. Secondly, extra capabilities and additional functions are continuously being added in modern processors while the applications tend to be more varying in time. Both the physical components (hardware) and the software of digital systems are expected to work under an intense dynamic fashion. We are entering, therefore, an era of inevitable performance variation and the need to describe and understand its sources, mitigate its impact and finally ensure system timely operation is crucial. While the source of performance variation can be traced on both the software and hardware, in our work we mainly focus on the latter and study how reliability phenomena trigger performance variation. Specifically, we will explain the physical mechanisms, creating the phenomena while briefly presenting the premises on reliability study. We will also discuss methodologies and develop relevant software tools in an attempt to estimate their impact on system performance; various other approaches and aspects of prior art will be thoroughly examined as well. Finally, we will propose certain frameworks that attempt to mitigate the variation effects, manage performance and guarantee timing constraints in a cost-effective manner. The latter is often neglected since many techniques adopt a design-time, worst-case approach that uses large safety margins and is proven quite suboptimal.

1.2 Reliability of Integrated Circuits

1.2.1 Premises and Terminology

In general, *reliability*, *availability*, and *serviceability/maintainability* (RAS) are features that all together describe the *dependability* of an electronic product; the ability, that is, of a system to provide trusted services within a time period[1]. In fact, RAS acronym was first introduced by International Business Machines (IBM) to describe the robustness of their computers in maintaining their availability and data integrity [253].

Definition 1. *Reliability* is the probability that an electronic system will perform correct service during its operation [20].

[1] Over time, other characteristics of dependability have also been mentioned such as *safety* and *integrity*. *Safety* is the probability that a system will either function correctly or "fail" in a safe manner that does not cause catastrophic consequences on other related systems, user(s) and the environment. *Integrity* translates to the absence of improper system alterations during a period of operation [20].

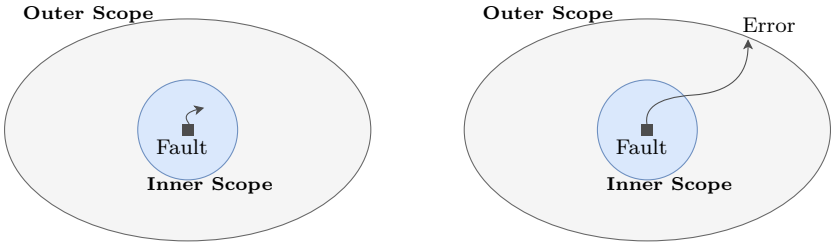


Figure 1.1: Illustration of faults and errors: a) a fault in the inner scope that is masked or corrected, is not propagating to the outer scope; b) a fault in the inner scope, propagated and manifesting as an error in the outer scope

Definition 2. *Availability* refers to the probability of an electronic system being available and ready for correct service for a given period of time [20].

Definition 3. *Serviceability* (or *maintainability*) is the probability that a failed system can go through repairs and modifications so that its functioning state is restored under a specific time period [20].

We have already highlighted how the reliability phenomena, are a dominant reason for the dynamic operation of digital systems. These phenomena typically arise from imperfections and defects on the silicon layer yet they can also be initiated from environmental conditions and interactions. These mechanisms can potentially cause *faults* in an electronic system, like for example a defective transistor due to oxide wearout or an open circuit because of intense electromigration[2]. Engineers classify faults in three main categories:

- *permanent*, describing faults that remain throughout system lifetime if repairing procedure does not occur;
- *semi-permanent* (or *intermittent*), referring to faults that persist for long periods and are likely to recur;
- *transient*, a term used to characterize faults that instantly occur before they disappear such as a a bit-flip in a memory cell caused by subatomic particle strikes coming from cosmic rays.

The manifestation of a fault is called an *error* [185]. Faults are the source of errors in digital systems however, not all faults necessarily cause an error. A

[2]Apart from faults in the process technology and circuit layers, they can also appear in the software as well, like for example software bugs in the application or the operating system.

fault in one of the inputs in a NAND gate, for instance, is masked, should another input of the gate be at the low logic level. Faults can also be tolerated through specific circuit and architecture designs. When a fault surfaces in a particular *scope*, it is called an error. If faults are either masked or tolerated in a given scope, they do not propagate outside hence, do not manifest as errors. Therefore the terms *fault* and *error* are directly correlated to the specific *scope* we are examining. An illustration of the above is presented in Figure 1.1, drawn from relevant work [185]. An error that is visible to the end-user is often referred as a *failure* thus, failures are a subgroup of errors propagating at the outer scope of the overall system. Similarly to faults, errors can also be classified as permanent, semi-permanent and transient.

1.2.2 Functional and Parametric Reliability

A variety of mechanisms can cause faults in the silicon of digital systems, ranging from process variability and manufacturing defects – such as contaminant particles deposited on the silicon devices – to radiation, gate-oxide degradation, interconnect wearout and more. If the fault manifests as an error, it can provoke either a *functional* or a *parametric reliability violation.*

Definition 4. *Functional reliability violation* is the event of a divergence from correct system behavior with regards to the binary digits that are communicated, processed and stored [228, 218].

Definition 5. *Parametric reliability violation* refers to the fluctuation of certain system's operational characteristics outside their predefined boundaries [228, 218]. These characteristics may include temperature, energy consumption, delay etc.

A functional violation for instance corresponds to a bit flip in a memory cell – turning a 0 into a 1 or vice versa – triggered by cosmic ray striking this component. Another example is the erroneous output of a P-type metal-oxide-semiconductor (pMOS) transistor suffering from Negative Bias Temperature Instability (NBTI) damage [104]. A parametric violation, contrariwise, relates to a system's parameter fluctuating outside of set margins; in the latter example in particular, NBTI slows down the transistor and should its delay exceed acceptable levels, it is rendered non-functional even if the output of the device is

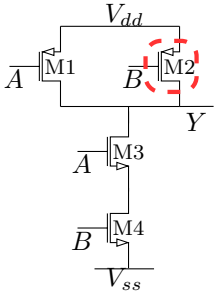
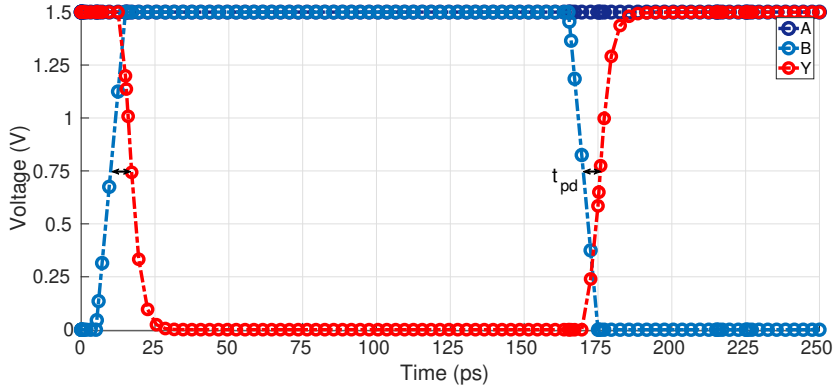
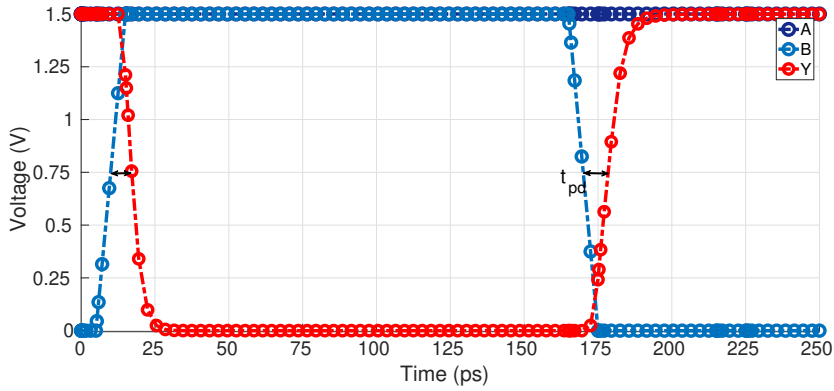


Figure 1.2: Schematic diagram of a NAND2 gate.

correct. Next, we attempt to clarify all the above by showing the impact a degraded transistor (M2) can have on the delay of a simple NAND gate (see Figure 1.2), implemented on the `NGSPICE` simulator, using PTM [50] at 90nm nodes. We adopt the definition of the *propagation delay* (t_{pd}) which translates to "the maximum time needed from the input crossing 50% to the output crossing 50%" [292]. In the absence of any degradation in the device, propagation delay is presented in Graph 1.3a. When NBTI damages M2, this delay rises as one can observe in Graph 1.3b. Notice that in both cases, input A does not change.



(a)



(b)

Figure 1.3: Propagation delay for the NAND2 gate: (a) Transient response when the M2 transistor does not suffer from NBTI; propagation delay is minimum. (b) In this instance, M2 is damaged by NBTI and the delay increases significantly.

A functional violation is strictly related to binary correctness; in our example, such a violation can happen if a wrong output is produced, regardless the gate's timing delay. When, however, the propagation delay is estimated to exceed a set margin, a parametric violation occurs, even if the output value is correct bit-wise. An exploration of the gate's delay versus the degradation of M2 device is shown in Figure 1.4. Assuming that an upper threshold for t_{pd} is set at $3.8ps$ by the designer, we can verify how a parametric violation could occur in the scenario when the ΔV_{th} of the transistor due to NBTI stress rises to (or goes beyond) $75mV$. It should be noted that such levels of threshold voltage shifts are not uncommon for devices after yearly operation.

It should also be stressed that even though the Definitions 4 and 5 appear complementary, they are significantly interdependent. On the one hand, a timing violation in a logic gate that is part of a sequential circuit could lead to a mismatch with the clock frequency of the latches and therefore a functional violation. Another example would be a parametric reliability violation concerning the temperature of the system's hardware. Subthreshold currents have an exponential dependence with temperature and since strong leakage currents can change a Dynamic Random-Access Memory (DRAM) cell's data (the capacitor is gradually discharged through the subthreshold current of the OFF transistor), the parametric violation can provoke a functional one.

On the other hand, circuit designers and computer architects have advised ways to tackle with functional violations at the system-level scope and prevent errors visible to the user by adding extra hardware or by imposing overheads

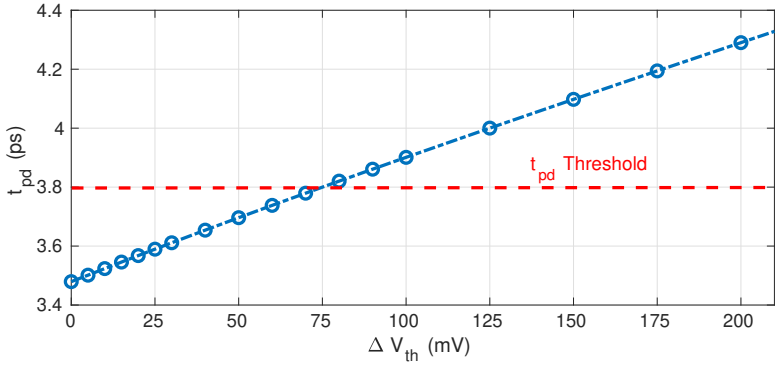


Figure 1.4: Graph showing the relation between the ΔV_{th} of M2 because of NBTI and the propagation delay of the NAND gate. In the instances when t_{pd} exceeds acceptable levels, parametric violation occurs

on execution time or power, risking a transition of these parameters outside set specifications. Deployment of specific hardware components like spare memory rows or columns for reliability purposes is such an example. Rollback-based mechanisms and Error Correcting Codes (ECC) are also widely adopted to correct binary errors at the expense, however, of extra clock cycles. In these cases, *while functional violations are prevented, timing overheads of such mechanisms contribute to performance variation and pose a threat for system parametric reliability and specifically, timely execution.* In our work, we attempt to manage system performance and limit this type of parametric violations; other system parameters such as temperature and power, will be briefly discussed as well.

1.3 Dealing with Performance Variation: What is Missing?

In terms of software, modern digital systems operate on highly workload-intensive environments, where different applications and tasks are running in parallel, executing functionalities with many conditionalities and constraints. We have also discussed how phenomena in the physical layer can cause reliability violations and contribute to performance variation. These phenomena are only expected to exacerbate as process technologies continue to scale. At the same time, embedded systems companies are becoming more and more interested in power consumption; laptops, smartphones and many other portable devices operate with batteries and their recharge or replacement may be impractical and costly. *In the intensely dynamic context of modern digital systems operation, the mitigation of performance variation and the avoidance of timing violations under minimum power budgets is clearly challenging.*

The features enabling low-power operation and the ones dealing with timing deadlines are, in principle, complementary. On the one side, many techniques exist that focus solely on low-power consumption, not dealing with timing dependability. For example, processor frequency and voltage is statically set to the lowest available points when the "Powersave" governor of the Linux kernel [45] is selected. This technology aims to minimize dynamic power consumption. On the other side, a first, widely-adopted methodology for ensuring dependable performance was the worst-case approach: timing constraints were guaranteed at design-time, after worst-case values for the error rates and task execution times were considered. To account for Process, Temperature and Voltage (PVT) variations, for instance, memory vendors assumed worst-case conditions and demanded substantial design margins [284]. This approach improves the component's reliability nevertheless, memory access latency is kept low, limiting overall memory speed. The two main guardband types still in industrial design

flows are the frequency guardbands and the voltage guardbands. In the first case, maximum clock speed is relaxed so that the variations in the delay of critical paths would always be regarded. In practise, clock period is increased by a time slack to account for the PVT and aging delay variations. In a similar manner, an additional voltage guardband can be added to the nominal voltage level, to compensate for the delay variations. These guardbands nevertheless, penalize performance and power efficiency. As another work claims, "the performance degradation associated with margins, shows that a 20% voltage margin to account for voltage swings in today's 45 nm node translates to $\sim$ 25% loss in peak clock frequency" [225].

Performance boosting methodologies can be utilized to speed up the central processing unit (CPU) clock and avoid timing violations: the "Performance" governor of the Linux kernel sets processor speed to the highest available level [45]. Intel's 'Turbo Boost" is a similar feature, statically boosting the operating frequency outside predefined margins when the user selects it and is already used in numerous processors like the Core i5, Core i7, Core i9 and the Xeon series [117]. Regarding AMD, it has already introduced the "OverDrive" technology that allows the user to overclock a CPU [12]. Evidently, while such approaches could be used to mitigate performance variation and provide timing guarantees, they are again clearly suboptimal in terms of energy consumption.

After realizing this, the research community started to employ adaptive control techniques. The basic principle is to allow the integrated circuit to adapt to certain working conditions like ambient temperature, energy and workload. In computer architecture, for example, numerous adaptive voltage scaling (AVS) methods exist that manage power and timing constraints [274]. Adaptive body biasing has also been proposed as a mitigation technique for threshold voltage variability [189, 273]. Generally, knobs in the circuit-level like voltage or frequency, and task mapping ones such as task scheduling and migration are used to control performance, power and other system characteristics at run-time. In this group of self-adaptive solutions, we can find DVFS technologies [282, 59] and multi-processor, system-on-chip (MPSoC) mapping methodologies [300, 167]. In our case, several works applying DVFS in order to maximize computation throughput with minimum energy budgets exist [33, 68]. *Nevertheless, when examining deadline constraints in dynamic systems, most of these schemes fall into best-effort approaches and cannot provide guarantees.* Such approaches attain "satisfactory deadline miss ratios" and can be found for example in the domain of mixed criticality and mode scheduling [98, 106], in systems utilizing closed-loop control [160, 162] or task mapping and scheduling [172]. Therefore, in conclusion, computer engineers and developers need to focus more on reestablishing guarantees without the use of (too) worst-case guardbands.

1.4 Contributions and Structure of the Thesis

1.4.1 Contributions

We explained the term of performance variation and briefly presented the basic premises and terminology of semiconductor reliability. We also highlighted how challenging it is to mitigate performance variation in dynamic systems while respecting timing constraints and operating at a low-power basis. We now move forward by underlining the contributions of our work:

- First, a classification of prior art, regarding mitigation methods for parametric reliability violations is attempted. This study includes a wide range of works concerning both hardware and software solutions, pre-market and post-market techniques as well as systems in package (SiP) and across packages.
- In the context of reliability analysis, relevant frameworks are developed that estimate time-dependent yield and failure probability of circuits under test. These tools are based on Monte Carlo sampling and the Most Probable Failure Point (MPFP) technique [132] while the dominant circuitry under test is the Static Random-Access Memory (SRAM) cell[3].
- A closed-loop controller that manages performance variation and ensures timing deadlines through proper Dynamic Voltage and Frequency Scaling (DVFS) actuations is introduced and instantiated on a target platform where its capabilities and limitations are explored.
- After discussing the system scenario methodology [95, 96], we examine an enhanced version of the feedback-based controller that can operate at the thread node granularity.
- Finally, we go beyond the system scenarios and develop a proactive DVFS controller based on dynamic scenarios. This way, we are able to guarantee timing deadlines on certain systems and applications which experience performance variation. Again, the controller is deployed on pure hardware and final results verify the approach.

[3]Embedded memories are one of the hardware components most susceptible to degradation because the emphasis on cell size/area minimization induces the use of minimal-sized devices in the cells [54].

1.4.2 Thesis Structure

The structure of the text is organized as follows: Chapter 2 studies the physical mechanisms of the most prominent reliability phenomena threatening electronic systems. These are the sources that under certain conditions create a fault that can manifest as an error/failure. In Chapter 3, we discuss the prior art on functional and parametric reliability since the two terms are closely connected. Emphasis is given however, on the mitigation techniques for parametric reliability and especially for timing violations. Chapter 4 presents specific tools we have developed, in order to efficiently capture and estimate the impact of pronounced reliability phenomena, namely BTI and Random Dopant Fluctuation (RDF) [7], on a system's performance. In Chapter 5, the feedback-based controller is introduced; by proper DVFS decisions we aim to ensure performance dependability and "absorb" the timing overheads of RAS interventions. Chapter 6 discusses the system scenario methodology and shows an enhanced version of the controller, operating on the thread node level. In Chapter 7, we show a proactive DVFS controller that is based on dynamic scenarios and aims to guarantee timing deadlines on specific steaming applications with evident performance variation. Finally, Chapter 8 presents some concluding remarks while aspects of future work are briefly noted. The chain of events trigerring performance variation and timing violations are shown in Schematic 1.5. The structure of the Thesis is presented in Figure 1.6.

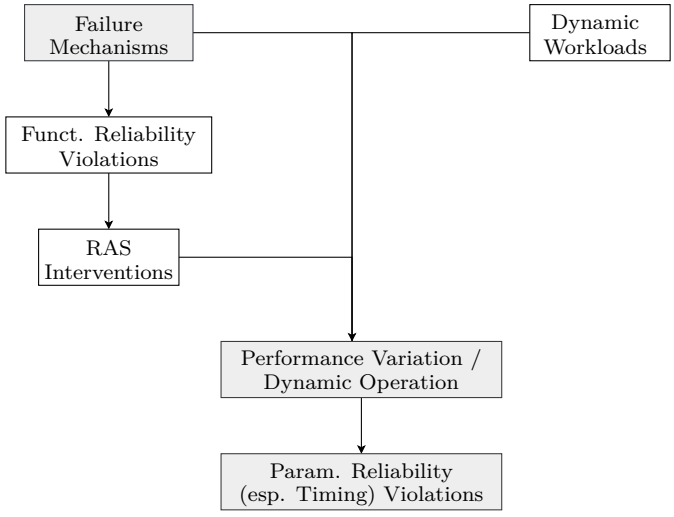


Figure 1.5: Schematic showing the dependencies between the events. Gray boxes highlight the terms thoroughly studied throughout our work.

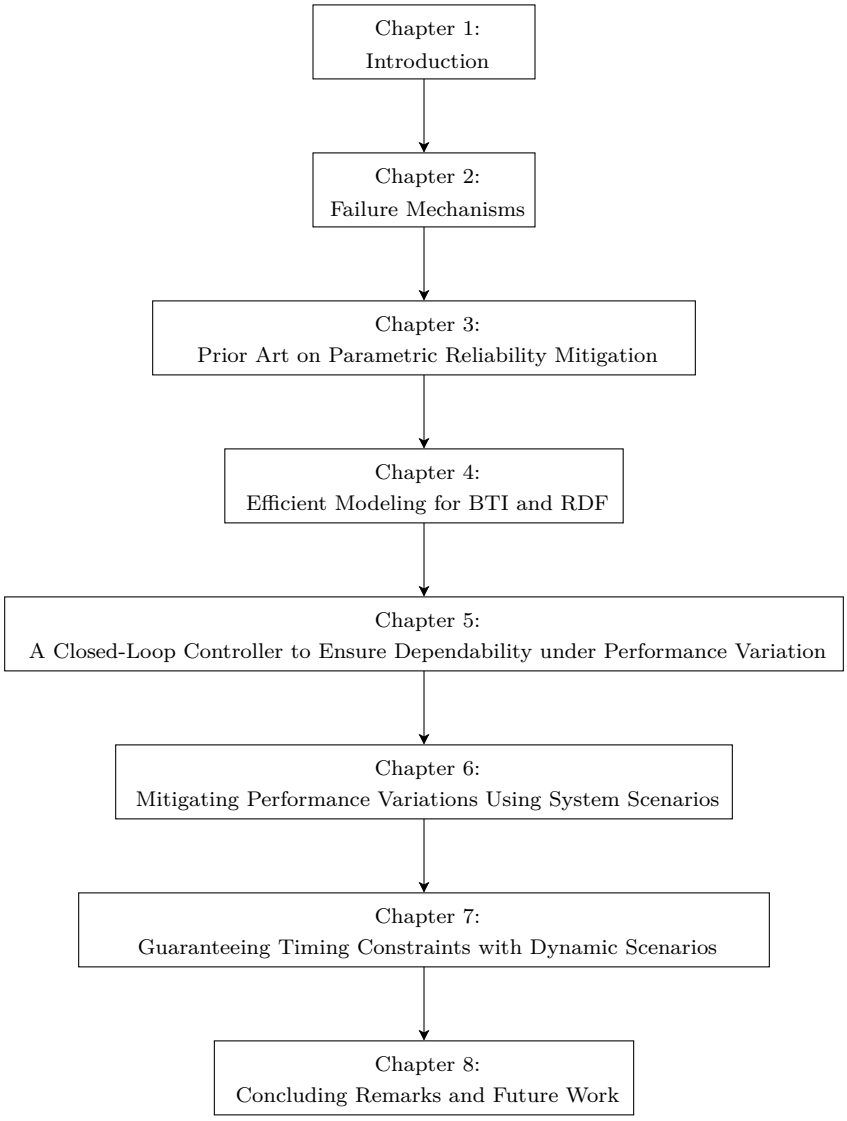


Figure 1.6: Chapters of the Thesis

Chapter 2

Failure Mechanisms

In this Chapter, we elaborate on the failure mechanisms of circuits leading to reliability violations and eventually performance variation. First of all, variations in the voltage supply can cause circuit delays as well as functional errors. *Time-zero variability*, caused mainly by manufacturing variability during the process development deteriorates system performance. For example, minor defects in the chemical or lithography process affect the characteristics of modern, downscaled devices. Variations in the implanted impurity concentration of the channel dopants can also alter a transistor's threshold voltage. In addition, aging phenomena dependent on a system's lifetime, trigger *time-dependent variability*; electric fields applied at the gate of a MOSFET, gradually degrade the oxide and generate defects either in the oxide-substrate interface or the oxide itself. Defects can also be created in the wires and interconnects of circuits, leading to voids and hillocks on the conductors. Finally, a brief discussion on radiation-induced transient faults completes this Chapter.

2.1 Voltage Supply Variations

While digital systems are designed to function at specific voltage levels, voltage supply fluctuations are likely to occur during a chip's operation. Typical reasons for this are the di/dt noise (also known as delta-I noise) and the IR drops, which are generated respectively due to parasitic inductances and resistances across the sypply rails. In most CMOS circuits, propagation delay is inversely proportional to the voltage supply hence, this type of V_{dd} variations have a direct impact on the system's performance. In fact, prior works estimated a

30% variation in the delay due to a 10% V_{dd} fluctuation for typical CMOS gates at 0.13 μm nodes, while in 90 nm technologies, 4% t_{pd} variation was caused by 1% variation in the voltage supply [269, 240]. As we have already explained in Section 1.2.2, large timing delays can also lead to logic errors, when manifesting in sequential blocks and circuits with matched delays. Because of this, system engineers are extremely careful when designing the power supply regulation and distribution components. Finally, it should also be stressed that most aging mechanisms are voltage-dependent therefore, voltage overshoots can exacerbate these phenomena.

2.2 Time-Zero Variability

2.2.1 Process Variation

Printed transistors, often have slightly different characteristics from the ones primarily designed. Furthermore, printed transistors of the same technology may not be identical with each other. This is because the manufacturing process is not ideal; on the contrary, during this procedure numerous sources of variability exist that alter certain MOSFET characteristics, to a very small degree[1]. At the end, this variability affects (among others) the current of the channel and eventually, the delay of a circuit and its energy consumption.

Random Dopant Fluctuations

In order to follow Moore's law and continue with device scaling, it is necessary to sufficiently dope the channel region with implanted atoms [249]. Fluctuations in the number of these dopants occurred even with transistors of older technologies. Nevertheless, the phenomenon was irrelevant before, mostly because this number was eventually averaged out. In modern devices however, and with the length of the channel being in the nano-scale dimension, this number is limited to less than a hundred. Moreover, even the spatial position of these dopants varies, affecting device performance and mainly the threshold voltage [249, 292]. According to the theory on RDF [259], a device's V_{th} is not fixed when exiting the foundry but is distributed near a specified value with a Gaussian being a sufficient approximation. This variability is formulated as shown in Equation

[1]This variation is classified between *systematic* (or deterministic) and *random* (or intrinsic). Systematic variation occurs in elements placed next to each other and includes, among others, lithography proximity effects (LPE) [249]. Systematic variation can ultimately be controlled by the designer and described through deterministic models [175]. In this Chapter, we focus solely on random process variability.

2.1, where W and L are the channel's width and length respectively. A_{VT} is Pelgrom's mismatch parameter [208]. According to this Equation and inline with RDF theory, as devices shrink, V_{th} variability becomes more pronounced.

$$\sigma_{Vth} = \frac{A_{VT}}{\sqrt{2WL}} \tag{2.1}$$

Oxide Thickness Variation

The aggressive shrinking of device dimensions, inevitably led to the scaling of the oxide dielectric. In transistors at the 65 nm nodes and beyond, physical oxide thickness (t_{ox}) is limited to 1 nm (= 10Å) which is equal to only five atomic layers [34]. The oxide affects the capacitance of the gate and its ability to attract charge to the channel. In addition, a MOSFET with a very thin oxide suffers from high gate leakage currents since mobility carriers can easily "tunnel" through the oxide. Controlling therefore the thickness of the gate dielectric with remarkable precision is crucial. Nevertheless, because of the non-ideal oxidation process, variation in the t_{ox} has been observed: recent experiments have estimated a $\sigma t_{ox} = 0.1$Å in oxides with 1 nm thickness [138]. In relevant simulations, the impact of this effect is shown to add $\sim 10\%$ in the overall σ_{Vth} (Equation 2.1) caused by RDF [17].

Line Edge Roughness

Similar to oxide thickness variation, the scaling of transistors has brought about severe challenges in the lithographic printing technology. Specifically, as the feature sizes now reach few tens of nanometers, the lithography process can no longer flawlessly meet the high industry standards. Line Edge Roughness (LER) refers to the variations in the gate patterns of fabricated devices or the roughness in the edges of the printed patterns [249]; due to the extremely downscaled technology nodes, its impact on device performance can no longer be neglected. In detail, "LER is mainly caused by erosion of polymer aggregates at the edge of the photoresist (PR) during development and fully depends on some complex chemical formulas" [26]. The variation of a channel's length across its width due to LER is called, in short, Line Width Roughness (LWR). LWR affects the drain and leakage currents and thus directly impacts the device performance. The variation in the channel length is typically estimated as a percentage of its nominal length; prior work, for example, mentions a $\sigma = 0.02$ μm, for a technology node of 0.4 μm [181]. While simulations have shown LWR

impact to be relatively small compared to the σ_{Vth} of the RDF, in extremely downscaled devices LER variation could be quite important [17, 18].

2.3 Time-Dependent Variability

2.3.1 Oxide Wearout

Apart from process variation, that occurs during the manufacturing process, several mechanisms can degrade a transistor's performance during its lifetime, such as BTI and Hot Carrier Injection (HCI). We have already explained how the dielectric layer between the gate and the channel uses ultra-thin oxides in order to keep up with the device scaling rules. During system operation however, these thin oxides become susceptible to certain wearout phenomena, also referred to as aging. The primary reason for the provocation of these mechanisms is the electric field systematically stressing the oxide; interestingly, part of the wearout damage is annealed when stress is removed. This renders these phenomena largely *workload-dependent*.

Bias Temperature Instability

The dominant source of aging in modern devices is BTI. The phenomenon was first observed to affect mostly short-channel pMOS transistors, thus it was referred to as Negative Bias Temperature Instability (NBTI); now it is also largely known to affect nMOS devices as well. BTI had already been identified as a reliability threat more than 50 years ago [73] however, its impact became prominent in technology nodes below 130 nm [243, 9]. While many details remain currently unknown, the physical mechanism creating BTI is believed to be the generation of interface traps and oxide charge because of the vertical electric field applied between the gate and the substrate [120]. It is also noteworthy that BTI is more intense in transistors with high temperatures with recent models suggesting an exponential dependence [104].

Specifically, when electric field is applied at the gate of the MOSFET (through the V_{GS}), some of the Si–H bonds at the oxide-substrate interface break and the H_2 atoms are diffusing towards the gate, while the remaining Si becomes charged, creating interface traps that may capture and release minority carriers [101]. In addition, minority carriers can also be trapped in the oxide due to defects in the Si–O bonds, such as a non-bridging oxygen atom or a silicon vacancy. Charge trapping affects carrier mobility and leads to a "shift" in the threshold voltage of the transistor, eventually slowing it down. According to

the model, some of the Si–H bonds are annealed when gate stress is removed, partially recovering the damage. The number of interface traps can be estimated based on the following Equation 2.2 [206]:

$$N_{it}(t) = \sqrt{\frac{k_f N_o}{2k_r}}(D_H t)^{\frac{1}{4}} \tag{2.2}$$

where k_f is the dissociation rate constant, k_r is the rate of reverse annealing of Si–H bonds, N_o represents the total number of Si–H bonds and D_H shows the coefficient of the hydrogen diffusion process. Due to the interface traps, the threshold voltage shift can be modeled through the Equation 2.3 [206]:

$$\Delta V_{th}(E_{ox}, t) = (1+m)\frac{qN_{it}(E_{ox}, t)}{C_{ox}} \tag{2.3}$$

E_{ox} is the electric field applied at the oxide, q is elementary charge and C_{ox} gate capacitance and m accounts for extra V_{th} shift because of mobility degradation. As the Equations 2.2 and 2.3 show, ΔV_{th} due to BTI follows a 1/4 power-law dependence with time (t). Recent work on BTI includes the atomistic theory which, besides the reaction-diffusion part, adds a stochastic component that captures the recovering phase and the workload dependency in a more efficient manner [104, 230, 290]. This stochastic component of the atomistic theory describes the probabilistic nature of oxide defects; it is also able to capture **random telegraph noise** (RTN). RTN, similar to BTI, can be modeled through the trapping and de-trapping processes of minority carriers at the gate insulator/Si interface.

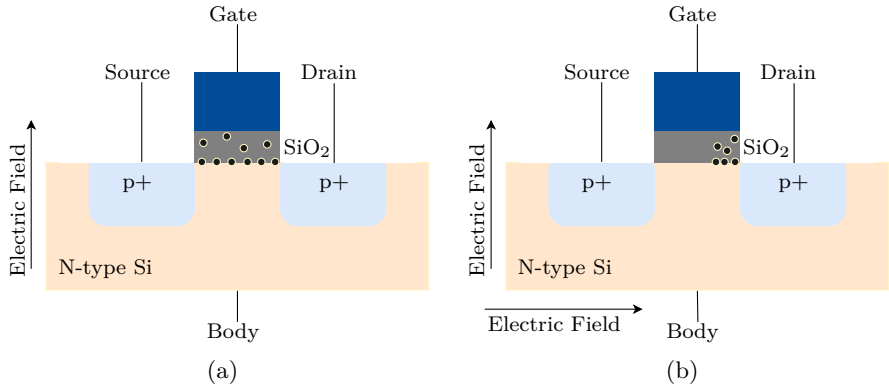


Figure 2.1: Interface traps and oxide charge in a pMOS: a) Because of BTI, defects are located across the oxide and the channel; b) due to HCI, defects are located on the interface and the oxide, mostly near the drain region.

Hot-Carrier Injection

Oxide charge can also be trapped during the device operation because of HCI and cause degradation. Due to the lateral electric field on the channel (caused by strong V_{DS} or a sudden overvoltage), some minority carriers collide with silicon atoms located on the drain-substrate interface. This collision can create a pair of electrons and holes with some of these particles having sufficient energy to surpass the energy barrier and move to the oxide [144, 61, 112]. Actually, these "hot" carriers may even tunnel through the oxide or to the substrate, increasing the gate leakage current in the former case and the substrate current (I_{sub}) in the latter. While fist empirical or phenomenological HCI approaches failed to describe the phenomenon in detail, recent physics-based models properly consider defect generation mechanisms and the carrier distribution functions and have been experimentally verified [38, 247]. Similarly to BTI, HCI also raises the threshold voltage and reduces carrier mobility. Again, after the trapping of the charge and during the relaxation phase, when V_{DS} is removed, partial annealing of the damage is enabled[2]. In contrast to BTI, HCI becomes more prominent with the dropping of ambient temperature, because of the relative decrease in carrier mobility [185]. An illustration of the BTI and HCI generated traps on a pMOS device is presented in Figure 2.1.

Time-Dependent Dielectric Breakdown

The gradual build-up of trapped minority carriers in the oxide damages the gate dielectric. When the vertical placement of this oxide charge allows the formation of a conducting path between the gate and the substrate, the current through the channel is suddenly reduced and intermittent or even permanent faults occur, preventing the transistor's correct operation. Oxide thickness, voltage supply and device temperature largely affect the phenomenon. In fact, the time to breakdown is also correlated to the material of the dielectric, which in turn depends on the manufacturing process node [299, 271]. Older generation products are based on SiO_2 while modern transistors use high-k dielectric materials such as Hafnium dioxide $Hf0_2$. Evidently, the phenomenon is more prominent in today's high-k dielectrics since they are significantly thinner. Relevant work [257] studies Time-Dependent Dielectric Breakdown (TDDB) in detail.

[2]It should be noted that hot electron injection on floating-gate MOSFETs (FGMOS) is the main mechanism utilized to create certain non-volatile memories like flash memories.

2.3.2 Wearout of the Interconnects

During the system operation, defects can be provoked in the wires and the interconnects of a circuit as well. The three widely discussed failure mechanisms that regard the interconnects are electromigration (EM), self-heating and TDDB of the low-k dielectric, insulating the conductors. Again, since these phenomena are highly correlated to the current density and the electric fields, their *workload-dependence* is underlined.

Electromigration

Electromigration (EM) is caused by the continuous electron flow on the interconnect [115, 266, 260]. Specifically, as the electrons are moving across the wire, they collide with the conductor's atoms, displacing them and creating voids in regions where several atoms have been drifted or extrusions in regions where a large amount of these atoms is accumulated (as shown in Figure 2.2). These voids may even result in open circuits, severely damaging the system while the extrusions can cause shorts between lines or layers. Following the scaling of transistor dimensions, interconnects are also shrinked and now, with increased current densities, EM poses a serious threat for reliability engineers. Evidently, the phenomenon typically damages interconnects in which DC currents are flowing [155] but also wires that are very thin (dense grids) in which current fluctuates in magnitude but is always flowing in the same direction. That is the case for signal wires in dense arrays like memories or logic arrays such as multipliers. To describe EM, Black's Equation is still widely used [42].

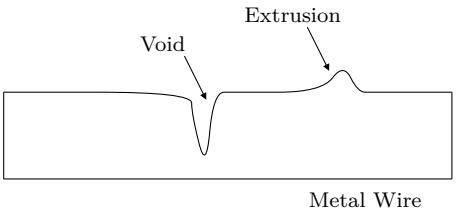


Figure 2.2: Electromigration on a metallic wire, forming voids and extrusions.

Self-Heating

The current flowing through the wires dissipates power that in turn, leads to a rise of the conductors' temperature. In modern chips, wires and interconnects are typically surrounded by oxides and dielectrics which are thermal insulators and since heat convection is not enabled, some wires may become alarmingly "hot". Resistance on these hot wires is increased significantly, along with the propagation delay [8, 235], rendering self-heating a reliability threat. In addition,

as EM is largely temperature-dependent, hot wires are also more susceptible to electromigration.

Time-Dependent Dielectric Breakdown in Low-k Interconnects

The low-k dielectric breakdown in interconnect stacks has become a considerable failure mechanism. First of all, the aggressive scaling of the electronic chips lead to a dramatic shrinkage of the distance between the interconnects. In order to reduce the RC parasitic delay of the wires and self-heating, low-k insulating materials have been extensively utilized . This, however, along with the fact that low-k dielectrics have intrinsically weaker breakdown strengths made the TDDB of the low-k dielectric a prominent reliability issue. Apart from temperature, there seems to be a clear dependence of low-k TDDB to applied electric field (E). Currently, numerous models exist trying to describe the phenomenon including the E model [291], $\sqrt{E}$ [58], E^2 [5] and $(1/E + \sqrt{E})$ [158] models, depending on the exact physical mechanism inflicting the damage on the insulator and the field's dependece.

2.4 Radiation-induced Transient Faults

Apart from process variation and degradation effects, transient faults can also occur due to radiation particles striking the semiconductor [265, 30]. There are two types of particles that can cause such faults on a digital system: alpha particles and high energy neutrons. An alpha particle consists of two protons and two neutrons and can be emitted as radioactive decay from system packages that typically contain small amounts of trace uranium and thorium impurities. In fact, transient errors due to alpha particles were first noted by Intel in 1979 [173]. High energy neutrons, on the other hand, are particles generated as cosmic rays "bombard" the atmosphere and are highly penetrative.

While the interaction of these two particles with the transistor's silicon atoms is different to a small degree, the physical principle behind the transient/soft fault provocation seems the same: the particle collides with the silicon atoms of the substrate across a specific depth, ionizing these atoms. The ionization creates numerous pairs of electrons and holes which are not recombined; on the contrary, due to the electric field and depending on the force of the collision and the geometry, they are swept into the depletion regions and finally to the contacts of the device, creating a spike current called *single-event transient* (SET). If the charge of the carriers drawn into the contact is higher than a

specific threshold, defined as critical charge (Q_{crit}), a *single-event upset* (SEU) occurs and the device malfunctions.

With transistor scaling, critical charge is rapidly dropping as well. Nevertheless, since the area of the device is also shrinking (and therefore there is a lower chance of a radiation particle actually interacting the silicon), the overall FIT rate due to radiation-induced errors remains nearly the same – or even decreases– with newer process technologies [185]. Actually, at the sea level, SRAM memories of a standard 0.6 μm CMOS process are estimated to suffer from 100 to 2000 FIT/Mb due to SEU [110] while at cruising altitudes, this number increases up to 10^6 FIT/Mb [304].

2.5 Conclusion

This Chapter briefly discusses the physics behind the most prominent, hardware-related failure mechanisms that affect the performance of digital systems. Transistor variability during the lithography or the oxidation process and fluctuations in the impurity concentration of dopants in the channel region deteriorate reliability even before the electronic products are shipped to customers. Gradual oxide and interconnect wearouts that occur throughout the system's lifetime, along with radiation-induced transient faults, constitute another risk for reliability violations. Apart from functional failures, the aforementioned phenomena eventually trigger system-wise parametric reliability violations and performance variation, posing a serious threat for the dependability of an electronic product. This highlights the need for developing effective modeling techniques and mitigation solutions.

Chapter 3

Prior Art

3.1 Introduction

We have already highlighted the importance of *functional* and *parametric reliability* on today's digital systems in Chapters 1 and 2. In this Chapter, we will focus on the current research in the mitigation techniques targeting the latter reliability component. In collaboration with Dimitris R. [228] we have classified this prior art in a complete top-down fashion, covering all aspects of parametric reliability. The classification methodology itself, has been elaborated in detail in previous work [228]. Nevertheless, we will start with a short summary of this process. Then, we will briefly discuss certain approaches that study the mitigation of functional reliability violations. While this is not the main focus of our work, we have already explained in Section 1.2.2 how it is highly correlated to parametric reliability and can often trigger performance variation. Then, the prior art regarding the mitigation techniques for parametric reliability violations will be examined in detail and the classification framework will be introduced. This related work covers, in fact, a wide range of research fields such as systems in package and across packages, pre-fabrication and post-fabrication implementations and even solutions employed in the software as well as the hardware of digital systems. Apart from reliability mitigation, computer engineers and the VLSI industry in general also focus on reliability analysis and utilize modeling tools to describe and quantify the impact of the failure mechanisms. Hence, a brief overview on existing reliability simulation approaches and related frameworks will be presented at the end of our Chapter.

3.2 Building the Classification Framework

The methodology creating our classification framework is shown in Figure 3.1 and is based on a top-down division principle [142]. The domain under study – in our case, *parametric reliability mitigation* – is the parent class (P) and the root of the tree.

Based on a relevant criterion, the parent class is split in two sub-categories S_1 and S_2 that are strictly:

- *complementary*, in a sense that S_1 describes an X part and S_2 the "not X" part, that is $\bar{X}$;
- *non-overlapping*, so that an intersection of the two sub-categories is zero ($S_1 \cap S_2 = \varnothing$);

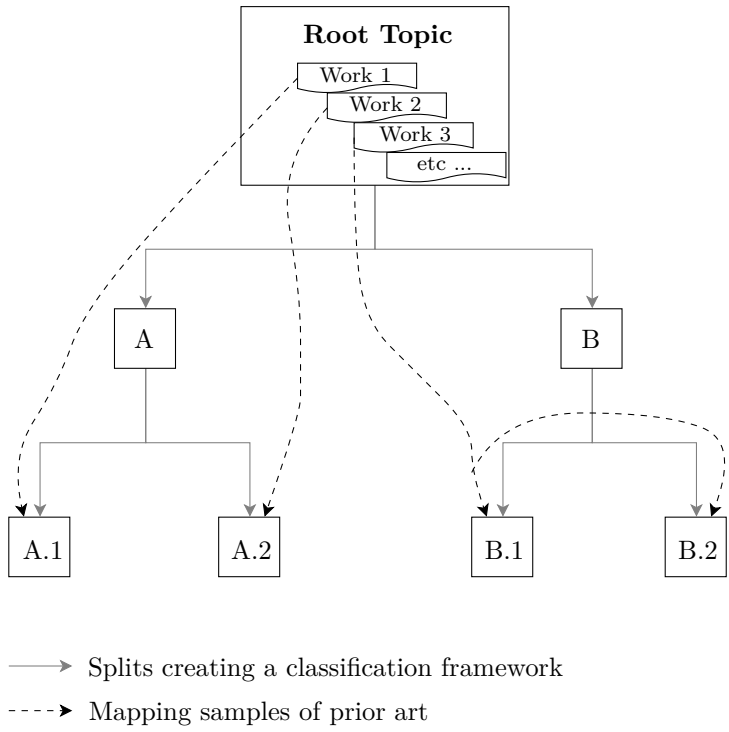


Figure 3.1: The classification framework built after the binary splits on the root topic. In this example, "Work 3" is a hybrid mapped in both "B.1" and "B.2".

- *complete*, meaning that the two sub-categories together can compose their primitive set ($S_1 \cup S_2 = P$).

The aforementioned properties enable the complete and consistent systematic classification. In addition, binary splits continue to decompose the tree nodes until the top-down division reaches sufficient depth. Finally, a large number of sub-categories are designed that are also sufficiently granular to completely describe and analyze the research domain. Evidently, the extent of the analysis is reflected in the tree's depth and the number of its nodes. In our instantiation, we have mainly focused on a broad classification and not a very deep one.

After the framework is created in the form of a binary tree, samples of prior art are *mapped* in the leaves of the tree. Starting from the root and following the splitting criteria, each work sample is classified to the most suitable leaf. In our example, "Work 1" belongs to the "A" node and is finally mapped in the "A.1" category/leaf. Similarly, "Work 2" is placed into the "A.2" leaf. It is quite usual for a prior art sample to discuss cross-layer approaches or cover multiple sub-domains and thus belong in more than one tree leaves. Such works are labeled as "hybrids" (see "Work 3" in Figure 3.1). However, based on this classification and the splits, it is always possible to rigorously partition the parts of the approach that belong to each of the leaves.

In contrast to other classifications of prior art found in typical surveys, our methodology shows the following unique features: i) since the splits create two complementary sub-categories, we always cover the complete research field and do not ignore potentially relevant sub-domains. ii) The framework can be easily extended by interested readers, using the same methodology; further splits can be induced at the leaves of the primary tree, to create a more granular analysis of the root topic. iii) Prior art samples can be seamlessly mapped in their relevant sub-domains and objectively compared, following the mapping principle of Figure 3.1. It is indeed much easier to compare the primitive partitions of an approach than to directly compare the complex composite. iv) We can easily highlight leaves in the classification tree that are not (sufficiently) well covered by existing literature, once we have mapped the main publications to the tree. In this way, we can identify promising research gaps. We can also map requirements of applications to these trees and select the most promising leaves for that application domain. This, however, is beyond the scope of the current text.

3.3 Mitigation of Functional Violations

The functional correctness of digital systems is of paramount importance for the computer industry. In order to ensure correct, bit-wise operation and to defer or avoid bit errors and system failures, computer engineers have developed a plethora of techniques spanning across different abstraction layers: from the microarchitectural and system level, to the firmware and software layers. Previous work [218] presents an interesting classification of the *functional reliability mitigation* domain, categorizing and briefly discussing the majority of those schemes. A related survey on mitigation techniques regarding degradation and aging can also be drawn from prior art [83] nevertheless, in this work, parametric and functional reliability are not distinguished. Moreover, while the survey presents a cross-layer approach, it fails to classify the mitigation techniques into complementary categories. Other relevant works focus solely on NBTI mitigation [60, 148]; again, however, their aim is to diminish the impact of the phenomenon and no discussion on the split between parametric and functional reliability is considered.

We will now discuss the basic functional reliability mitigation approaches, first focusing on the hardware and architecture-based techniques. The addition of multiple modules, for example, with the same specification as the primary one, and the majority voting at the output is a classical mitigation technique. No error detection scheme is required since the voting masks the error. In general, this methodology uses an odd number N of modules to avoid uncertain output votes (N-modular redundancy). Typically however, the scheme is deployed in the form of triple modular redundancy (TMR) [277, 165, 286]. Additional hardware can also perform another set of functions and not act as a replica module or a spare component; the addition of an extra hardware register and the utilization of Error Correcting Code (ECC) memory is such an example. ECC memory is a type of computer data storage that is able to detect and correct internal data corruption and bit errors [57, 126, 56]. While the advantages of ECC memories are the low area and power costs, the induced performance overhead cannot be neglected. To prevent erroneous operation, faulty components can be bypassed or powered off; this way, the system continues to operate but usually at a degraded performance. CPU hot-plug is a feature in Linux kernels, for example, that allows the user to deactivate a defective CPU or bring it back online when its operation is restored [99]. Finally, we can find several hardware-based task migration techniques [217, 283] that simply modify the manner in which hardware resources are assigned and migrate error-prone tasks to the more robust components.

Several techniques can exploit the software layers of a digital system. The introduction of additional tasks, for example, to provision the functional

reliability has already been studied. In contrast to using additional hardware, parallel tasks can also run on the software stack and through voting can provide error protection [87, 124]; previous work suggests the technology of redundant multi-threading [78]. This usage, however, of the same resources for the redundant tasks inevitably leads to performance degradation. Compilers can reschedule and reorganize an application's instruction flow, after a characterization of the application's vulnerability profile [298, 227]. In multiprocessor systems, rescheduling and task migration/allocation policies have already been proposed to prevent system failures [97, 35, 187, 170]. Another widely used technique involves storing the state of the system and retrying task execution at intermediate points. Specifically, at several checkpoints across the workload, part of the system state (and potentially additional information) is proactively stored; after the detection of an error, a *rollback* to the last checkpoint of correct system operation is set. The checkpoint placement can be decided, during the offline phase, at design time or at runtime (on-line phase) although in the latter case, extra effort has to be invested. Several examples of offline checkpointing can be found in prior art [53, 10, 200]. Works that address online checkpointing are typically more complex and are implemented at the kernel [80] and user levels [213, 255].

3.4 Mitigation of Parametric Violations – Related Classifications

A crucial concept in the current text is the fundamental distinction between functional and parametric reliability violations and the isolation of the latter component. While both counterparts are equally important in order to address dependability issues, in this Section we will discuss – and classify – prior works and techniques developed to deal with parametric reliability. As we illustrate in Figure 3.2, the parametric aspects in general can include techniques managing direct figures of merit during the IC operation as well as cost-related strategies and decisions regarding design economics. For instance, plans to drastically reduce non- recurring engineering (NRE) costs for the design and masks, including personel and prototype manufacturing costs, belong in the latter group and our outside the scope our work. Regarding the techniques targeting direct figures of merit, they can be further split into the ones dealing with dependability and timing-related constraints and the ones aiming to control other parameters of the operation instance; these parameters typically include either power or temperature related aspects (low-power operation, dark silicon technologies etc.). Contrariwise, the timing-related issues can involve short-term solutions such as methods to reduce critical path delay and manage deadline

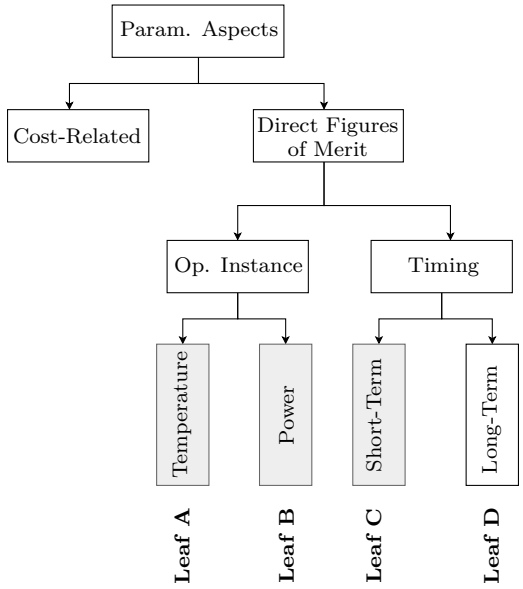


Figure 3.2: Aspects of parametric reliability. Gray boxes highlight the direct figures of merit we explore in our classification.

constraints and approaches affecting the system's long-term timing aspects, like for example improving IC lifetime. In our work, we concentrate mostly on short-term, timing-related techniques (Leaf C) while certain aspects of temperature and power management are also explored (Leaves A and B).

Throughout the literature, we can find several related surveys on parametric reliability. In most cases, such attempts restrict themselves and analyse mitigation techniques on a limited scope, not covering the entire potential space. For example, numerous works target parametric reliability from the scope of *temperature* management (Leaf A); we have already explained how system performance and reliability are degraded by the rise of temperature since degradation phenomena are exacerbated. Prior work tries to classify thermal management techniques however, the classes of each hierarchy level are not complementary [139]. Therefore, hybrid works that belong in more than one category are not properly accounted for. A more systematic classification on thermal management is performed in another work [79]. The analysis however, is not granular enough and the binary tree is not sufficiently deep. Therefore, fine-grain distinction of thermal management techniques is not achieved.

During the last twenty years, *energy/power* management has become a critical

feature in many embedded platforms. To this end, numerous techniques exist throughout the abstraction layers, from hardware design to architecture and software domains belonging in the *B* Leaf of our classification in Figure 3.2. Most efficient methods include a top-down approach, targeting the application and architecture domain and moving to lower system and design levels. In fact, since leakage currents are strongly dependent on transistor temperature, most thermal management solutions also apply for power management. A recent survey on such techniques for mobile processing units (CPUs and graphics processing units a.k.a. GPUs) is presented in prior art [137] and focuses solely on OS-level schemes. Another interesting classification for power management methodologies is presented in relevant work [25]. Techniques are divided in two separate categories: Dynamic Power Management (DPM), where a few, low-power, inactive states are introduced by power-gating certain circuit blocks and DVFS, where voltage and frequency levels are set to reduce energy consumption.

When focusing on parametric reliability in the sense of *timing dependability* and avoiding timing violations (see Leaf *C* of Figure 3.2), we can find numerous related works dealing with highly dynamic workloads. Contrariwise to static task sets, where the complete characterization of the workload and its timing demands are known beforehand, in dynamic workloads, scheduling and matching of jobs to appropriate resources and resource attributes is performed at runtime. Certain survey works deal with feedback-based, real-time task scheduling algorithms [2, 161]. A related survey on hard, real-time scheduling has also been presented [71], targeting however, solely multi-processor systems. A closed-loop scheduling framework has been presented for soft, real-time systems [162] as well. While the aforementioned work falls into a best-effort approach and cannot provide deadline guarantees, it attains satisfactory deadline miss ratios.

For strict deadline guarantees, modern processors typically utilize voltage and frequency margins [121]. However, this level of robustness comes at the cost of lower processor performance and power efficiency [226]. A recent paper [149] presents a DVFS scheme that introduces a Worst-Case Ready Queue (WCRQ) algorithm to guarantee timing deadlines and tests this scheme on the Gem5 simulator. The authors assume that future tasks are partly known and reside on this ready queue. Due to the stochastic nature of highly dynamic workloads however, such an assumption cannot always be safe. The same applies for another related work where a real-time scheduler is also exploited [211]. Towards this direction, certain DVFS schemes in the Linux kernel are actuated based only on the last period CPU-usage statistics. "*Ondemand*" and "*Conservative*" Linux governors, for example, set the CPU frequency depending on the current system load [45]. Again however, the latter implementations fail to provide timing guarantees.

3.5 Mitigation of Parametric Violations – Our Classification

As explained in Section 3.2, we first define parametric reliability mitigation as the root of our tree (see Figure 3.1). Then, we derive a split based on the type of electronic packaging. Two branches are created: one focusing on *systems within package* and a second regarding *systems across packages*; we further split the former, in order to equally address *single-die* and *multiple-die* arrangements.

3.5.1 Systems Within Package: Single-Die

Mitigation techniques for parametric reliability on a single-die can be further split between *pre-fabrication* and *post-fabrication* ones. The classification of related work on this branch is presented in Figure 3.3.

Pre-Fabrication

Pre-fabrication solutions are applied before the die is fabricated and can be classified between two complementary domains: *hardware* and *software*. In addition, the former group can be further split into *design related* and *process technology*. Design-related techniques can be implemented either by *optimizing the existing design* (without the addition of new features) or by *enhancing* the design by incorporating extra components and properties with respect to parametric reliability.

Mapping 1. A transistor sizing technique to account for BTI wearout, is an example of an optimization design solution [134]. An "NBTI-aware processor" created through optimized combinational and memory-like blocks has also been proposed [4]. Finally, related work suggests a layout co-optimization framework to improve pin access of the standard cells and maximize pin access flexibility for routing [296].

Mapping 2. A gate splitting methodology that adds redundant paths to reduce delay variability is proposed in prior art [77]. The use of redundant columns in SRAM memories to reduce SRAM leakage in the presence of random delay variation has also been introduced [103].

Mapping 3. An aging-aware compiler to uniformly distribute the stress of instructions in order to minimize aging of general-purpose GPU architectures is discussed in previous work [221]. Another work studies a timing variation-aware scheduling and resource binding in high-level synthesis (HLS) approach to mitigate the effects of performance variation [183].

Regarding the category of process technology, we refer to all mitigation techniques implemented during the manufacturing procedure of the wafer in

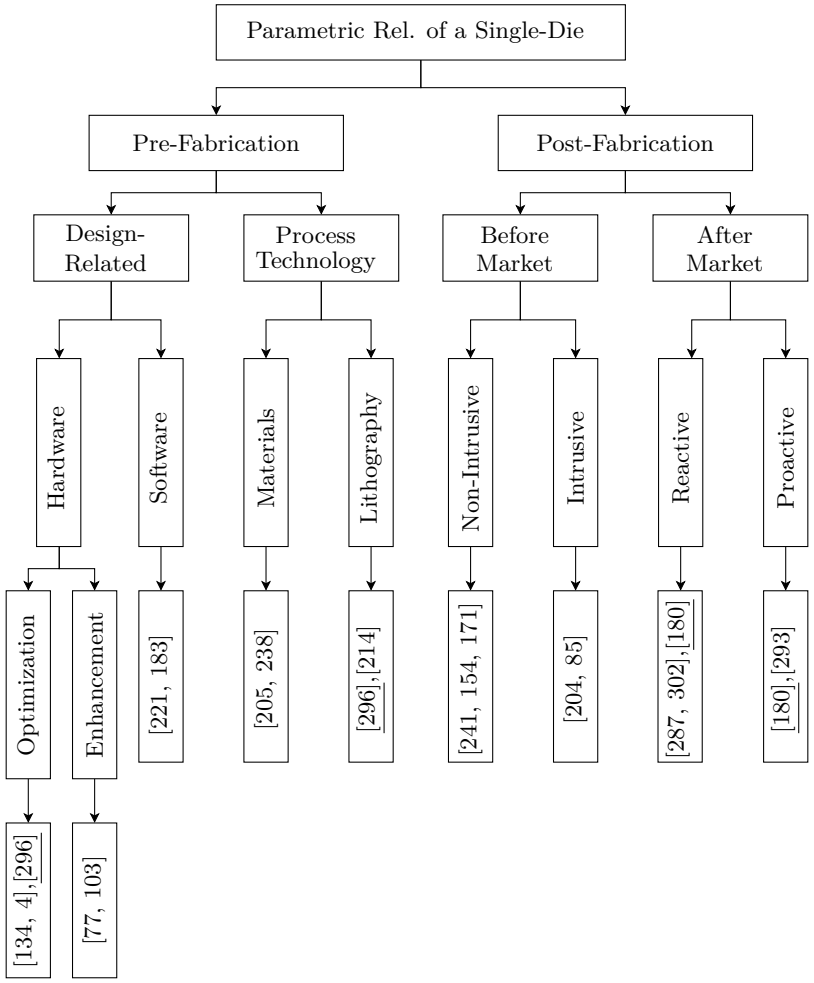


Figure 3.3: Classification Branch for Single-Die Solutions.

order to minimize parametric reliability violations. Such techniques may concern either the selection of *materials* or the *lithography* process.

Mapping 4. An alternative to typical CMOS technology has already been available and refers to the Silicon on Insulator (SoI) technology [205]; this technology suggests that devices are fabricated on an insulator and stands in contrast to to conventional bulk processes leading to higher-speed devices. Another example is the surface nitridation of metal wires that reduces line resistance and increases reliability [238].

Mapping 5. Enabling designers to increase the feature density of chips through the double patterning technique of the lithographic process is discussed in a previous paper [296]. Another work suggests the use of extreme lithography wavelengths and multiple patterning to deliver reliable flash memories [214].

Post-Fabrication

After the IC has been manufactured, certain mitigation techniques can be applied *before shipping the die to the market*. Alternatively, several solutions have been developed to mitigate parametric reliability *after the packaging and shipping of the chip to the market*. Before bringing the die to the market, available techniques can be further divided between *non-intrusive* and *intrusive*.

Mapping 6. A testing phase to properly characterize and predict platform and application runtime variability is discussed in prior art [241]. Voltage binning to improve product yield has also been proposed [154]. Moreover, measuring the supply current in the quiescent state for the presence of manufacturing defects (iddq testing) is already widely used [171].

Mapping 7. An intrusive circuit repairing methodology using localized laser pyrolysis has been presented in previous work [204]. In addition, pulse current trimming can be utilized to mitigate process variability in polysilicon sheet resistance [85].

After the die has been shipped, mitigation solutions can be split between *reactive* and *proactive* ones.

Mapping 8. A reactive speed control solution for runtime, temperature-constrained applications has been studied in relative work [287]. Other thermal control policies for runtime situations based on a reactive response also exist [302]. A self-tuning technique to improve product lifetime under aging phenomena is discussed in previous work [180].

Mapping 9. Mintarno et al. [180] also study a proactive DVFS scheme based on degradation predictions and aging models. An energy efficient scheme through the use of artificial neural network predictions has been discussed in prior art as well [293].

We should note the hybrid papers of Xu et al. [296] and Mintarno et al. [180]; these works are "local hybrids"as they are concetrated in this specific branch of the tree.

3.5.2 Systems Within Package : Multiple-Dies

In this category, we refer to the Systems in Package (SiP) where the set of dies are packaged into single discrete components. Our classification is then further split between the *integration* of dies and the *addition of extra features* in the SiP (apart from the features that provide system functionality). This categorization is illustrated in Figure 3.4.

Integration

Regarding the multi-die integration, we can study techniques that look into the *wiring* at the various levels or the *stacking* procedure per se. In the former domain, the two most widely used options are : *mechanical* wiring (such as wire bonding) and *thermal/chemical*, like soldering.

Mapping 10. Previous work studies Pd-coated, Cu wire bonding for high reliability semiconductor devices [254]. Specifically, this work examines the critical aspect of bringing Cu and PdCu wire technology to wire bonding process development and achieving high reliability requirements.

Mapping 11. The in-situ reliability monitoring of micro-bumps is the main focus of related work regarding through-silicon via (TSV) interposers [28]. Another paper looks into the process-level reliability of solder bumps in 3D semiconductor packaging [46]. Bae et al. [22] discuss the Solder Bump Maker (SBM), a novel paste material from solder powder and polymer resin, that manages to reduce the alignment requirements.

Concerning the die stacking strategy (regardless of conducting connections), we find works focusing either on the *arrangement* of each die level on the final SiP (stacking) or the *selection* of the die samples.

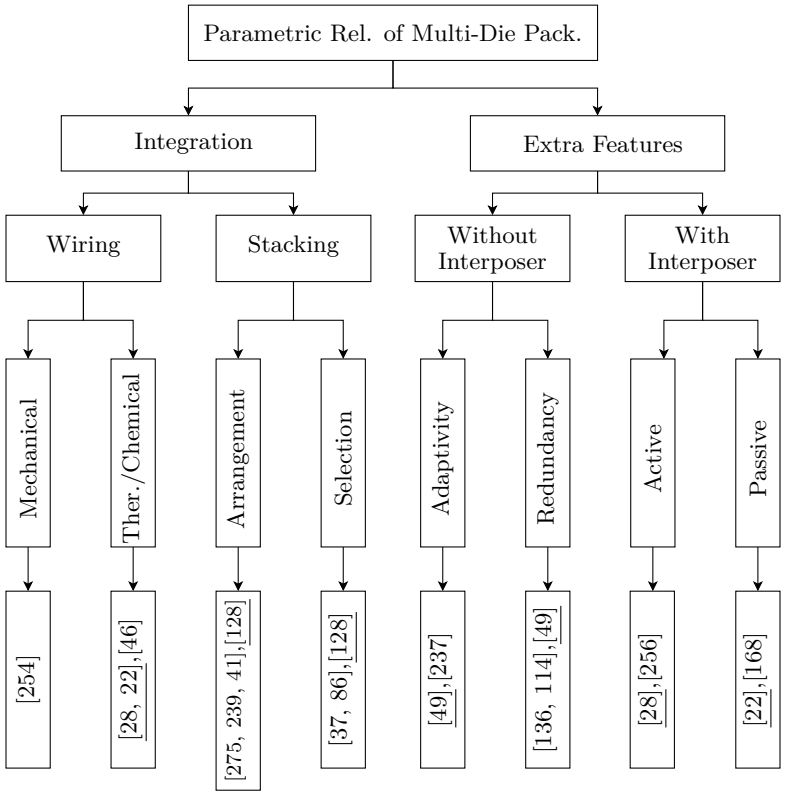


Figure 3.4: Classification Branch for Multi-Die Solutions.

Mapping 12. A reliability-constrained, die stacking methodology to mitigate process variability in 3D SiP is presented in previous work [275]. Another paper studies a die-to-wafer integration technology for high yield and performance concerning 3 dimensional stack applications [239]. Finally, prior art extensively discusses the ideal arrangement of the memory hierarchy within a 3D processor [41].

Mapping 13. Related patent discusses a voltage binning technique for 3D integrated circuit integration [37]. Certain strategies to improve the parametric yield of a 3D system after profiling the process variation of all dies and wafers involved, are mentioned in prior art [86].

Adding Extra Features & Components

We have covered, so far, parametric reliability solutions that target the issue of multi-die integration with respect to the interconnections and relative stacking strategies. Actually, integration procedure can also be facilitated through the addition of extra hardware components. We further split this branch based on whether an *interposer* affects the system's parametric reliability [151].

If such an interposer is ignored and is not assumed, we can find works that focus mainly on Through Silicon Vias (TSVs) and cavity cooling. In this case, we can divide this group of techniques between the ones utilizing *adaptivity* and those exploiting *redundancy* to counteract parametric violations.

Mapping 14. An adaptive scheme to detect and replace faulty vertical connections between stacked dies is introduced in a previous patent [49]. Another related work extensively discusses a thermal control principle for liquid-cooled 3D stacked architectures [237].

Mapping 15. A technology to provide TSV redundancy that improves signal propagation and reduces cross-talk is presented in a relative patent [136]. The deployment of a redundant TSV per block to replace defective TSVs to boost recovery rates in 3D integrated systems is discussed in prior art as well [114]. The patent by Camarota [49] is actually a hybrid paper since it also refers to the use of redundant vias in order to enhance parametric reliability of stacked dies.

When an interposer is involved, we can classify works that utilize *active* or *passive* interposers.

Mapping 16. The approach of the TSV Keep-Out-Zone, for an active silicon interposer is presented in previous work [28]. Another work proposes a wireless power transfer technology through an active silicon interposer that targets low-voltage applications in 3D integrated systems [256].

Mapping 17. Related work discusses the fabrication of top and bottom silicon interposers to stack SRAM chips in a SiP [22]. A passive silicon interposer with TSVs to successfully implement the heterogeneous stacked-silicon 3D integration is studied in another paper [168].

We identify four localized hybrids in the aforementioned tree branch. First of all, the paper of Banijamali et al. [28] discusses both the reliability monitoring of micro-bumps and the employment of passive interposers for 3D integration. Similarly, the paper of Bae et al. [22] studies a novel soldering material and a passive interposer as well. The work of Camarota [49] is also a localized hybrid paper. The paper of Juan et al. [128] discusses a novel selection strategy for 3D ICs and an optimal stacking order to mitigate performance variation.

3.5.3 Systems Across Packages: Hardware Solutions

The current Section studies parametric mitigation for systems across packages. Our focus is outside the semiconductor packaging; we inspect, instead, mitigation techniques for discrete component systems such as processors, storage peripheries or a memories. Defects and fault manifestations on a single node of these components could threaten parametric reliability of the overall system. Hence, in this case, the mitigation techniques are applied to deal with the consequences of a faulty component. We further split this branch between *hardware* and *software* solutions. The hardware solutions will be divided based on the node granularity; we distinguish therefore prior work into *intra-node* and *inter-node* techniques. This classification is presented in Figure 3.5.

Intra-Node

Intra-node methodologies target specific components and processing elements of the relevant system. They can be further split between *domain-specific* methodologies, i.e. the ones that focus solely on a specific domain (like for example the automotive), and *general purpose* ones. The latter can be split into the techniques that target the *processing elements* of the system and the ones referring to its *memory*.

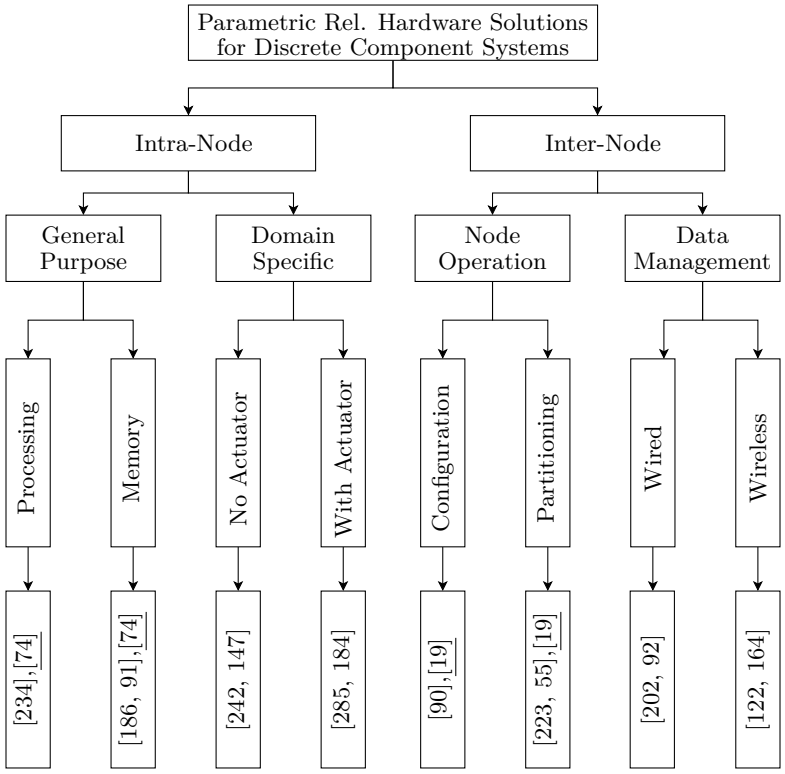


Figure 3.5: Classification Branch for Hardware Solutions on Discrete Component Systems.

Mapping 18. The Power8 processor from IBM contains an integrated On Chip Controller (OCC) [234]. The OCC, which is in fact a micro-controller for power and thermal management, monitors several system parameters and actuates changes in the processor's operating frequency, voltage, memory bandwidth and fan speed among others. Another work that discusses parametric reliability solutions for processing components is the white paper from Intel and Dell [74].

Mapping 19. The authors of a relevant work [186] utilize certain information from the data bus to develop an optimal, fine granularity refresh (FGR) technique for DRAMs. Another example is a real-time, in-memory, checkpointing scheme that improves the reliability of large-scale, parallel processing systems [91]. Certain hardware solutions in the memory are the subject of a technical report from Intel and Dell [74]

For the domain-specific methodologies, a relevant distinction can be made based on whether an *actuator* is embedded into that specific hardware component.

Mapping 20. The thermal headroom along with several cooling and packaging approaches for modern, on-board flight equipment are proposed in previous work [242]. Furthermore, heat convection aspects and copper thickness of the printed circuit board (PCB) for automotive, electronic control modules are studied on the paper or Kumar et al. [147].

Mapping 21. The parametric reliability of automotive components is the focus of the work from Valentin Von Tils [285]. Specifically, the author identifies relevant semiconductor solutions for the automotive domain and discusses examples of temperature-aware discrete components. Moreover, thermal management and other parametric reliability solutions (such as air-cooling friendly materials) for automotive electric traction-drive systems are proposed in another relevant work [184].

Inter-Node

Contrariwise to intra-node techniques, inter-node solutions are employed over the entire nodes of the system. A first basic distinction can be made between the techniques that focus on the *operation* of the system's set of nodes and the ones that affect the *data management* between the nodes. The former group can be further split between the methodologies targeting the *collective node configuration* and the techniques that refer to the *grouping or partitioning* of nodes.

Mapping 22. Energy variability among the set of nodes of a fully operational, supercomputer prototype is leveraged and DVFS actuations are implemented by Fraternali et al. [90] to save energy. To enable a reliable power distribution and system failures, the configuration of the system nodes is studied in another relevant work [19].

Mapping 23. Rasmussen's white paper [223] partitions the nodes of a relevant system into specific blanking panels so that the thermal profile inside a computer rack can be significantly improved. The grouping and partitioning of computer racks in internet data centers to develop an optimized air flow management system and achieve thermal control is also the focus of a related patent by Frederic Charron [55]. The partitioning of the system's nodes in data centers to improve reliability is also the focus of the work from Avelar [19].

Regarding the techniques affecting the data management across system nodes, they can be split based on whether the interconnection is *wired* or *wireless*.

Mapping 24. Deploying solid state disks to reduce the overhead of checkpoint writing is suggested in a related paper [202]. Another work studies previous failure rates computed from files and predicts the reliability and availability of the relevant storage architecture [92]. Then, they extrapolate the results to a petascale system and propose spares to achieve acceptable availability.

Mapping 25. In energy-bounded wireless sensor networks (WSNs), Jan et al. [122] study different routing techniques and detect the one achieving maximum lifetime. Another paper related to WSNs deals with energy efficiency and introduces the use of "redundant transmission on fusion routes with no acknowledgments" to deliver higher system reliability [164].

Localized hybrid papers can be identified in the current branch as well. A white paper from Intel and Dell [74] for instance, discusses parametric reliability mitigation solutions that focus on both the processing component and the memory. In addition, Avelar [19] reviews both the partitioning and configuration of nodes in data centers to improve availability and enable a reliable power distribution that avoids single points of failure, namely the manifestation of errors in certain system parts that can cause failure of the entire system.

3.5.4 Systems Across Packages: Software Solutions

Since digital systems have become significantly complex in the last years, software techniques for parametric reliability mitigation have started to be widely applicable as well. These techniques can be first split between *design/setup time* schemes and *runtime* ones. We show this classification in Figure 3.6.

Design/Setup Time

Before the execution of the software, schemes can be applied either under the *development* of the relevant software itself or as *admission policies* for the system's workload. Regarding the former group, we can distinguish methodologies that focus purely on the *algorithmic* part and techniques based on the *code implementation* to tackle with certain parametric reliability aspects.

Mapping 26. A specific data sampling algorithm to improve the reliability of wireless body area networks (WBANs) is proposed in prior art [24]. Moreover, to enhance reliability and performance of WBANs, Arrobo et al. [16] introduce a relevant algorithm for information coding.

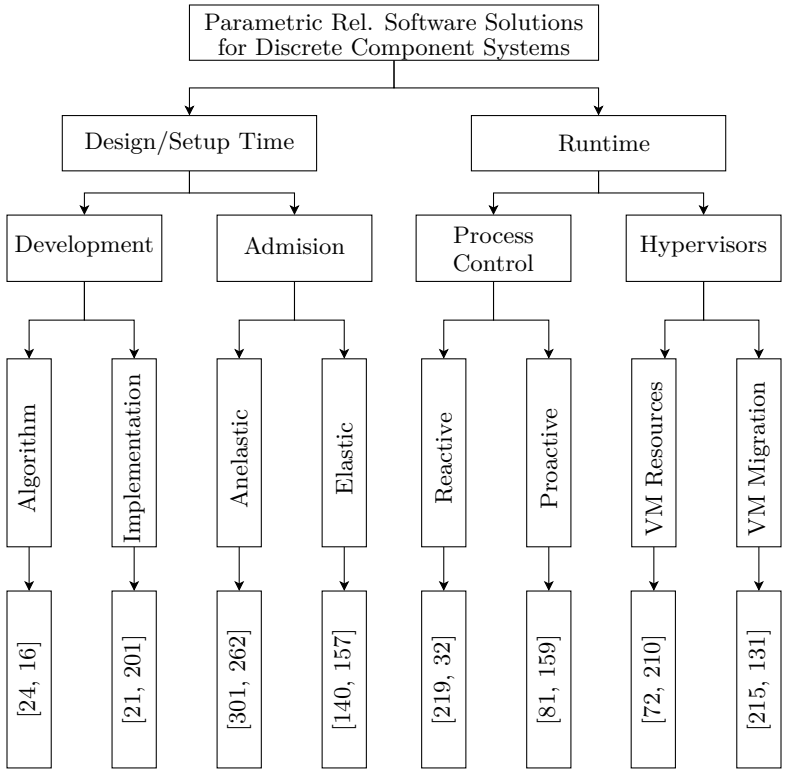


Figure 3.6: Classification Branch for Software Solutions on Discrete Component Systems.

Mapping 27. Checkpoint/Restart implementations on a message passing interface (MPI) to mitigate system performance variability is the study of relevant prior art [21]. To increase performance dependability and reduce the overall effect of OS interference on large-scale parallel applications and MPI communication, Oral et al. [201] suggest to accumulate all OS noise-related sources into a specific core.

Solutions relevant to admission policies can be further distinguished between *anelastic* and *elastic* ones; the elasticity of the services can be defined as the dynamically changing resource requirements that depend "on the varying number of users and patterns of requests" [140].

Mapping 28. A fixed priority scheduling policy for multi-node systems under strict latency constraints is the subject of previous work [301]. Another work studies the schedulability of I/O devices in hard real-time systems under energy-optimal operation [262].

Mapping 29. An admission control policy for elastic cloud services is proposed by Konstanteli et al. [140]. A task scheduling algorithm along with proper DVFS actuations on the mobile cloud computing environment is studied in another prior work [157].

Runtime

These techniques are activated at runtime, during the software execution. They can be distinguished between *OS-related* solutions and *hypervisor-like* schemes. The former group can be further split between *reactive* and *proactive* schemes.

Mapping 30. In recent work [219], features of the operating system are enabled through the "Thread Tranquilizer" to dynamically reduce performance variation of the application workload that runs on a 24-core server. A reactive, runtime resource management scheme using Linux control groups is also studied in another recent article [32]. The main goal of the authors is to ensure timing deadlines while respecting power constraints.

Mapping 31. El-Sayed et al. [81] focus on a checkpoint/restart mechanism realization and first evaluate its performance and energy overheads. Then, the move forward and suggest simple formulas that optimize checkpoint scheduling by adapting to the failure rates of HCP systems and thus achieve high energy gains. An adaptive I/O method to mitigate performance variation due to the concurrent use of petascale storage systems by multiple users is proposed in the work of Lofstead et al. [159].

On the branch related to hypervisor-like solutions, we typically find software schemes dealing with virtual machine (VM) instances. Therefore, we can distinguish these schemes between the ones handling the *resource provision* of VMs and the others engaging in VM *migration*.

Mapping 32. To guarantee efficient workload VM allocation and optimize application performance on co-hosted VMs, Dawoud et al. implement a specific QoS controller [72]. Another work [210], introduces the hypervisor-level framework "Hyper Tap" to efficiently support VM reliability monitoring.

Mapping 33. To ensure server system dependability in a proactive manner, the authors of a prior work propose the "concept of anticipatory virtual machine migration that proactively moves computation away from faulty or suspicious machines" [215]. In addition, to reduce network bandwidth utilization and achieve a reliable and efficient VM migration, Kashyap et al. [131] introduce the Reliable Lazy Copy (RLC) approach.

The overall classification framework we have created is illustrated in Figure 3.7.

3.5.5 Classifying Globally Distributed Hybrids

An important feature of our binary classification is the ability to decompose prior art samples to their primitives. This enables identification of hybrid samples, i.e. those that can be mapped to more than two leaves. So far, we have addressed localized hybrids. Table 3.1 contains widely distributed hybrids, spanning across distant leaves (i.e. across Subsections). From such papers, one can draw interesting observations:

- Certain domains that are quite new in terms of commercial readiness (in comparison to more "mature" domains like single-die integration) feature highly localized hybrids. This is supported by the degree of hybrid dispersion for the multi-die packages branch (Subsection 3.5.2). We have identified four localized hybrid papers which are mapped there

and are concentrated in that very region, not spanning to more distant domains. It is reasonable to expect that as multi-die packages are further developed, more diverse research (i.e. more global hybrids) will appear in that domain.

- It is noteworthy that many *mitigation techniques targeting process variability are valid hybrid papers*. The hybrid nature lies in first receiving feedback from low-level hardware and then effectively tuning these adaptive, variability-aware techniques at higher abstraction layers (usually at the system level). A case can be made for the papers of Qureshi et al. [220] and Bathen et al. [29]. In both cases, a degree of semiconductor variability (variable retention times and power variability, respectively) is exposed for improved performance and energy budget.

- Finally, it is also reasonable to expect white papers and technical reports from *hardware vendors* to appear in many leaves of our classification. Even though detailed technical information is not disclosed, such sources provide a very good bird's eye view of parametric reliability solutions that are available in commercial RAS-oriented products. A typical example is Intel's technology journal on "Managing Process Variation in 45nm CMOS Technology" [69]. This report studies thoroughly mitigation techniques

Paper	**Brief Description**	**Classification**
[182]	Report from IBM on available RAS features	Mappings 18, 23, and 32
[1]	Oracle white paper on SPARC Mainframe-Class infastructure and its capabilities	Mappings 18, 23, 32, and 33
[70]	Intel white paper on the RAS features of a specific processor family	Mappings 18 and 32
[220]	Adaptive DRAM refresh technique leveraging on variable retention times	Mappings 19 and 27
[29]	Framework of policies for variability-aware virtualization of the memory hierarchy	Mappings 19 and 27
[216]	Leveraging performance variability to save energy by frequency scaling upon MPI collectives	Mappings 22 and 27

Table 3.1: Hybrid papers instantiated in our binary classification and components thereof

adressing the size of the transistors, the introduction of novel device materials, patterning techniques and placement and routing optimizations. The white paper from AMD regarding power management and performance variation is another such example [13].

3.5.6 Classification Extensibility & Reusability

Another unique feature of our classification is its systematic extensibility. Apart from mapping new papers to an existing classification, additional splits can be introduced depending on the intended focus. For the sake of brevity, we substantiate the extensibility of our classification by elaborating the sub-domain of single die solutions for parametric reliability (Subsection 3.5.1 – Figure 3.1). In Figure 3.8, we illustrate remapping of previously mapped papers and the addition of new ones.

Hardware design optimization is further split between *circuit-level* techniques which imply mostly wire/transistor sizing and physical design techniques which impact placement and routing (*physical design*). In the former category, we find aging-aware sizing of standard cells [130] and lithography-aware sizing of SRAM cells [127]. Design enhancements can be distinguished between adding hardware, which implies here some form of *redundancy* techniques (round-robin redundant SRAM deactivation [251]), and changing the existing hardware to allow more *adaptivity* (as in the case of tunable sense amplifiers [66]).

Regarding the software strategies, we can further split this group between the techniques that focus on the *prediction and prevention* of parametric reliability violations and the ones deploying a type of *tolerance* for the designed system implementation, to temper the impact of reliability threats to the system's operation. The work of Rahimi et al. [221] belongs in the former category since it involves the use of an aging-aware compiler to generate code that balance the workload distribution thus preventing possible wearout phenomena. In addition, a runtime software that prevents the execution of problematic code sequences that can cause recurring voltage supply variations is presented in relevant work [109]. On the contrary, the work of Mittal et al. [183] discusses a technique that mitigates performance variation by binding certain conscecutive tasks to chained resources and minimizing therefore the overall latency of future scheduling decisions. Another paper belonging in this category is the work of Baek et al. [23] where the "Green" program is proposed to approximate expensive functions and trade-off certain quality of service requirements for improvements in energy consumption.

On the process technology front, the material side is split between *interconnects* (capping layer for Cu interconnects [118]) and *devices* (process optimization

for gate leakage and BTI [119]). The lithography side is split between *single patterning* with advanced wavelengths and *multiple patterning* (hybrid paper by Joshi et al. [127]). Previously instantiated lithography papers (at higher levels of the tree) are remapped accordingly [296, 214].

The before-market side, of the post-fabrication domain contains a variety of testing and tuning techniques. Non-intrusive techniques are split between *switching* (where the fabricated die is dynamically operating, as in the case of variability profiling for optimal DVFS [111]) and *quiescent* (where techniques are applied on a die that is operationally static, as in the case of I_{ddq} binning). Intrusive techniques for parametric reliability are further split between *contact-based* (e.g. current pulse injection [84]) and *contactless* [94].

Finally, in the after-market side of the tree, reactive and proactive techniques have been elaborated. The reactive side is split based on the scope of the technique in terms of power islands. For instance, in previous work, a PID-controlled DVFS technique for dependable performance has been proposed [229], whereas Nasir et al. deal with efficient, fine-grain voltage regulation [190]. The hybrid paper of Herbert et al. deals with variability-aware DVFS per core/memory module [111]. Proactive techniques have been split between *model-based* and *monitor-based*. The work of Zoni and Fornaciari is instantiated in both leaves, since it provides sensor-wise and sensorless mitigation of BTI in network-on-chip buffers [309]. With the above extension, we substantiate both the reusability of our classification framework, by adding new papers, and its extensibility, by inducing further splits to the binary tree.

3.6 Reliability Modeling

In Chapter 2 we studied the physical principles behind the most prominent failure mechanisms while in the previous Section 3.5, we created a top-down classification of prior art regarding current mitigation techniques for parametric reliability. The computer engineering community, during the last several years, has also developed models to capture, describe and estimate the impact of these failure mechanisms on the overall system performance. By studying the literature on prior art, we can find several implementations that model these mechanisms either on a device or a system/architecture level. The former approaches achieve a very detailed and accurate description of these phenomena however, when it comes to simulating large circuit inventories and over long time strides, they often require lengthy and computationally intensive simulations.

Briefly, regarding process variation we can find respective models describing the impact of LER and LWR to the circuit's layout [27] and its operation [67].

Furthermore, an analytical description on the variation of V_T due to RDF can be found in previous papers [145, 156]. Referring to the gradual wearout of the gate oxide, a surplus of work can be found on BTI and HCI phenomena [270, 129, 281]. Recent work suggests that the latter is more pronounced in the subthreshold operation of low-power CMOS circuits [169]. For BTI, a bulk of experimental data verifies the shift from the previous, reaction-diffusion model to a new atomistic model [104] capturing the phenomenon in a more efficient manner. This new approach is introducing a stochastic part on BTI modeling and presenting workload memory. Such device-level models come at the cost of elevated simulation times, while computation complexity is tightly coupled to the form of signal activity [233]. As a result, a cycle-by-cycle modeling of variability can be computationally prohibitive, especially on complex circuit inventories. This problem is partially alleviated through the framework presented in previous work [233] (see Appendix Section A.2). There, signal activity is compressed and through pseudo-transient simulations, the functional yield of an SRAM cell is estimated at specific, user-defined time instances.

Many simulation tools in the industry use the Statistical Static Timing Analysis (SSTA) methodology [123, 14, 40]. However, SSTA methodology still faces certain challenges, like for example the incapability to include the delay correlation of different circuit paths and spatial correlations of process variation and to capture all the workload-dependent effects [93]. Another way of characterizing variability effects on large circuit inventories could be the worst-case timing analysis [89] where worst process and operating conditions are assumed. This methodology is extremely fast and produces safe yet rather pessimistic estimations in the form of guardbands. Another work [278] understands the bottleneck of further increasing guardbands to deal with the threat of variability. Hence, the authors focus on shaving design margins while studying long-term as well as instantaneous BTI effects. Therefore, a trade-off exists and engineers can choose either to accurately model the reliability threats and use time-consuming simulation tools or employ more efficient approaches by paying however a price in accuracy.

3.7 Conclusions

In the current Chapter, we have focused on the prior art of parametric reliability mitigation. We have also briefly discussed the prevailing mitigation techniques for functional reliability including error correcting codes, modular redundancy and checkpoint/restart (rollback) mechanisms. Specifically, we have performed a top-down, orthogonal classification approach with complementary splits to categorize previous work samples regarding parametric reliability mitigation,

into representative groups and eventually form a binary tree. The methodology to build the aforementioned framework is studied in Section 3.2. Former survey papers attempting similar categorizations are mentioned in Section 3.4 while the complete classification framework is extensively analysed in Section 3.5. Finally, specific aspects of current reliability modeling approaches and tools are briefly mentioned in Section 3.6.

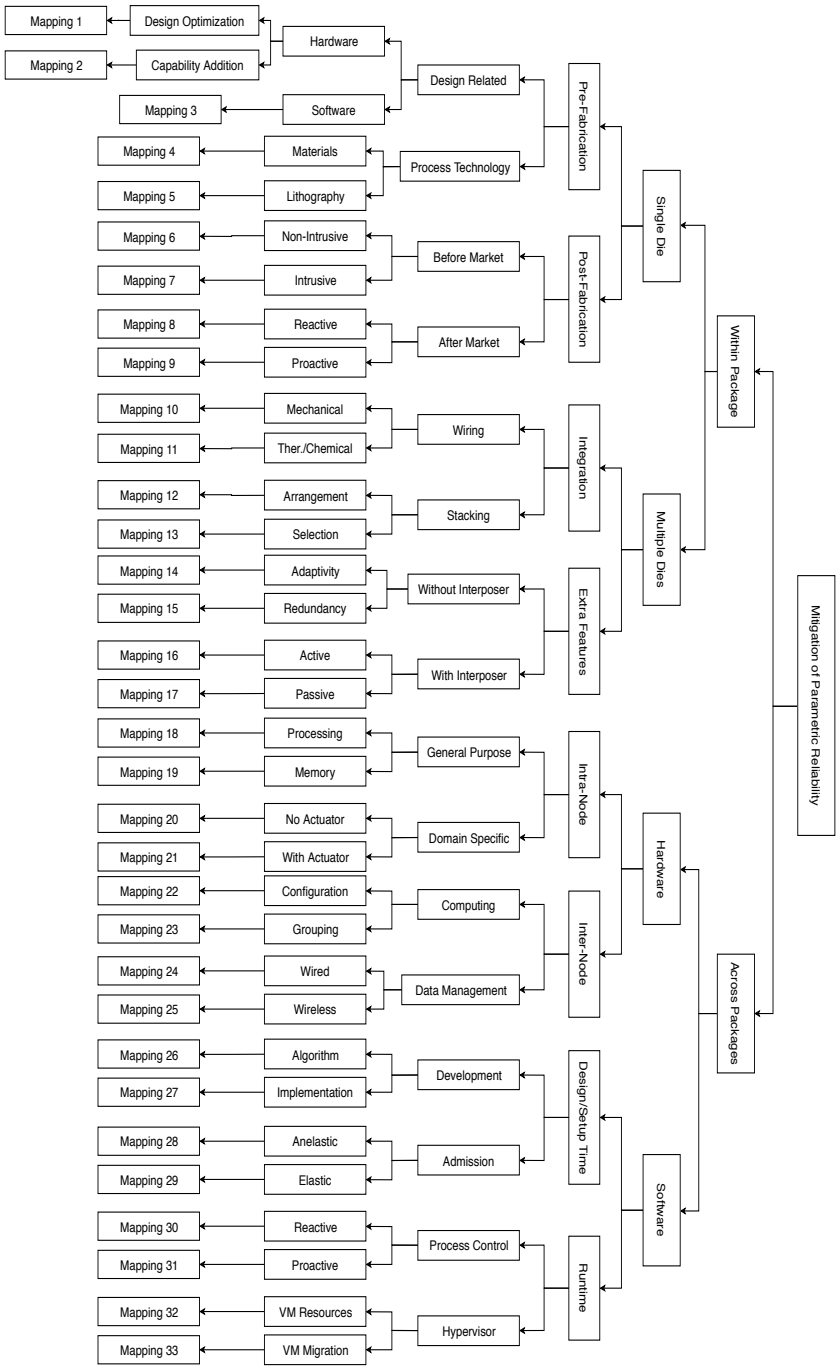


Figure 3.7: Overall Classification Framework for Parametric Reliability Migitation.

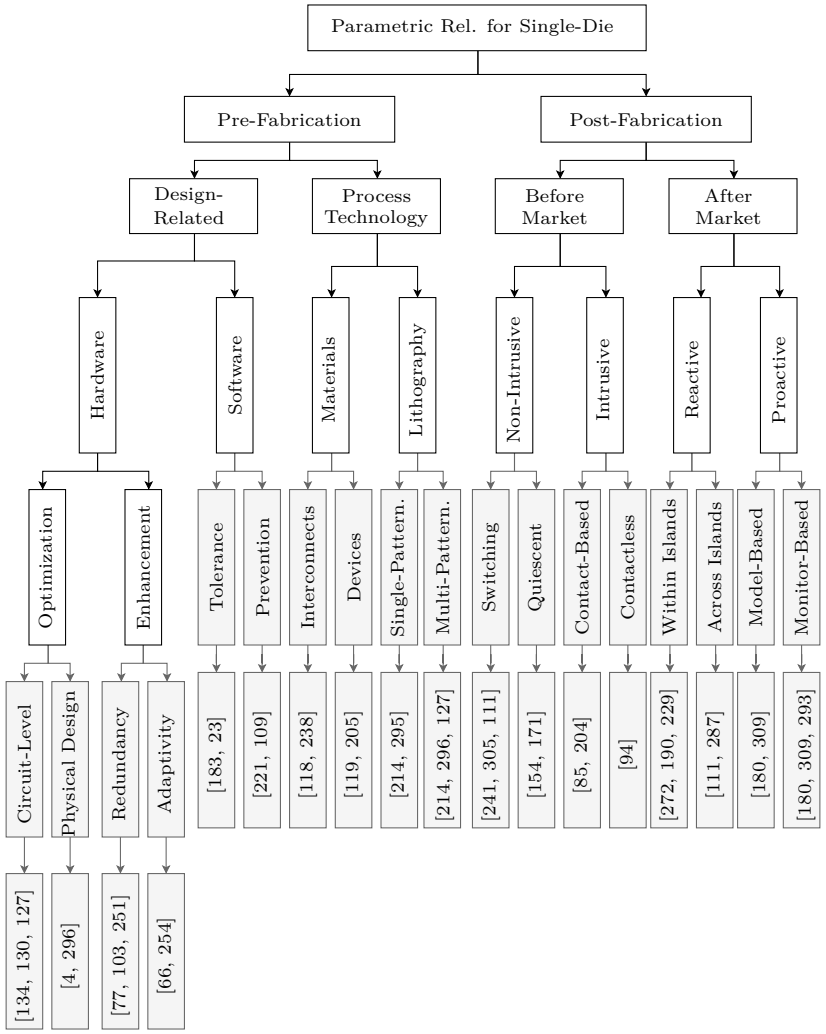


Figure 3.8: Extending the classification tree with extra splits for the single-die branch and re-mapping of prior art. New splits are shown in the gray boxes while the former ones in black. Former papers are classified accordingly.

Chapter 4

Estimating Failure Probability Using MPFP

4.1 Introduction

A broad categorization of techniques targeting parametric reliability violations was presented earlier in Chapter 3 while a brief description of the failure mechanisms triggering these violations has been studied in Chapter 2. To help them determine and develop fitting mitigation schemes, tailored to a specific digital system, engineers usually perform reliability analysis and deploy certain software tools that estimate the impact of the failure mechanisms on the IC design. Towards this direction, in the current Chapter, we develop a related tool that captures the dominant failure mechanisms and estimates the failure probability of the circuit under study in an accurate and efficient manner, utilizing the most probable failure point (MPFP) methodology. First, we will discuss other reliability analysis approaches, that are based either on static timing analysis or some type of Monte-Carlo simulations. In Section 4.4 we will extensively describe the MPFP methodology and highlight its core advantages, focusing especially on the SRAM cell since memories are in general heavily susceptible to variation and degradation. Finally, we will develop a related simulation-based software tool, using MPFP, to estimate the robustness of the SRAM buffers of a Network on Chip (NoC).

4.2 Functional and Timing Verification of the Design

When designing an electronic chip, typically in the Register Transfer Level (RTL) abstraction layer, logic simulations are applied across the circuitry. These type of simulations verify the functionality of the design and allow designers and verification engineers to identify the numerous simple design errors that occur in the initial version of every design. Since modern ICs consist of billion devices and perform a wide variety of functionalities, this *logic verification* is a complex task, and requires a considerable amount of time and effort. To efficiently perform logic simulations, coding test-benches are generated, exploring the circuit's functional I/O behaviour after all the possible (or specific and representative) input vectors are applied. ModelSim from Mentor Graphics [178] is such a tool, widely used for field-programmable gate array (FPGA) designs. The same company later introduced QuestaSim which provided high performance Verilog and SystemVerilog simulations [177]; other commercial as well as open-source simulators exist. Approaches for logic verification are discussed in the relevant Chapter from Thornton et al. [166].

After the functionality of the design has been verified, engineers use *timing analysis* to determine whether the timing constraints imposed by the components or interfaces are respected. This analysis is usually performed later in the development cycle and can be applied at the gate or the device level; certain tools can also perform timing analysis at the physical design or even post-layout, after placement and routing. In pre-layout simulations, a circuit schematic is built to include the relevant elements of the design (gates, transistors, wires etc.) while their characteristics are translated into mathematical models. In post-layout simulations however, physical information is extracted from the routed chip or board as well. In general, most simulation work should take place at the pre-layout phase since design changes tend to be costlier further into the design cycle. The goal of post-layout simulations is to finally confirm the compliance with determined timing constraints. Therefore, both logic verification and timing analysis are vital to understanding the IC design, and designing reliable and successful electronic products. The design flow is illustrated in Figure 4.1.

In principle, through timing analysis engineers determine whether all signals arrive at specific points of the design within their prescribed timing intervals. To achieve this, designers can select to use *static* or *statistical static timing analysis* (STA/SSTA) techniques to compute the timing behaviour of the circuitry by propagating gate or device information through the netlist and without performing simulations of the full circuit. Alternatively, one can choose to simulate the complete circuit and perform *dynamic timing analysis* (DTA),

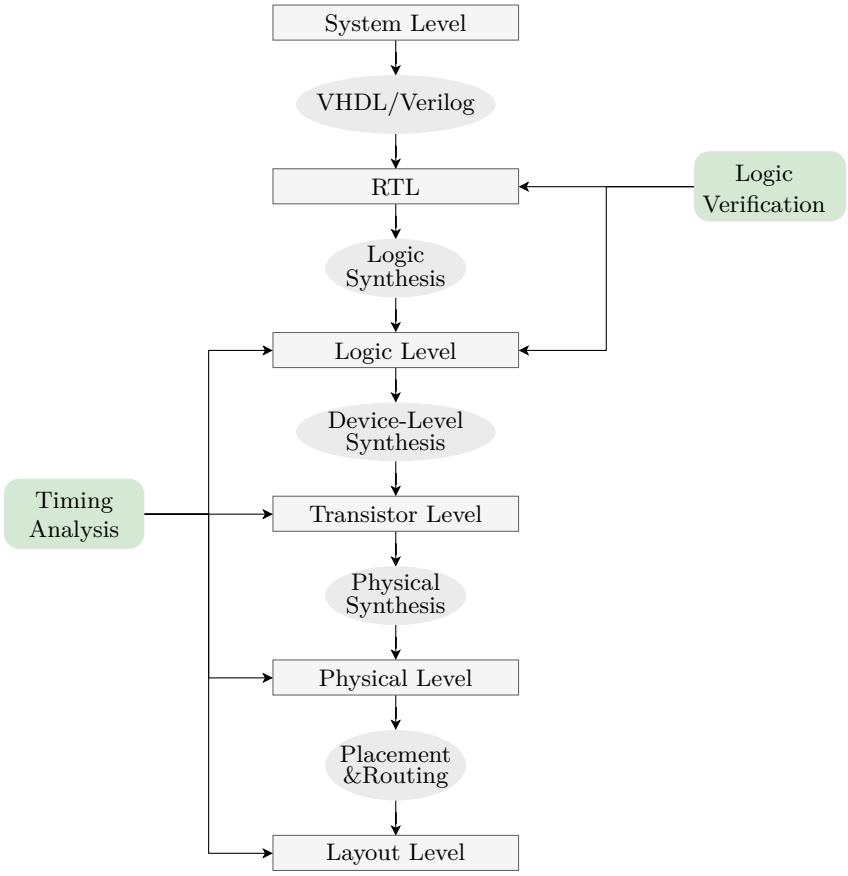


Figure 4.1: CMOS design flow.

using relevant input stimuli and extracting timing information along with other interesting data such as power measurements, waveforms and so on. In this case however, the complete timing analysis of all netlist's nodes usually requires some type of exhaustive, *Monte-Carlo*-like experiments and hence significant (and even prohibitive) simulation times for large SoC circuit netlists.

4.2.1 Static Timing Analysis

Static timing analysis performs an exhaustive and complete timing characterization of the circuit design, estimating the propagation delay over all the design's elements. It is defined as static since the analysis results are independent from

the data values applied at the input pins. The primary task of STA is to consider the propagation delays of all the circuit's components, identify the critical path and determine maximum clock speed while respecting the design's timing constraints. It is important to note that in STA, the entire circuit is analyzed only once and the timing checks are carried out for all the possible paths and scenarios of the design.

The relevant timing information of all the components in the design are stored in library cell descriptions. In fact, standard cell library also contains data irrelevant to timing characteristics such as area, functionality, geometry etc. The STA tool uses this type of information and through respective timing models – that can be simple linear ones or even complex, non-linear and advanced– estimates the propagation delays of complex circuit components and evaluates eventually the design's timing behaviour. Therefore, standard libraries are crucial for STA and are usually provided by the fabricating companies to the electronic design automation (EDA) vendors.

To account for the different operating conditions, STA is performed at different *corners* for Process, Voltage and Temperature (PVT) [36]. Specifically, STA recognizes three corners for process variation, namely slow, typical and fast; evidently, slow and fast corners are the extreme corners when addressing process variability in STA. Similarly, the corners for voltage supply and temperature variations are minimum, nominal and maximum. The operating conditions used in STA are combinations of the aforementioned corners; for example, the *Worst-case Slow* operating condition uses the slow corner regarding process variation, maximum temperature corner and minimum voltage supply. On the contrary, the *Typical* operating condition assumes typical process variation and simulates the nominal corners for voltage and temperature. Standard libraries contain information regarding all respective corners. In fact, modern STA tools usually contain more design corners, like for example a distinction between nMOS and pMOS devices or corners for the interconnects [292]. PrimeTime from Synopsys performs STA analysis for designs at the gate level [263] while NanoTime (again from Synopsys) is used for STA at the transistor level [264] and provides results of notable accuracy; nevertheless, this transistor-level simulation requires a very detailed description of the intellectual property (IP) block which is not always available. When there is not accurate view of the IP design, the use of transistor-level simulators like NanoTime is prohibitive.

4.2.2 Statistical Static Timing Analysis

With the aggressive downscaling of device and interconnect dimensions, process variability and failure mechanisms in general are aggravated. Therefore, the need

to design robust ICs while taking into account the aforementioned phenomena is crucial. Traditionally, STA methodology could verify the timing constraints of a design through different design corners, especially the *Worst-case Slow* one. In modern devices however, the principal sources for performance variability have multiplied. Hence, a complete corner analysis in this case could require numerous STA runs –from 2^7 up to 2^{20} [258]– and significant computation times for large SoCs. Again, a worst-case analysis could be applied, assuming worst-case conditions for all possible variability sources; nevertheless, such a scenario proves rather pessimistic, does not take into account the actual correlations between these sources and finally, suggests the use of overly large safety margins, generating too low timing and power gains when migrating to ultra-scaled technology nodes.

To overcome the above bottleneck, SSTA methodology was introduced. In this type of analysis, the statistics of the variability sources are taken into consideration, including their probability distributions, variances and co-variances. Hence, instead of fixed delay values per corner, the delay is described through its statistical distribution. Compared to STA, SSTA estimation is more costly because a large amount of information is now needed. Furthermore, this type of information is currently difficult to obtain since accurate process data, correlating propagation delay to variability sources, requires extensive experiments and substantial effort from fab companies.

Several SSTA approaches have been discussed so far [6, 113, 125]. Nevertheless, the main problem of this technique still exists: an accurate mathematical formulation that describes the propagation delay as a function of the process parameters ($t_{pd} = t_{pd,nom} + f(x_1, x_2, ..., x_n)$) – x_i is considered a random variable– is essential and not always available. Moreover, the correlations are not correctly modeled in the statistical distributions that are employed today. This leads to overly large global margins for ultra-scaled technology nodes.

4.2.3 Dynamic Timing Analysis

In contrast to STA and SSTA, DTA uses input stimuli to verify the timing constraints of the circuit design. This type of analysis however, considers only the specific input vectors applied each time hence, in order to increase the quality of DTA, engineers need to expand the input vector. Evidently, a complete DTA characterization of the circuit's timing response usually requires an exhaustive number of simulations. Nevertheless, it is more analytic compared to static analysis, containing relevant information such as voltage and current waveforms of the output nodes. Furthermore, DTA is the only method to analyze asynchronous designs since static analysis cannot be applied in these

cases. It is also important to note that typically, DTA can also be employed for functional verification. Gate-level DTA can be performed with ModelSim [178] while the SPICE framework is widely-used for transistor-level DTA [268]. Tools for DTA at the physical level include the Virtuoso Layout Suite from Cadence [48] and Calibre from Mentor Graphics [176].

When the dependence of propagation delay to process variation parameters and other device characteristics (like for example ΔV_{th} caused by BTI/HCI) is unknown, STA/SSTA provides results of questionable accuracy. In this case, DTA is often used through *Monte-Carlo*-like experiments to achieve a proximate estimation of the aforementioned dependence[1]. The standard Monte-Carlo (MC) methodology [224, 143] relies on generating a large number of random sample netlists and after simulations, the function f under study – in this case, the function describing the relationship between propagation delay and variability and reliability parameters – is approximated through a second f' function, based on the strong law of large numbers; the more experiments are conducted, the more accurate the approximation, so that $f' \to f$ when $N \to \infty$. Evidently, MC is a strong tool for circuit designers and reliability engineers when the cost of elevated simulation times is accepted.

An infinite number of samples for the MC methodology is impossible to be simulated. Nevertheless, to achieve accurate approximations of the f function, large sample populations are normally required and considered. In fact, the bottleneck of the general MC methodology is its slow convergence rate ($O(\sqrt{N})$). This problem can be partly alleviated through certain approaches such as Importance Sampling, Quasi MC [31] and corner-based Monte Carlo [174]. These statistical methods may enable a faster convergence rate but only under certain conditions. Another way to reduce MC simulation time is to use only a small number of samples and then "fit" the resulting f' to a known distribution (often the Gaussian); again, the fitting procedure inevitably introduces errors. It is clear therefore that approximating f and performing a complete DTA demands extensive simulation times where large sample populations are simulated to capture rare events/cases.

4.3 MPFP Premises

The *Most Probable Failure Point* (MPFP) search technique has recently been proposed for failure probability estimations [133], as an alternative to the typical MC method. While this methodology was first utilized in the mechanical and

[1] Actually, combinations of Monte Carlo algorithms with SSTA methodology have also been discussed [280, 64].

aerospace domains [108, 294], it is now employed in many works concerning reliability and timing analysis of circuit designs [222, 153, 231]. We have already explained the main problem of the typical MC methodology: since netlist samples are randomly generated, the simulation of rare samples and the capturing of events with low probability of occurrence requires large sample populations and therefore, extensive simulation times. On the contrary, the MPFP technique allows the generation of a more representative sampling population because, in this case, samples are not randomly created. Finally, through MPFP, the identification of rare events and faults with low probability of occurrence requires less simulation iterations.

Next, we will estimate the failure probability (P_F) of a finFET-based 6T SRAM cell[2], utilizing the MPFP methodology. Generally, P_F hints to the parametric yield loss (YL) of an embedded SRAM; YL is the product of the failure probability of an SRAM cell and the overall number N of cells in the SRAM, as shown in Equation 4.1 [133].

$$YL \approx P_F \times N \tag{4.1}$$

In this study, variability sources include the dominant, time-zero variation phenomenon RDF and time-dependent wearout caused by BTI. Instead of performing a typical DTA analysis we will focus on the hold stability of the SRAM cell, studying the cell's *Static Noise Margin* (SNM) to present an illustration of the MPFP approach. The SNM for the hold operation describes the retention ability of the cell and shows the tolerance of the cell to noise; in fact, the SNM value represents the maximum noise level the cell can tolerate for the hold operation [244]. In principle, the SNM is quantified graphically through the butterfly curve. Specifically, it is estimated as the side length of the maximum square that can

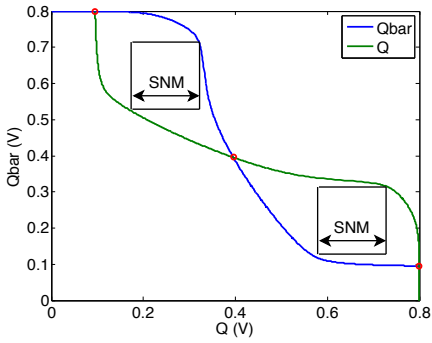


Figure 4.2: Example of a butterfly curve for a finFET-based, 10 nm 6T SRAM cell. SNM hold is extracted graphically.

[2]A finFET (or 3D) transistor is a multigate device where the gate is wrapped around the channel, allowing better controllability of the channel, faster switching times and higher current densities among others [297]. This is in contrast to planar MOSFETs, where the gate is placed at the top of the channel.

be fitted inside the lobes, as in Figure 4.2[3]. Other figures of merit, like the write stability of the cell, can also be considered as an extention for future work.

4.3.1 MPFP Methodology and Definitions

The core concept of our MPFP tool is based on the following definitions[4]:

- Aging wearout and process variability, alter certain device parameters –in this case study V_{th}– which are described by a vector $\mathbf{x}$. The vector size of $\mathbf{x}$ depends on the netlist and specifically, on the number of the transistors involved. For example, in the 6T SRAM cell, $\mathbf{x} \in R^6$ [5].
- For the metric that we examine (the SNM in this case) a y value corresponds to every $\mathbf{x}$ point. We show this through the function $y(\mathbf{x})$. The exact mathematical description of this function however is extremely difficult to find. Only after simulation and experimental results, a y_A value is matched to a certain $\mathbf{x}_A$ vector.
- We assume a certain margin Y for the metric value we examine. Depending on the metric under study, the failure criterion can be either $y < Y$ or $y > Y$. In our case, a sample is considered failed when $y < Y$ where Y is a margin value for SNM metric. Nevertheless, assuming the metric under study is the delay of a path, it is self-evident that the failure criterion switches to $y > Y$, considering maximum delay Y.
- Then we consider a certain margin Y for the metric under study and the failure criterion can either be $y < Y$ or $y > Y$. In this case, the failure criterion is $y < Y$ hence, the SNM value y_N of a vector $\mathbf{x}_N$ is considered a failed sample when $y_N < Y$. Other studies, that focus on the delay of a critical path for example, can have a different criterion; if Y is the maximum accepted delay of the path, then a failed sample is when $y_N > Y$.
- Finally, we sample the $\mathbf{x}$ space - either in an exhaustive manner or through an optimization method - and we focus on a) finding the most probable

[3]Stability of a cell can be related to other figures of merit as well. For instance, relevant work [179] develops a methodology based on a prediction tool for the estimation of read SNM, that indicates the robustness of the cell during its read operation.

[4]These definitions are not restrictive. They can also be applied beyond the SRAM cells or the memory periphery in general. The work of Rodopoulos et al. [232] for example, employs MPFP to perform reliability analysis and DTA for basic logic gates.

[5]$\mathbf{x}$ can be even more complex if we take into consideration the variability of the wires in the BL/WL of a memory cell, especially for large SRAM blocks. However this study is beyond the scope of our work.

failure point $\mathbf{x}_{MPFP}$, which is the point that fails the criterion and is most likely to appear and b) isolating the F space that contains all the failed $\mathbf{x}_N$ points; evidently $\mathbf{x}_{MPFP} \in F$. In fact, in order to estimate F space it is important to first identify the the most probable failure point. Then, the failure probability is simply $P(\mathbf{x} \in F)$.

Several methods of MPFP analysis can be seen in prior art. The work of Khalil et al [133] for example, exhaustively samples the space to obtain a highly accurate estimation of F and the failure probability. Nevertheless, this method requires significant simulation time. Realizing that the number of the netlist's devices is directly coupled to the computational complexity of the MPFP methodology, certain optimization techniques that alleviate this complexity exist. Related approaches, that study the 6T SRAM cell, assume that the operation of some transistors is negligible compared to others. Hence, instead of sampling the complete 6-D space drawn from all devices, they choose to reduce the dimensions and neglect the variance of some transistors. A previous paper for instance [135] considers a 1-D approximation while the work of Rodopoulos et al. [231] realizes that the operation of WL devices in the SRAM's hold ability needs not be examined and minimizes the space to only four dimensions. Other efficient sampling algorithms should also be noted [222].

In order to estimate the failure probability we should then identify the most probable failure point. Previous works utilized Equation 4.2 that is representing an optimization problem. Ideally the MPFP lies on the failure boundary and gives us an idea of the F space; a proper estimation of the F space allows to accurately calculate P_F.

$$\mathbf{max}\left\{ P_{\mathsf{fail}} = \prod_{i=0}^{N-1} P(|\Delta V_{th,i}| \geq x_i) \right\} \quad \text{so that } y(\mathbf{x}) < Y. \tag{4.2}$$

We have already mentioned that failure probability is $P(\mathbf{x} \in F)$ and regardless of our estimations on the MPFP location (the boundary of the F space), it is crucial to have a solid indication of the F space. This problem is discussed in the paper of Rodopoulos et al. [232], where the authors focus on a simple inverter and differentiate their implementation from previous approaches, adopting a x^2 distribution and drawing hypersphere-like shapes around F. Interestingly, a hypersphere sampling method is also described in another previous work by Ranat et al. [222] nevertheless, this methodology only studies the identification of the most probable failure point and does not look into the exact shape of the F space.

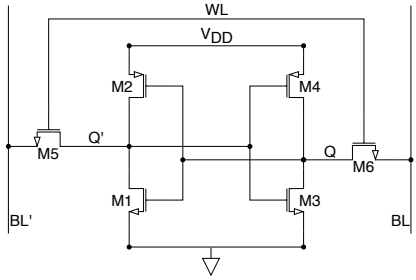


Figure 4.3: The typical 6T SRAM cell.

4.4 Utilizing MPFP – Estimating the SNM

4.4.1 Modeling the Failure Mechanisms

Figure 4.3 shows the design of a typical 6T SRAM cell. In our case study, we have chosen to model variations in all 6 transistors therefore we examine a subset of the 6D space. While this approach results in elevated simulation times, we aim to achieve final P_F estimations of significant accuracy. We focus on a finFET-based SRAM cell at 10 nm nodes described by PTM modelcards [50]. We study P_F of the cell after 3 years of continuous operation, which is a typical lifetime assumption for electronics. We also model aging wearout, accounting for the BTI effect (since this phenomenon is the dominant failure mechanism) and describe the mean threshold voltage shift through the Equation 4.3, drawn from relevant work that regards finFET devices [146]. This formulation is evaluated based on wafer-level, extended Measure-Stress-Measure (eMSM) measurements while the parameters α and γ of our technology nodes are also estimated.

$$\langle \Delta V_{th,TD}(t) \rangle \cong A t^a E_{ox}^{\gamma} \tag{4.3}$$

A is the fitting coefficient, E_{ox} is the electric field applied between the gate and the substrate and t represents the stress duration. Regarding $\sigma_{Vth,TOT}$, it is depending both on time-zero and time-dependent variability ($\sigma_{\Delta Vth,TOT} = \sigma_{\Delta Vth,0} + \sigma_{\Delta Vth,TD}$) [290]. Equation 4.4 shows the connection between time-zero and time-dependent variability and we can identify how pronounced process variation phenomena aggravate the time-dependent variation.

$$\sigma_{\Delta Vth,TD}(t) = \sigma_{\Delta Vth,0} \sqrt{\frac{< \Delta V_{th,TD}(t) >}{100mV}} \tag{4.4}$$

$\sigma_{\Delta Vth,0}$ represents the threshold voltage shift variability at time-zero due to process variation. For process variation we will again focus on the dominant phenomenon that is RDF. Hence, Equation 2.1 (Section 2.2.1) describes $\sigma_{\Delta Vth,0}$. Finally, the mathematical formulation to describe the variability of V_{th} is given in Equation 4.5.

$$\sigma_{Vth,TOT}(t) = \left(\sqrt{\frac{\langle V_{th,TD}(t) \rangle}{100mV}} + 1 \right) \sigma_{Vth,0} \tag{4.5}$$

4.4.2 Employing the MPFP

To locate the MPFP and therefore estimate the failure probability, we first need to explore the SNM space and identify the point $y(\mathbf{x})$ with the highest SNM value. For this reason, we use the Cadence Spectre simulator [47] and estimate the SNM at each point of the 6D space. It is important to note that we only examine "realistic" values of threshold voltage shifts. Therefore the range for $|\Delta Vth_i| \leq 0.2V$ while our step size is $50mV$. A reduction of the step size would of course lead to a more accurate representation of the space, however, it would also require more simulations and increase computational overhead. Hence, there is a trade-off between the analysis accuracy (in terms of space mapping) and simulation time. In fact, when studying netlists with numerous devices, inhibitive simulation times would render the exhaustive sampling of the space impractical.

After we sample the space and estimate the SNM values at each $\mathbf{x}$, we immediately observe the *symmetry* of the space. Looking closer at Figure 4.3 and at basic design principles of the typical 6T SRAM cell [292], we easily realize that the cell is symmetric. This symmetry is confirmed in our analysis of the SNM as well. Specifically, we have verified that the impact of threshold voltage shifts on any of M1, M2 and M5 devices to the SNM value is identical to the same amount of shifts on M3, M4 and M6 respectively, as shown in Figures 4.4a–4.4c. Obviously, this does not apply for the case for other combinations of devices, like for example M1 and M6 as we can verify from Figure 4.4d. Since a thorough representation of the whole SNM space is practically infeasible, we show certain instances where the 4 ΔV_{th} of the devices remain fixed while for the other two transistors, we sweep the threshold voltage shift while estimating the SNM.

Next, we move forward and develop a tool that locates any maxima of the SNM space. To this end, we are based on the `coordinate ascent` algorithm, as presented in previous work [232]. In fact, we have observed in our simulations

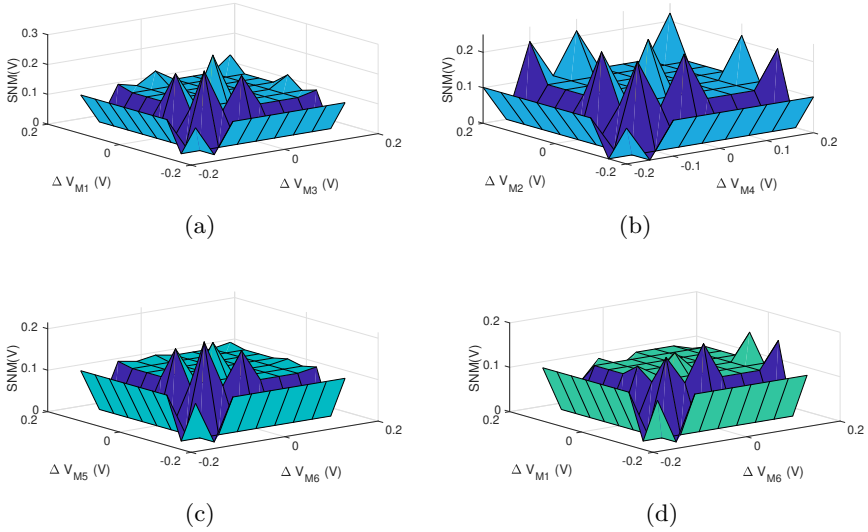


Figure 4.4: Instances of the SNM space: a) SNM space for $\Delta V_{th,M1/M3}$; b) SNM space for $\Delta V_{th,M2/M4}$; c) SNM space for $\Delta V_{th,M5/M6}$; d) SNM space for $\Delta V_{th,M1/M6}$. Symmetry of space is verified through Figures a, b and c.

that the SNM space contains a global maximum and local maxima also exist. Interestingly, we have noticed that this global maximum is located upon the symmetry axis (the pairs of M1-M3, M2-M4 and M5-M6 have the same ΔV_{th}). Our tool starts by initializing the threshold voltage shifts near zero while the results of `coordinate ascent` are presented in Figure 4.5.

It is clear that as the algorithm proceeds to next steps, the SNM value is increasing until reaching the maximum at the final step. Using an exhaustive search, we compared the aforementioned point to all the other sampled points of the SNM space and have verified that this maximum (identified by the algorithm) is in fact the global maximum. It should also be stressed that when utilizing this algorithm and have reached a maximum, we cannot guarantee if it is the global or just a local maximum. To assess whether the first or the second case stands, a study on the *concavity* of the SNM space around this point is essential. Should the maximum be local, we will find points in a distance r to have a higher SNM value.

A strict mathematical proof of the *concavity* of the space is infeasible because the exact formulation of the equation $f(\mathbf{x}) = y$ cannot be known. Nevertheless, we can focus on the area near the maximum and examine whether points in this area have an SNM value lower than the one of our maximum. In addition,

we expect that as distance r from the maximum increases, the SNM value will decrease. This observation would be rendered as substantial proof on the *concavity* of the SNM space and show whether failure is monotonic. Figure 4.6 shows a representation of the SNM space around our maximum. The curves

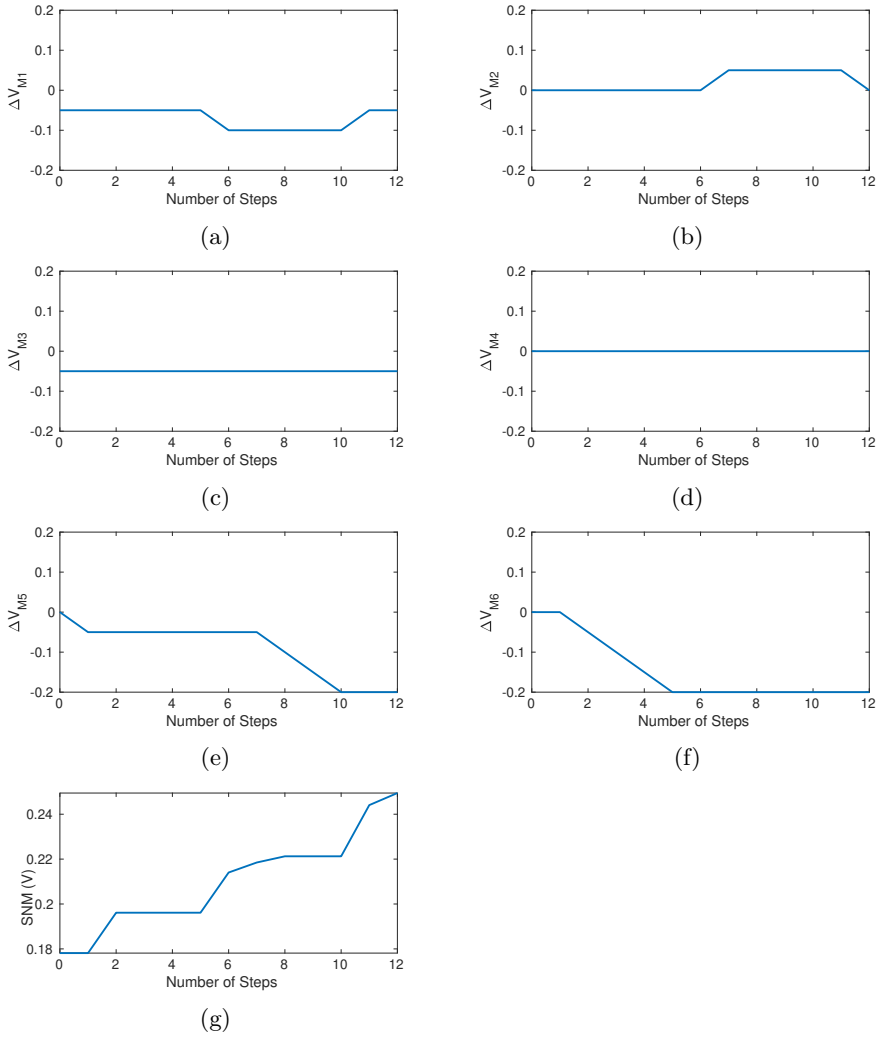


Figure 4.5: Results for the Coordinate Ascent Algorithm: a) CA for $\Delta V_{th,M1}$; b) CA for $\Delta V_{th,M2}$; c) CA for $\Delta V_{th,M3}$; d) CA for $\Delta V_{th,M4}$; e) CA for $\Delta V_{th,M5}$; f) CA for $\Delta V_{th,M6}$; g) CA for SNM.

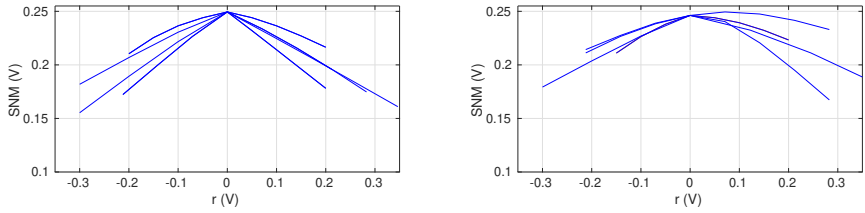


Figure 4.6: Studying the concavity of SNM space. Lines depict the different directions: a) Near $\mathbf{x}_{max}$ space is concave; b) An example of a local maximum in the SNM space, where no concavity is observed.

depict the SNM as we move away from the maximum at multiple directions; each curve depicts the movement away from the maximum at a different direction. The directions were calculated as stated in the Equation 4.6, where $\mathbf{x}_{max}$ is the identified maximum point and $\mathbf{x}$ hints to the direction that we move parallel to each time.

$$\mathbf{y} = t * \mathbf{x} + \mathbf{x}_{max} \tag{4.6}$$

In Figure 4.6 the parameter r represents the distance ($r = sgn(t) \times ||\mathbf{y} - \mathbf{x}_{max}||$) of the threshold voltage shifts from the maximum. All the curves confirm that the point we identified is indeed the global maximum since the SNM value is decreasing in all directions while we are moving away (see Figure 4.6a. Therefore, as variation increases (having the maximum as our starting point) failure is monotonic. On the contrary, Figure 4.6b shows an example of a local maximum in the SNM space where no concavity is observed.

4.4.3 Estimating Failure Probability

In order to estimate the failure probability, we continue and try to locate the ΔV_{th} combination, namely $\mathbf{x}_Y$, with the minimum distance from the maximum r_Y, that fails our criterion. For this reason, we use the `gradient descent` algorithm [232]. According to the algorithm, we initialize the V_{th} shifts to the values of the maximum and at every step, we move towards the most descending direction of the SNM until we reach a local minimum. This procedure is shown in Figure 4.7. It has to be noted that this algorithm follows a different concept from `coordinate ascent`: instead of searching one coordinate at a time, it examines multiple coordinates and then moves towards the most descending direction

hence, this methodology requires more computations per iteration. Previously, we chose `coordinate ascent`, since the path that led to the maximum was not of interest.

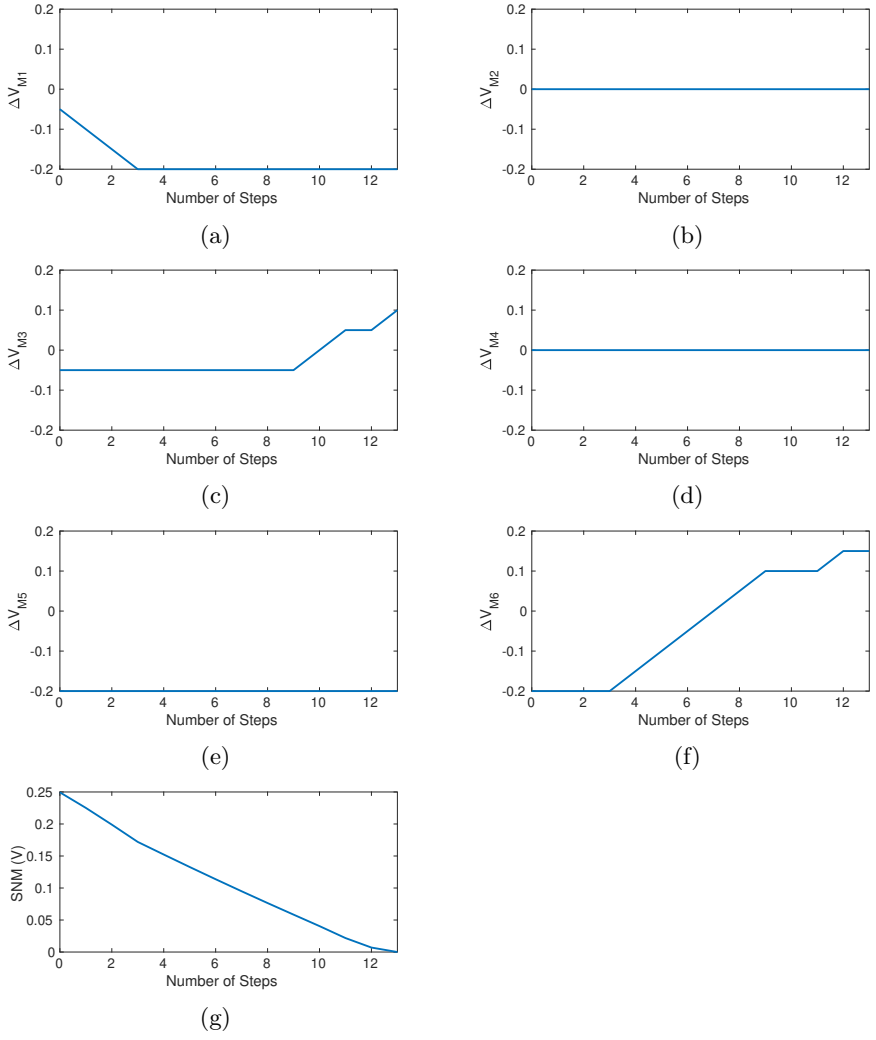


Figure 4.7: Results for the Gradient Descent Algorithm: a) GD for $\Delta V_{th,M1}$; b) GD for $\Delta V_{th,M2}$; c) GD for $\Delta V_{th,M3}$; d) GD for $\Delta V_{th,M4}$; e) GD for $\Delta V_{th,M5}$; f) GD for $\Delta V_{th,M6}$; g) GD for SNM.

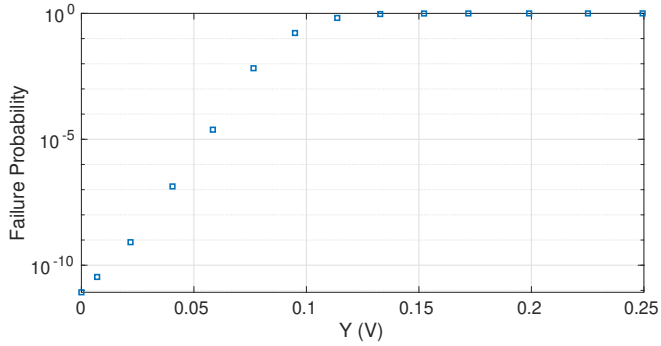


Figure 4.8: Failure Probability for various values of the SNM margins Y

draw hypersphere-like shapes around the maximum with a radius equal to r_Y. Through this way, we are able to estimate the F space and calculate the failure probability. Specifically, we calculate the probability $P(\mathbf{x} \in F)$. To use the non central χ^2 distribution, we follow the Equation 4.7 to calculate the random variable; N is the number of devices and σ_i represents the standard deviations of the threshold shifts of the transistors. To estimate the non centrality parameter λ we use μ to represent the mean ΔV_{th} of each device at the global maximum.

$$z^2 = \sum_{i=1}^{N} \frac{x_i^2}{\sigma_i^2} \qquad \lambda = \sum_{i=1}^{N} \frac{\mu_i^2}{\sigma_i^2} \tag{4.7}$$

Finally, failure probability is estimated using the following methodology: For each $\mathbf{x}_Y$ point with distance r_Y from the maximum, we calculate the failure probability regarding as Y the respective SNM and utilizing the χ^2 distribution. Specifically, P_F for each Y margin value satisfies Equation 4.8, where CDF and PDF are the cumulative and probability density functions of the χ^2 distribution.

$$P_{fail} = 1 - CDF(r < r_Y) = 1 - \int_{-\infty}^{r_Y} PDF_r dr \tag{4.8}$$

Figure 4.8 shows are failure probability estimations for the respective Y margins. We can see that a decrement of Y results to lower P_F values and vice versa. This is expected, since failure is directly depending on the criterion and the specified margin level. Therefore, when high levels of SNM margin are demanded, the failure probability of our cell rapidly increases. We can move forward with our work by approximating these data to a known distribution. This way, we can

have a mathematical formulation to "connect" the margin parameter to the failure probability.

4.4.4 An Efficient MPFP Framework

Until now, we have utilized the MPFP methodology to estimate failure probability nevertheless, we have exhaustively sampled the SNM space and this required considerable simulation times. We now focus on upgrading the MPFP tool by taking the symmetry of SNM space into consideration. Specifically, this time, we will not sample the whole space. On the contrary, we will use Spectre "*in-the-loop*", while the two algorithms are executed. In addition, we will improve the `coordinate ascent` algorithm and exploit the symmetry of the SNM space. We have verified that the global maximum lies on the symmetry axis hence, this upgraded version of the algorithm utilizes the coordinates of the symmetry axis. An example of the optimized implementation of `coordinate ascent` is shown in Figure 4.9. Simulation time is now drastically reduced therefore, we have decreased the step size to $25mV$. On a typical 6T SRAM cell, the symmetry of the SNM space between devices M1-M3, M2-M4 and M5-M6 exists regardless of the technology nodes. As a result, when studying the P_F of any 6T SRAM cell, the current efficient MPFP version can be utilized without recurring computations studying the concavity or symmetry of the SNM space through exhaustive sampling.

In the same spirit, we move forward and utilize the `gradient descent` tool starting from the maximum. Because of the decreased step size, the steps to reach the minimum are almost double. Moreover, since our simulator is running "*in-the-loop*" with the `gradient descent`, total computation time is drastically reduced. In fact, we can decrease step size to achieve even higher accuracy in our estimations. Figure 4.10 presents the new results of the algorithm. Following Equation 4.8, we have estimated the cumulative distribution of Y and eventually the distribution of P_F (shown in Figure 4.11).

4.4.5 Comparing Efficient MPFP to Monte Carlo

Since the Monte Carlo (MC) technique has been widely employed to evaluate the reliability of CMOS circuits, in this Section we will estimate the failure probability following this methodology and compare it against the results of the MPFP technique developed earlier. The simulation framework for our MC experimentation is illustrated in Figure 4.12. First, we add time-zero variability to a median population size (10^4 samples) of "fresh" SRAM netlists, according to Equation 2.1. Afterwards, we inject time-dependent variability based on

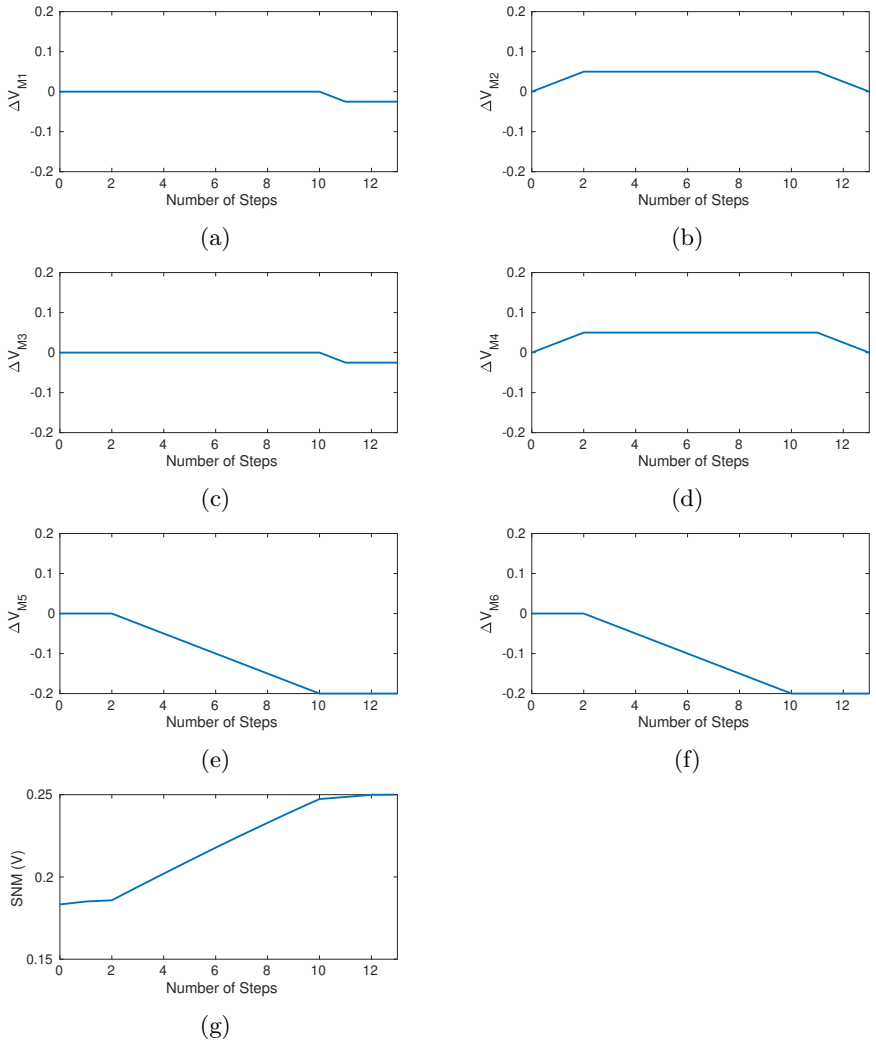


Figure 4.9: Results for the Optimized Coordinate Ascent Algorithm: a) Opt. CA for $\Delta V_{th,M1}$; b) Opt. CA for $\Delta V_{th,M2}$; c) Opt. CA for $\Delta V_{th,M3}$; d) Opt. CA for $\Delta V_{th,M4}$; e) Opt. CA for $\Delta V_{th,M5}$; f) Opt. CA for $\Delta V_{th,M6}$; g) Opt. CA for SNM.

Equations 4.3 and 4.5 to generate the aged netlists, suffering from time-zero and time-dependent variations. Next, we estimate the SNM value of each sample using the Spectre simulator.

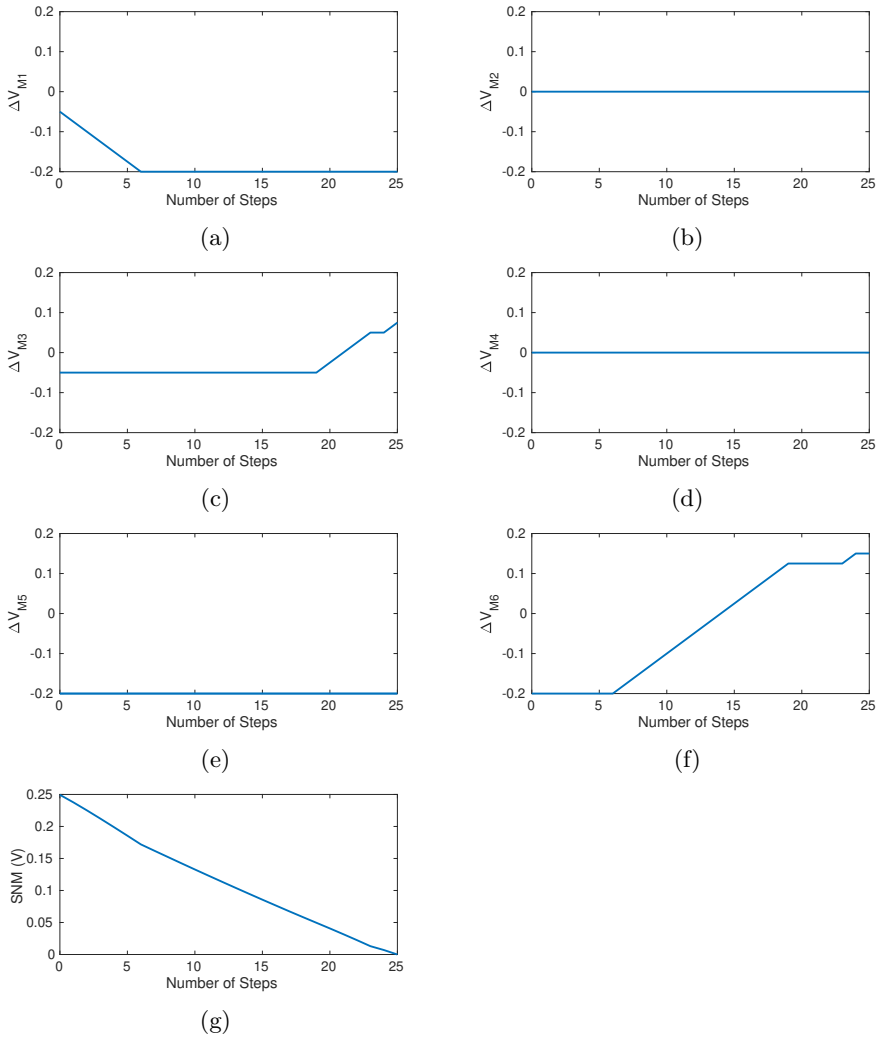
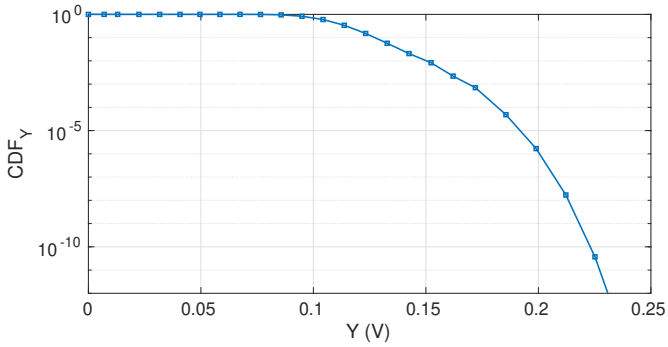
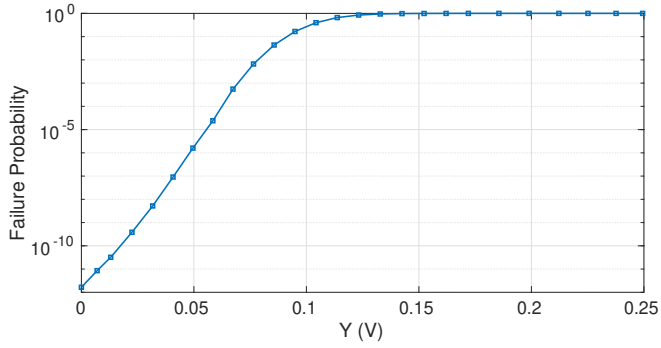


Figure 4.10: Results of the Gradient Descent Algorithm for the Efficient MPFP Approach: a) GD for $\Delta V_{th,M1}$; b) GD for $\Delta V_{th,M2}$; c) GD for $\Delta V_{th,M3}$; d) GD for $\Delta V_{th,M4}$; e) GD for $\Delta V_{th,M5}$; f) GD for $\Delta V_{th,M6}$; g) GD for SNM.

Figure 4.13a shows the histogram of the estimated SNM values along with the design margin Y. Next, we present the respective failure probability estimated by our MPFP tool (Figure 4.13b) for the same Y and we compare the two results. We can notice when targeting a high margin Y value, the estimated



(a)



(b)

Figure 4.11: Results of the Efficient MPFP Approach: a) CDF_Y and b) P_F for our SRAM cell in terms of SNM design margin.

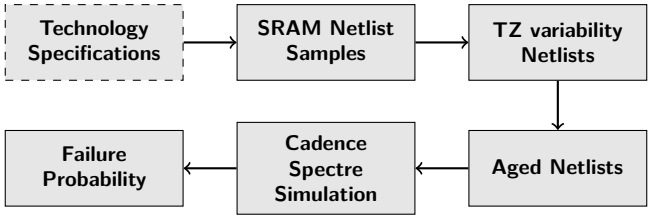
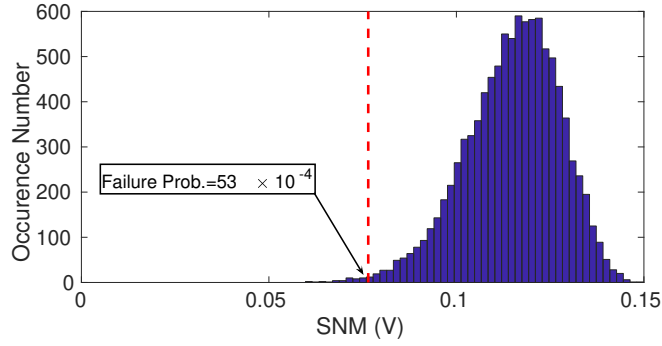


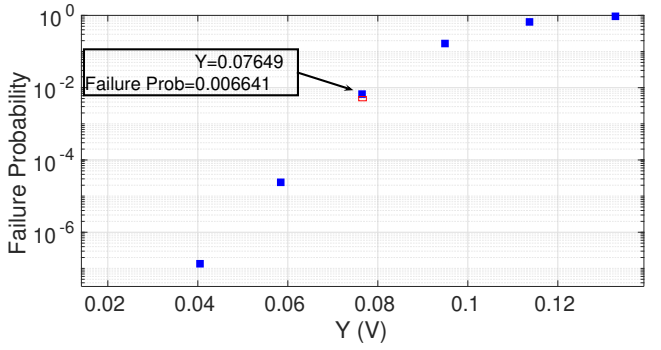
Figure 4.12: The Flow of our Simulation Framework for MC Experiments.

failure probabilities for two different methodologies are (approximately) equal.

To estimate failure probabilities at significantly lower orders of magnitude (i.e. in the scale of 10^{-8}), the MC methodology requires a vast sample population



(a)



(b)

Figure 4.13: Comparison between MC and MPFP for a high value of Y: a) MC Experiments b) MPFP Approach

and long simulation times. This is challenging, especially when low CPU time is a constraint. In our case study, no failed samples are detected when a more realistic margin level is chosen (as shown in Figure 4.14). We should note that in order to overcome this bottleneck of prohibitive computation times, engineers work with a small population of MC samples and later fit the data to a known distribution (usually the Gaussian). However, the SNM histogram shows a rather prolonged tail towards lower values which the aforementioned fitting cannot accurately capture [289]. Hence, we can conclude that while both MC and MPFP methodologies are accurate when estimating failure probabilities of high Y margins, when we examine the probabilities of lower magnitude scales, our MPFP approach – **only after the symmetry and concavity of the space are verified**– presents a more efficient solution. To address this issue regarding MC experimentation, one can select a vast sample population however,

this would render the technique inefficient. Figure 4.15 outlines the difference in the computation time on a Intel CPU i3 8100 processor for the two schemes. It should be stressed that the computational gains between the efficient and the basic MPFP approach are even more drastic since, in the latter case, the complete SNM space is sampled.

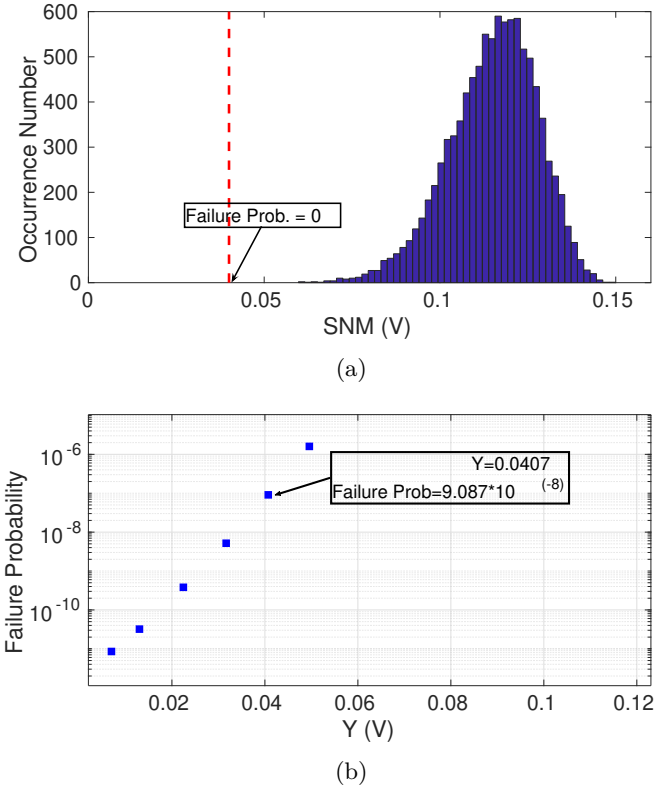


Figure 4.14: Comparison of Monte Carlo with MPFP for a realistic value of Y: a) MC Experiments b) MPFP Approach

4.5 Case Study Using NoC Buffers

After extensively analyzing the MPFP methodology and employing it on a typical 6T SRAM cell, we now study the development of an efficient modeling framework, that a) captures RDF-induced time-zero variability and BTI aging

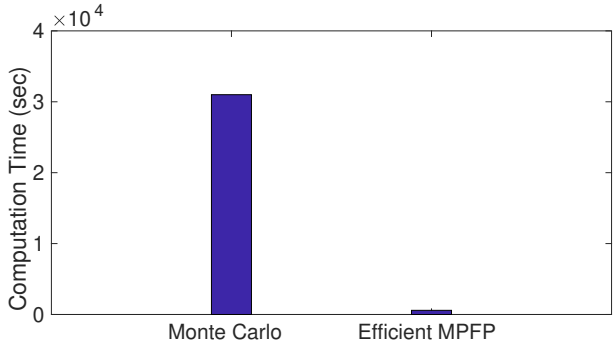


Figure 4.15: Difference in Computation time for the two methodologies.

on a cycle-by-cycle basis, and b) estimates P_F over lifetime operation. We focus on the SRAM memory of a Network on Chip (NoC) router since NoCs are in general crucial components for the operation of a multicore processor; given that the NoC constitutes the communication backbone of the various on-chip modules (as well as the gateway to the off-chip components), a fault in the NoC can render the entire chip useless[6]. Specifically, we will perform a reliability analysis and estimate the failure probability of the NoC buffers (often implemented using SRAM technology), which play an instrumental role in the router's datapath and overall NoC operation.

4.5.1 Proposed Simulation Framework

The flow of our simulation framework is shown in Figure 4.16. The initial input files describe the signal activity of the circuitry, on a cycle-by-cycle basis. Simulation time for a complete DTA analysis can be prohibitive, especially for long time spans. Therefore, we first utilize `CDW approximation` to compress the signal activity and alleviate computation complexity. A brief analysis on CDW compression is presented in Appendix Section A.2 while the tool is introduced and thoroughly described in our related work [233].

Next, we utilize the `variation modeling` tool, that is based on state-of-the-art formulations presented earlier. Regarding process variation, we focus on RDF which is the dominant phenomenon for aggressively downscaled nodes; due to RDF, a transistor's threshold voltage is not fixed when exiting the foundry. On

[6]To use the entire System on Chip (SoC) as case study would go beyond the scope of our aim and means, so instead we have focused on a representative critical component in the SoC, namely the communication network.

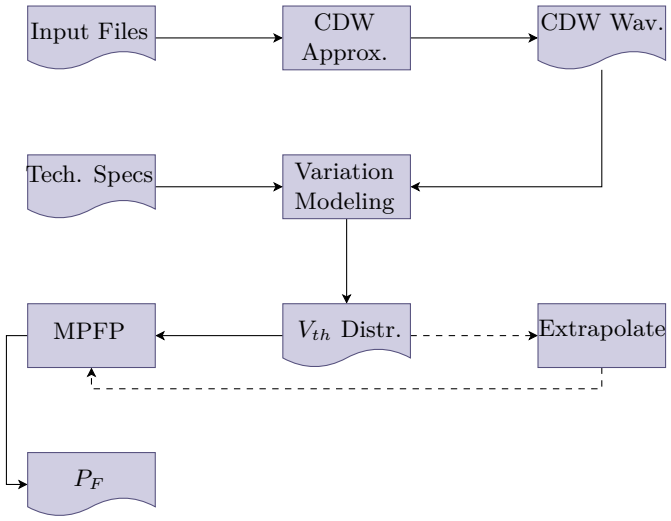


Figure 4.16: Flow of our Simulation Framework.

the contrary,$V_{th,0}$ is distributed near a specified value and can be approximated based on the Equation 2.1. For time-dependent variability, our focus is again on the BTI phenomenon which is the dominant aging mechanism. First, we deploy the reaction-diffusion model: according to it, ΔV_{th} depends on stress time t_s and is fixed at a mean value $\mu(t_s)$, which is formulated in Equation 4.9 (K_v, β_{t_s} are defined in [288]). This time we model BTI in planar CMOS transistors, as opposed to Equation 4.3 that concerns finFET devices.

$$\Delta V_{th} \sim \mathcal{N}(\mu(t_s),\, 0) \qquad |\mu|\,(t_s) = \left(\frac{\sqrt{\frac{K_v^2 a}{f}}}{1 - \beta_{t_s}^{1/2n}} \right) \tag{4.9}$$

The most recent atomistic concept differentiates from the RD theory by including standard deviation. The standard deviation is correlated to the mean value and also, to the time-zero variability, as described in Equation 4.4. The mean ΔV_{th} value remains similar to the one of the RD model. Consequently, the total standard deviation of both time-zero and time-dependent variability is given by Equation 4.5.

To have a more accurate modeling of ΔV_{th} distributions, the relaxation phenomenon of BTI is taken under consideration as well. BTI recovery, according to previous reaction-diffusion models, is formulated as shown in

Equation 4.10 [104]; t_r represents the relaxation time while t_s the previous stress phase. While previous approaches distinguished between a permanent and a recoverable degradation component, recent discussions suggest that recovery of the permanent component can be achieved via a thermally activated process. Up to this day, a mathematical model fully capturing BTI recovery does not exist and the true nature of the distinction between the recoverable and the permanent part remains unknown.

$$\Delta V_{th,t_r} \propto \Delta V_{th,t_s} \left(\frac{t_s}{t_r + t_s} \right)^{\frac{1}{2}} \tag{4.10}$$

Then, failure probability is calculated using the `MPFP` tool presented earlier. It should be noted that since simulation time in this case was not our main focus, we employed the typical version of the tool and not our efficient approach. Nevertheless, the enhanced version of `MPFP` can seamlessly be integrated in our framework as well.

Finally, we use the `extrapolate` tool to project the estimated ΔV_{th} distributions over lifetime spans, reaching up to three years of continuous operation. Hence, we use a power-law-like formulation (similar to older reaction diffusion approaches [207]); n is the H_2 diffusion parameter (used also on Equation 4.9) and b represents the relevant coefficient depending on previous cycle-accurate simulation.

$$\mu(t) = b * t^n \tag{4.11}$$

We should underline that the aforementioned extrapolation enables reliability estimations for the circuitry over long time windows. One could choose to stretch the cycle-accurate simulations over longer time spans, disabling the extrapolation tool and achieving results of notable accuracy. Nevertheless, circuits are typically stressed with numerous workloads under different duty cycles that lead to continually changing stress degrees. This introduces randomness on a circuit's characterization which is hard to capture, constituting lifetime cycle-accurate simulations infeasible. Extrapolation is therefore a necessity if reliability estimations are to be performed in the scale of yearly operations.

4.5.2 NoC Buffers

As proof-of-concept for the framework discussed previously, we hereby present a case study that focuses on the buffers of NoC routers. In general, the advent of the multicore processor era has rendered the NoC as the de-facto on-chip

communication infrastructure. The performance of the NoC is critical to the performance of the workloads running on the multicore processor: an ineffective NoC can have a serious impact on the performance of the workloads, severely affecting the computational power of the entire chip. In addition, an error within the NoC can damage the entire multicore system. If a core fails in a multicore CPU, the system may be able to continue to operate with fewer resources. However, an error within the NoC can lead to various types of deadlocks, or to the network disconnection of various on-chip components, for example the communication of some cores with the memory controller. Therefore, NoCs have become mission-critical actors within multi/many-core environments that need to be studied.

Our goal is to present a case study focusing on the buffers of the NoC routers operating under the *BlackOut* architecture; this architecture [308] focuses on the reduction of static power consumption through power gating certain input port buffers of the NoC.

The BlackOut Architecture

Each NoC router has a pipeline that consists of 4 stages: routing computation (RC), virtual channel allocation (VA), switch allocation (SA) and switch traversal (ST). Depending on the number of requests per each pipeline stage (that concern a specific output port), the upstream router can safely decide on whether the traffic heading towards the downstream router will increase or decrease. For instance, if the sum of requests for a specific output port in stages RC and VA is higher than the number of requests in the next stage, then the traffic is increasing. Therefore, the upstream router needs to signal the downstream router that incoming traffic is rising and more switched-on buffers are required in order to leave performance unaffected.

When receiving a signal, the input port of the downstream router decides to switch on/off buffers. Built into the BlackOut mechanism is the late binding technique: it refers to the timing difference between the instance that a virtual channel is allocated to a packet and the instance that an actual physical buffer is allocated to the packet in the downstream router. By utilizing this time difference, BlackOut can dynamically ”rename” buffers and use them in any order. BlackOut can turn buffers on and off in the context of an expanding and shrinking window.

4.5.3 Simulation Results

We implemented and integrated the network architecture with an updated version of the *gem5* simulator [39]. The simulated topology is a 2D 8×8 mesh network with deterministic XY routing. The network was stressed under a uniform traffic pattern with an injection rate of 0.36 flits/node per cycle leading the whole NoC to reach the onset of saturation[7]. We will perform a reliability analysis for a router that contains 4 input ports (IP). We choose a representative router from the middle of the topology to secure that all input ports receive traffic (because of their physical location, corner/edge routers in a 2D mesh do not use all their input/output ports).

All ports manages incoming traffic via 5 buffers which are used as storage space and create first-in, first-out (FIFO) queues. The router's architecture along with the IPs is depicted in Figure 4.17 where we label the 4 ports as VC0 – VC3 (virtual channels). For this analysis we will utilize our aforementioned simulation framework and will consider the SRAM buffers to be synthesized on HP-PTM 45 nm technology nodes [50] to present a realistic case study. Our analysis covers all NoC buffers nevertheless, we will only focus on the most critical ones, i.e., the buffers where P_F is higher.

Under uniform random traffic, a flit can have any of the network's nodes as destination with equal probability. A fine-grained control of the traffic's injection rate is enabled through synthetic traffic hence, it is used to stress the NoC to its limit, i.e., to the threshold of saturation, revealing the attributes of the network itself (in contrast to application-specific nuances).

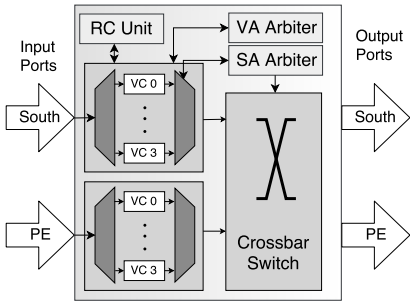


Figure 4.17: The architecture of the router with the relevant IPs.

Initially, input traces are compressed and generate the CDW representation of the overall signal activity. Signal files describe the buffers' workload for 10^5 cycles, with an operation frequency of $f = 1GHz$. For the CDW approximation, the user-defined error $\varepsilon = 0$ is chosen, while other ε values can also be selected. Figure 4.18 presents the simulated ΔV_{th} estimations after the CDW waveforms. Based on the `extrapolate` tool, we fit ΔV_{th} data to formulation 4.11 and extrapolate to approximately 3 years

[7]We define the injection rate as the probability of a node injecting an atomic unit of a packet (flit) in the network in any given cycle.

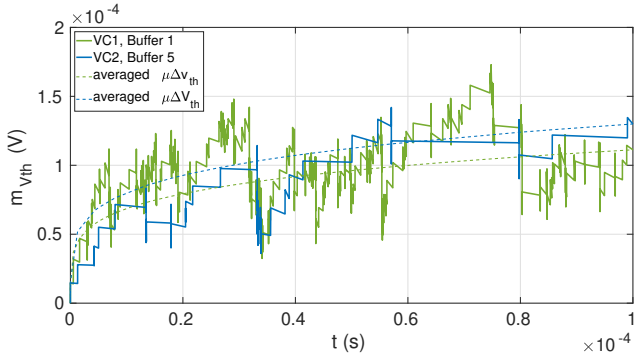


Figure 4.18: ΔV_{th} after CDW for normal operation.

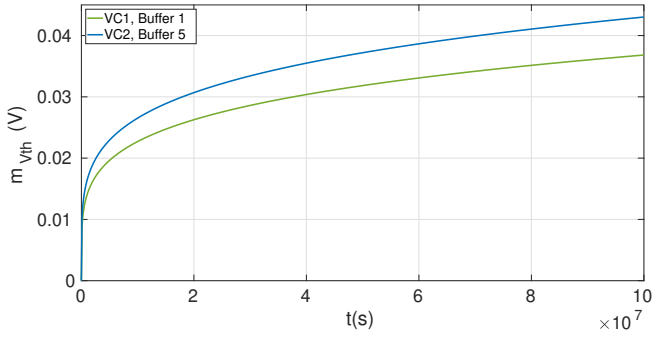


Figure 4.19: ΔV_{th} after 3 years of normal operation

of operation (10^8 seconds). Figure 4.19 presents the results. Again we should underline that only "critical" buffers are shown.

Then, our `MPFP` tool is utilized to estimate the failure probability P_F of each buffer at specific time instances (labeled as "aging epochs" in other works [100]). Again, we will study the robustness of the SRAM cell during the hold operation, represented by the SNM metric. At the same time, we will also take into account recent measurements on the measured levels of thermal noise for our current technology [261]. Therefore, our failure criterion is shown in Equation 4.12, where 25 mV represents the design margin to account for thermal noise. Estimated failure probabilities for the two "critical" buffers are presented in Figure 4.20. It should finally be noted that P_F values represent failure probabilities for one SRAM cell of the buffer.

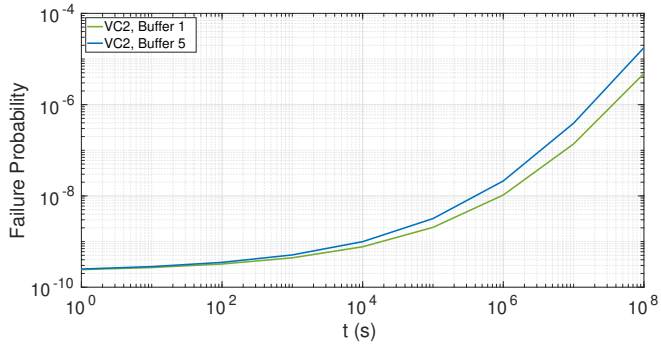


Figure 4.20: P_F after 3 years of normal operation.

$$SNM(\mathbf{x}) < 0.025 \tag{4.12}$$

4.6 Conclusions

In this Chapter we have discussed typical techniques for functional verification and especially timing analysis. While STA and SSTA can provide an efficient characterization of circuit's timing profile, their results strongly depend on relative information from standard libraries. When such information is not available DTA analysis and a type of MC experimentation is usually performed. In Section 4.4 we have extensively studied the MPFP methodology and estimated the failure probability of a 6T finFET-based SRAM cell. We discover the symmetry of the SNM space and have developed an efficient MPFP approach. Finally, in Section 4.5 we employ the MPFP tool on a complete framework modeling RDF and BTI on a "cycle-by-cycle" basis and estimating P_F over lifetime operation, attempting an efficient and accurate approach.

Chapter 5

A Closed-Loop Controller to Ensure Dependability under Performance Variation

5.1 Introduction

In Chapter 4, we presented an efficient and accurate modeling framework that captures BTI and RDF and estimates their impact on typical SRAM cells of downscaled technologies. In general, the usage of accurate models to assess the effect of failure mechanisms on a system's reliability and dependability is crucial. Ultimately however, the final goal of computer engineers and software developers is to ensure correct system operation in a functional and parametric level; we have already mentioned (see Section 3.3) the deployment of numerous RAS techniques that target the functional reliability of digital systems. These techniques can secure correct bit-wise operation nevertheless, they often come with the cost of extra clock cycles (among others), deteriorating performance and posing a serious threat to systems with deadline constraints. To this end, we will now study a reliability mitigation technique that manages performance variation in a reactive manner; as a result, we cannot provide full guarantees but the complexity of the approach, especially the online technique, remains very low. The current implementation will be deployed upon a single-die target platform hence, belongs in *Mapping 8* of our presented classification (see Figure 3.7 of Chapter 3). In Section 5.2, we will discuss the fundamental aspects of our scheme, which are based on control theory. Also, we will outline basic

information regarding our implementation such as the development of a rollback mechanism or the available DVFS steps on our processor. In Section 5.3 we will examine the capabilities of our scheme both in terms of efficiency and energy costs; performance variation is generated not only from RAS interventions but also from dynamic workload applied to the processor. As a result, we cannot provide full guarantees but the complexity of the approach, especially the online technique, remains very low. Section 5.4 presents a qualitative comparison between our scheme and a "*conservative*" CPU-Freq governor approach. Finally, Section 5.5 proposes a flavor of the PID controller targeting thermal management.

5.2 A PID Scheme For Dependability

5.2.1 Premises

Slack

We will now discuss the theory of our closed-loop PID (proportional, integral, derivative) controller, which provides a widely-applicable mitigation technique for performance variation. The controller was first introduced in prior art [229] and its basic principle is the reactive response to performance fluctuations actuating DVFS to mitigate timing delays and respect performance constraints. The previous work however illustrated the concept only on small test vehicles. To this end, we differentiate our approach since now: a) real-time RAS events and dynamic workload instances are triggering performance variation occurrences, b) a quantitative comparison in terms of energy cost and timing dependability between our controller and a Linux-based DVFS governor is presented and finally, c) we implement a flavor of our PID scheme for thermal management purposes. For our case study, we instantiated the controller on the NXP IMX6Q board [245], on top of a GSM spectrum sensing application [63] with deadline constraints. To control performance, first timing delays are monitored and then the PID switches frequency and voltage of the processor. To assist us, we introduce the term *slack*, to describe the timing difference for specific code chunks, between their reference execution time (under nominal frequency) and the actual execution time, under performance variability. Hence, in our case, we define *slack* simply as:

Definition 6. *Slack* ($s[n]$) is the difference in the execution time for a specific task between an error-free execution, operating at nominal frequency f_{nom} and the current execution at frequency $f[n]$.

$$s[n] = t_{ref} - t[n] \tag{5.1}$$

where t_{ref} specifies the reference time and $t[n]$ time execution at current frequency.

From Equation 5.1 we realize that negative slack can be monitored when errors are manifesting in the hardware and RAS events are actuated in order to correct system's functional behaviour. Another example of performance variation and slack fluctuating from zero is the case when extra workload is applied that competes for the same logic and memory resources. In these cases, the PID controller should decide to elevate frequency and voltage in order to manage dependability and drive slack to zero. On the contrary, positive slack can be monitored when the frequency is over-boosted, translating into unnecessary energy losses. Then the PID should slow down the processor, increasing the clock's period. To ensure slack convergence to zero, the controller monitors slack periodically and enables DVFS, choosing from a number of available DVFS points. It is important to distinguish between functional reliability and performance dependability: in our scheme, the former is taken care of by a rollback-based mechanism, similar to the ones presented in prior art. Instead, the controller focuses on ensuring application's timing demands.

Energy Model

Two components are responsible for the power consumption in CMOS circuits. Dynamic power is attributed to the charging and discharging of circuit's total capacitance and at a voltage and frequency step n can be estimated through Equation 5.2[1]:

$$P_{dyn}[n] = \alpha C V_{dd}^2[n] f[n] \tag{5.2}$$

where α is the activity factor, depends on the workload and represents the probability that the circuit node transitions from 0 to 1. C is the total circuit capacitance, coming from the wires and transistors and $f[n], V_{dd}[n]$, stand for the frequency and supply voltage levels at the n-th step.

The power consumption from all the leakage currents is defined as static power. Three main leakage currents are generating it: a) the subthreshold current, which is the current flowing across the channel when a device is in the OFF state, b) the gate-leakage current, that is the result of carriers tunneling through the gate oxide and c) the reverse-bias junction leakage current, flowing between

[1] In reality, dynamic power is also the sum of another component $P_{short-circuit}$ which is the result of short-circuit currents when both pMOS and nMOS paths are conducting. With increased loading as in our case however, this component is negligible compared against the switching power counterpart.

source/drain and the substrate. From the aforementioned leakage currents, the dominant one, that contributes most to static power, is due to subthreshold leakage and its magnitude depends on the temperature, the supply voltage and device size. In fact, it has an exponential dependence with temperature therefore thermal management techniques are of paramount importance for reducing static power. While for devices of older technology nodes, static power was negligible compared to the dynamic counterpart, for technology nodes below 90 nanometers, static power reaches up to 30% of total power consumption [102].

To estimate static power, we will follow the mathematical formulations of previous work [141]. Note that in Equation 5.3, we do not capture the temperature dependence (primarily because of the high non-linearities of the model) however, we consider that this approximation is sufficient for our work.

$$P_{static}[n] = V_{dd}[n] I_0 e^{\frac{-V_{th}}{\eta V_\tau}} \tag{5.3}$$

I_0 represents the nominal subthreshold leakage, η is a technology parameter and V_τ the thermal voltage. Immediately, we realize that static power shows a linear dependence to the voltage supply. Finally, total energy consumption for all discrete n steps is the sum of the dynamic and static components (Equation 5.5) and can be calculated as in Equation 5.4[2].

$$E = \int_0^T P(t)dt = \sum_{m=1}^{m=n} P[n_m]\Delta t[n_m] = P[n_1]\Delta t[n_1] + ... + P[n_n]\Delta t[n_n] \tag{5.4}$$

where

$$P[n] = P_{static}[n] + P_{dyn}[n] \tag{5.5}$$

Spectrum Sensing Application

Our spectrum sensing application is used to detect GSM signals within a wide band acquisition. First, the input signal is decomposed in 200 kHz-bandwidth channels and the power spectral density (PSD) associated to each channel is computed. Then, the obtained PSDs are compared to a threshold and the active channels are indicated. The application is streaming in nature, contains heavy digital signal processing computations and has deadline constraints since specific time slots are allocated for the spectral sensing of each channel.

[2]In our work, we use normalized energy units. Searching though the platform's data sheets we have not found enclosed information relevant to power and energy data. Nevertheless, since the study will be performed on the same workload (α) and hardware configuration (C, I_0), results from relative and not absolute energy comparisons are considered sufficient.

5.2.2 RAS Instantiation – Rollback Mechanism

To ensure functional reliability, errors need to be detected and corrected. The development of an error-detection mechanism has been part of our team's previous work [65]. A strategy based on signature comparison has been built which alters the signature trace progressively and uses data from application variables. This strategy consists of two distinct phases: a) the *recording phase*, where the golden reference is created while the application runs without any error occurrences and b) the *comparison phase* in which the application is run on the processor. Then, binary errors can occur and the comparator, operating in real-time, can detect the mismatch at certain checkpoints during the application. It is important to note that the golden trace is stored locally, on the memory of the target board. It should also be underlined that through this configuration run-time error identification is enabled, within 1 ms from its occurrence. Evidently, our signature-based detection scheme cannot capture errors in all application data structures nor can identify the source of the error in the faulty hardware component. These data corruption cases fall into the category of undetected errors.

Apart from the signature configuration that detects the error, significant part of our work has been the development of an error-recovery mechanism to mitigate functional violations and ensure binary correctness. Hence, we have developed a rollback technique to correct system functionality after the *comparison phase*.

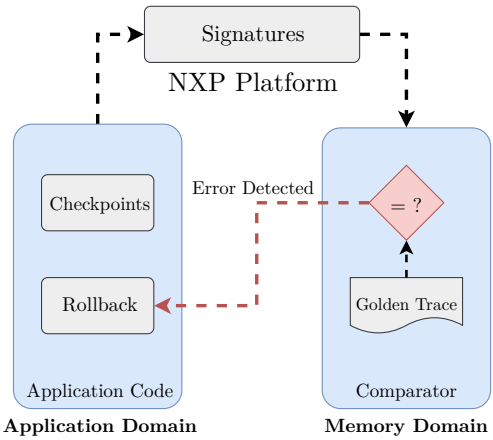


Figure 5.1: The configuration of our RAS mechanism. At certain checkpoints of the application, the system computes the signature and compares it against the golden trace. In case of a mismatch, a rollback event occurs.

Specifically, after an error has been detected, the system is rolled back, the task is repeated and the sensing analysis of the current GSM channel restarts. Naturally, this procedure introduces extra clock cycles and timing delay (negative slack) since the repeated task now needs to be completed on tighter timing demands. It should also be highlighted that with this rollback scheme, the system only recover transient errors; in the presence of permanent or semi-permanent errors, this scheme cannot ensure correct system operation. In this case, fault-tolerant architecture features need to be added in order to start up/power up redundant architecture resources when the initial ones become unusable due to the permanent faults. This extension is outside the scope of our work because sufficient literature is available on this direction [252, 116]. Figure 5.1 summarizes our developed RAS mechanism.

5.2.3 Controller Instantiation

The key features of the PID implementation are a) the monitor that measures timing delay and estimates slack along with b) the knobs used to perform dynamic voltage and frequency scaling. For slack monitoring, we use the `clock_gettime()` function of the *C* library, that allows measuring time intervals between code chunks with a nanosecond resolution. This is one of the most accurate methods to measure wall-clock time. One other way to calculate function time is by reading cycle counters (`CCNT`). These counters are normally disabled, so first they should be enabled from kernel-mode. In our processor, in particular, they are accessible via CP15 instructions [75]. DVFS is supported on the platform, however default support is minimal; three DVFS points are available, corresponding to minimum, nominal and maximum operating points. Nevertheless, this restriction is only a software limitation, and not a hardware one. To this end, we have used a custom DVFS driver by Thales [63], that enables the use of numerous operating points.

Table 5.1: Operating Points of our Processor. Highlighted are the default levels available.

Available DVFS Levels	
f (MHz)	V_{dd} (V)
396	**0.975**
424	0.975
444	1.00
480	1.025
504	1.050
528	1.050
564	1.075
588	1.075
612	1.100
648	1.125
672	1.125
696	1.150
732	1.150
756	1.150
792	**1.175**
816	1.200
840	1.200
864	1.225
900	1.225
924	1.250
956	1.250
996	**1.275**

Our DVFS configuration can be seen in Table 5.1. The timing overhead of DVFS is measured in the scale of hundreds of μs, therefore it is negligible for our case. The main contribution to this is given by the time the protocol requires to set up a new voltage point (through I^2C interface); this time can be reduced by using designs with on-chip regulators [245]. Overall, in this case study, the timing overhead of DVFS is taken into consideration during the slack estimations.

The basic task of our aforementioned application includes the spectrum characterization of a single GSM channel. This task is performed for all channels, until the complete spectrum band has been analyzed. After monitoring the slack, the decision for the DVFS switch is based on the proportional, integral and differential components of the controller. The first component k_p produces an output value that is proportional to the current slack value. The integral term k_i is proportional to the magnitude and the history of the slack, therefore the integral part represents the memory of the controller. Lastly, the derivative part k_d is depending on the slack's slope. Hence, based on this scheme, the frequency multiplier $m[n]$ is selected according to the Equation 5.6:

$$m[n] = \underbrace{k_p s[n]}_{\text{proportional}} + \underbrace{(s[n] - s[n-1])k_d}_{\text{differential}} + \underbrace{(m[n-1] + s[n]k_i)}_{\text{integral}} \tag{5.6}$$

The gains of the controller in this Equation are configured through trial-and-error since they depend on system characteristics. In our case, the gain values are set at $k_p = 0.25, k_i = 0.005, k_d = 0.3$. From prior art, various methods exist to tune a PID controller (such as the Ziegler and Nichols [303] approach) however, these works are outside the scope of this work.

5.3 PID Controller To Manage Timing Deadlines: A Case Study

After we instantiate the controller, we proceed with our case study to examine the capabilities of our proposed scheme. We should highlight that the NXP board is fabricated at $40nm$ TSMC technology nodes [199] and the processor consists of an ARM Cortex-A9 [15] quad-core. The board runs the version 3.0.35 of a Linux kernel image, while the software is cross-compiled through the `arm-fsl-linux-gnueabi` toolchain. In this case study, we focus solely on the spectrum sensing application nevertheless, our proposed scheme can seamlessly be integrated on any software that meets the target application domain assumptions.

5.3.1 Dependability in the Presence of Rollback Interventions

For our experimentation, processor clock frequency is set at 792 MHz and voltage level at 1.175 V. First, during the recording phase we generate the golden trace, in error-free conditions. It is important to notice that our target platform has already proven to be notably tolerant against soft-errors and reliability nuisances. In fact, previous published work [65] has performed accelerated tests and a system-level reliability analysis on the board under aggressive V_{dd} and T conditions. In such conditions, and for typical stress times (in the order of a few days), *Mean Time Between Failure* (see Appendix Section A.1) estimations have shown that $MTBF > 10^5 s$. Therefore, under typical voltage and frequency levels and at ambient chip temperatures, one can expect that $MTBF \gg 10^5$ seconds. This is the reason why *throughout our case study, functional errors are, in reality, software-injected.*

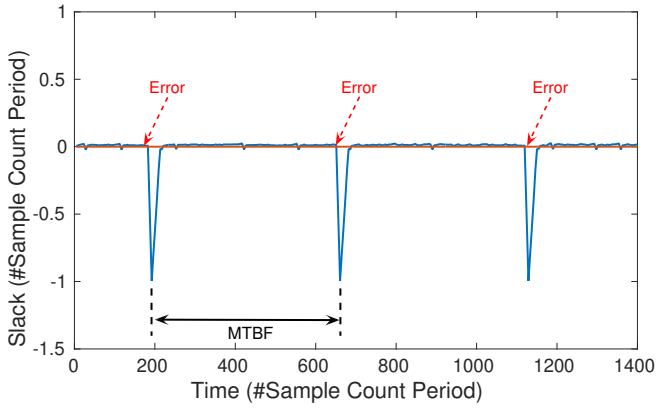


Figure 5.2: Slack over Time for Rollback Interventions.

Figure 5.2 shows how the slack measurement is progressing over time for one run of the application. Time is viewed as sample count period, which is the time for the application to perform one basic data processing computation at nominal frequency[3]. An error is injected, the signature mismatch is detected and finally a rollback event occurs, forcing the system to repeat its previous task. This introduces delay, and drops slack to negative values. When this drop is monitored, the controller takes action and boosts the frequency to absorb the timing overheads and converge slack to zero. In this case, the *MTBF* was

[3]Since the target board is low-power oriented, this basic computational block is measured to last significantly longer than in a high performance platform.

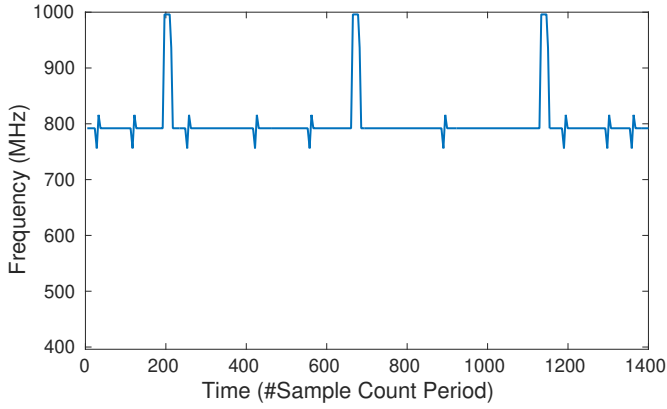


Figure 5.3: Frequency over Time for Rollback Interventions.

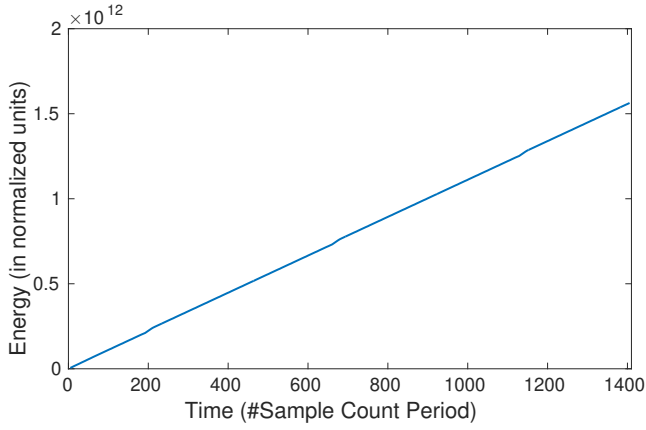


Figure 5.4: Energy Consumption over Time for Rollback Interventions.

selected so that we can manage to compensate for negative slack. An exploration regarding the error rate limits the controller can mitigate is presented later.

In Figure 5.3 we observe the controller's frequency decisions. As expected, the controller drives frequency to the maximum level just after the rollback event and gracefully converges slack to zero - thus meeting our deadlines - while frequency remains around the reference value. We should note that the selection of frequency steps is a function of the error rate, the controller gains and the granularity of the DVFS points. Different error rates and different values for the PID gains can activate different groups of DVFS points. Finally, in Figure 5.4

we present an estimation of the energy consumption (both static and dynamic) for our controller instantiation, estimated in normalized units.

5.3.2 Dependability in the Presence of Dynamic Workload

We move forward by studying the controller's response at the presence of dynamic workload while the board operates at error-free conditions. Such workload variation may result, for example, from input-data dependent computations and is largely unknown at design-time. In our experiment, workload variation occurs when extra workload is running on the same core competing for processor and memory resources. Hence, the controller's challenge now is to compensate for the extra load by optimal DVFS actuations. The imposed timing penalty of performance variation is visible in Figure 5.5.

We can identify the time instances when the extra load is applied to the CPU and a sudden slack drop is monitored. Then, the controller increases the processor's clock speed to converge slack to zero. Positive slack is observed when the extra load is removed and in this case, the processor slows down, since meeting the timing deadlines can be achieved at lower energy budgets. Figure 5.6 presents the frequency response of the controller while the energy estimations are shown in Figure 5.7.

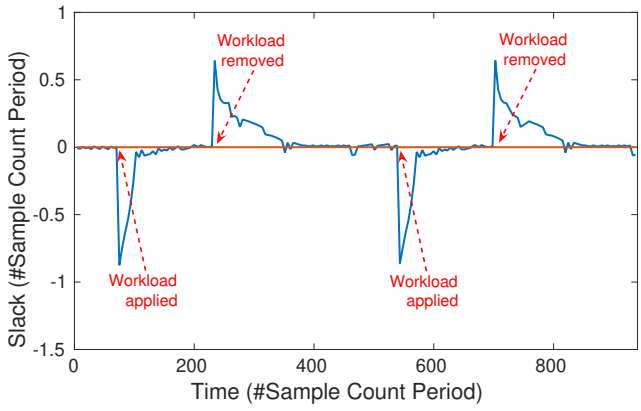


Figure 5.5: Slack over Time for Dynamic Workload.

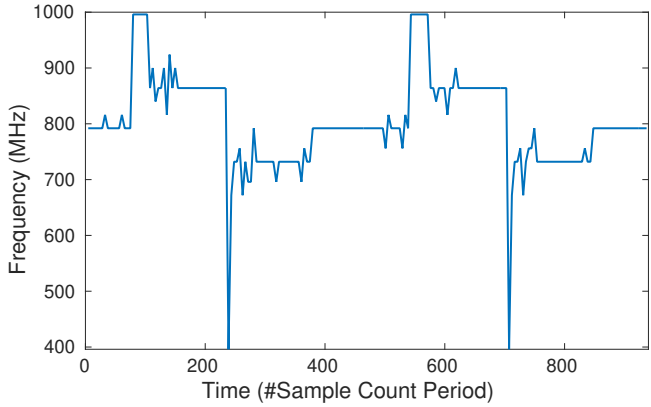


Figure 5.6: Frequency over Time for Dynamic Workload.

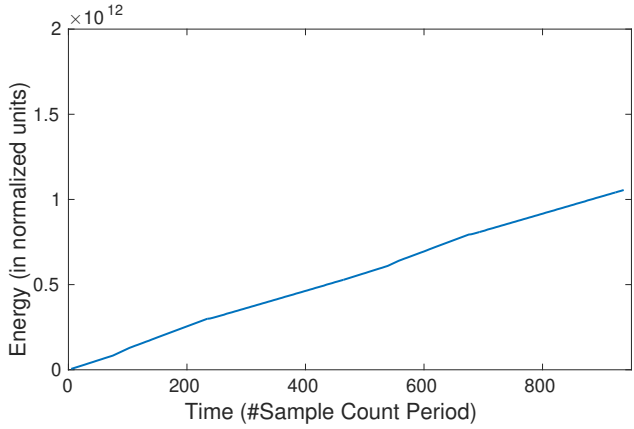


Figure 5.7: Energy Consumption over Time for Dynamic Workload.

5.3.3 Hardware-related Limitations of our Scheme

Our scheme is able to manage timing deadlines, we should however explore its hardware-related limitations. These limitations are solely associated with the hardware's maximum frequency; a higher operating frequency is can absorb timing overheads (imposed by RAS events or extra workload) faster.

Definition 7. *Critical Time* (T_{crit}) : in our case study, we define T_{crit} to be the minimum time needed for the processor to reach zero *slack*, after a timing overhead event (such as a rollback) occurred.

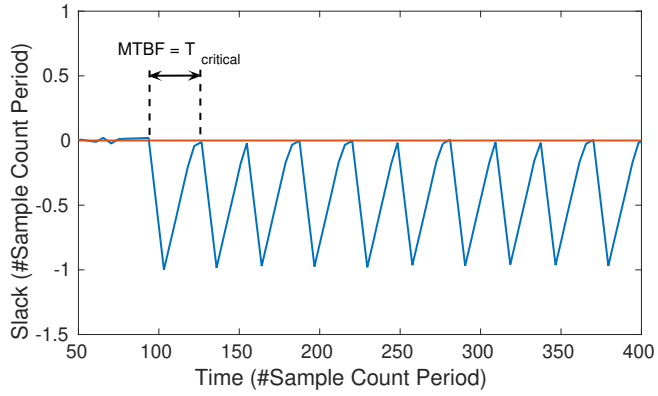


Figure 5.8: Slack versus time for $MTBF = T_{crit}$.

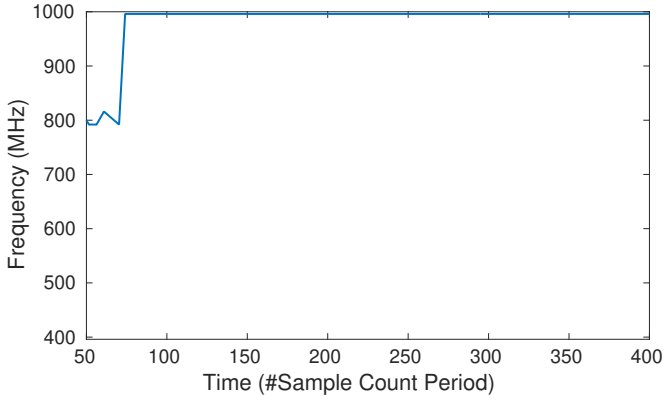


Figure 5.9: Frequency versus time for $MTBF = T_{crit}$.

This approach can be visible if we focus on the *Critical Time* of our instantiation. In our experimentation, assuming timing overhead is imposed by a rollback event, the lowest $MTBF$ rate the controller can handle is shown in Figure 5.8. For this error rate, the controller's action is to drive frequency to the highest step, as seen on Figure 5.9.

For an error rate that is higher than the aforementioned one ($MTBF < T_{crit}$), timing deadlines cannot be met. Figure 5.10 shows such an example. Again, as we can see in Figure 5.11, frequency is "boosted" to the highest level so that timing delays are absorbed. Nevertheless, at this increased error rate, the slack cannot converge to zero; on the contrary, it is progressing to more negative

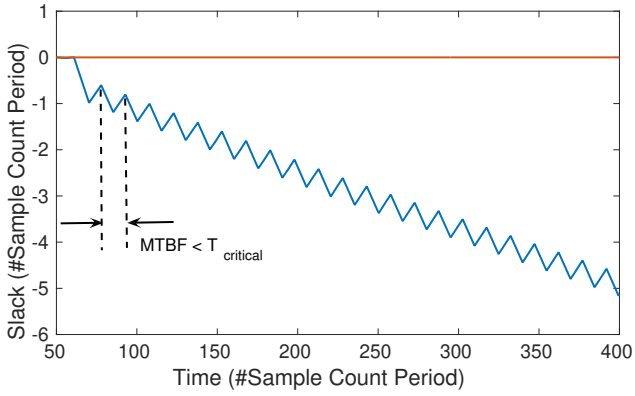


Figure 5.10: Slack versus time for $MTBF < T_{crit}$.

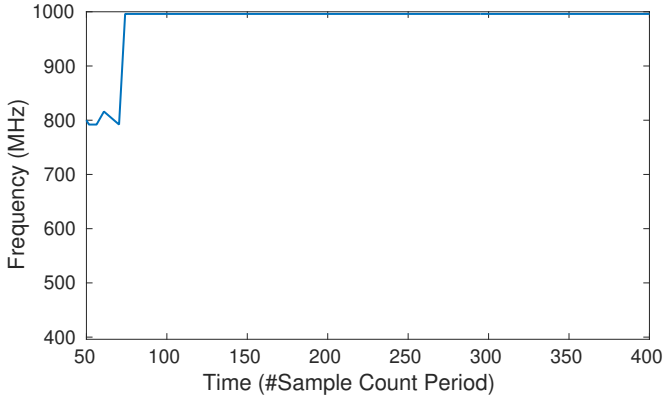


Figure 5.11: Frequency versus time for $MTBF < T_{crit}$.

values. This is not a limitation of the controller but of the hardware. If a higher frequency step would be available, the controller could utilize it and absorb the timing delays, mitigating thus higher error rates[4].

[4]Alternatively, core resources that enable more parallelism would also have enabled performance management for a larger application workload or higher error rates.

5.4 Comparison of our Scheme versus a "Conservative" Governor

Next, we implement a DVFS scheme that is based on the "*conservative*" Linux CPU-Freq governor algorithm , as a reference method to compare with. Another governor, the "*ondemand*", operates in a more aggressive manner, and jumps to max frequency the moment there is any load on the CPU hence, "*conservative*" behavior is more suitable in a battery powered environment. A static, *flat* guardbanding policy is discussed in previous work however, it is evidently less efficient [209]. The selected governor specifically gracefully increases and decreases voltage and speed, depending on the current slack measurement. To characterize this scheme one needs to set certain parameter values, which are described in related work [45]:

1. `SamplingRate`, defines how often the kernel looks at the slack/CPU usage. Typically, it is measured in *usec* (10^{-6} seconds) and its value is around 10000 or more. For this case study, we select a rate equal to 1 #Sample Count Period T_{nom}

2. `FreqStep`, describes the percentage at which the frequency steps are increased or smoothly decreased. By default, the CPU frequency increases by 5% chunks of the maximum frequency. For our experiments and based the DVFS configuration, we chose finer granularity of nearly 2.5%

3. `SamplingDownFactor`, represents the rate at which the kernel actuates a decision on applying DVFS, while running at any speed. In our experiments, it is set to 1 (the default) hence, decisions to re-evaluate frequency are made at same interval with the `SamplingRate`

4. `UpThreshold`, typically defines the upper threshold of CPU usage before the kernel makes a decision on decreasing the frequency. In our experimentation, this threshold is directly correlated to the slack and set equal to 15% of T_{nom}. Therefore, when slack is monitored to be higher or equal to $0.15 \times T_{nom}$, speed is decreasing by one frequency step

5. `DownThreshold`, is similar to the `UpThreshold` however, it regards decisions on increasing the frequency. For our experiments, it is set to -15% of T_{nom}.

Table 5.2 summarizes the governor parameters for our current implementation. Next, a comparison of the two configurations in the presence of rollback interventions is presented. From Figure 5.12, we can identify how both schemes

Table 5.2: The configuration for our "*conservative* governor

Parameter	Description	Value
`SamplingRate`	How often we monitor and estimate the slack	T_{nom}
`FreqStep`	Available freq. step as a percentage of CPU maximum frequency	$2.5\% \times f_{max}$
`SamplingDownFactor`	The rate, as a percentage of `SamplingRate`, at which the kernel actuates DVFS	1
`UpThreshold`	The upper slack limit of the kernel before actuating a decision to to decrease frequency	$15\% \times T_{nom}$
`DownThreshold`	The lower slack limit of the kernel before actuating a decision to increase frequency	$-(15\% \times T_{nom})$

manage to compensate for negative slack that is invoked by our RAS mechanism. However, the PID controller faster converges slack to zero, by approximately 3# Sample Count Periods. A closer look on the Figure shows that the governor scheme, after a rollback event, converges slack to zero after nearly 9# Sample Count Periods whereas the PID controller only after 6. Hence, our controller configuration manages performance variation nearly 33.3% faster. Figures 5.13 and 5.14 show the frequency decisions and DVFS points that are actuated throughout the process.

We can now identify the advantage of the PID controller compared against the "*conservative*" governor: the governor scheme by default increases (and decreases) voltage and frequency gracefully (that is per one frequency step (`FreqStep`)). Therefore, even when slack values away from defined thresholds are monitored, voltage and frequency scaling is actuated per step. This leads to rather suboptimal decisions. On the contrary, our PID controller provides a smarter DVFS policy that allows the processor to switch freely to any voltage and frequency level, in order to compensate for performance variations. This is underlined in Figure 5.14 where we can see that certain DVFS points are utilized by the governor and not from our controller. Figure 5.15 shows a comparison between the energy consumption for the two configurations for this case study; the two lines are in fact overlapping. Hence, we can conclude that the two

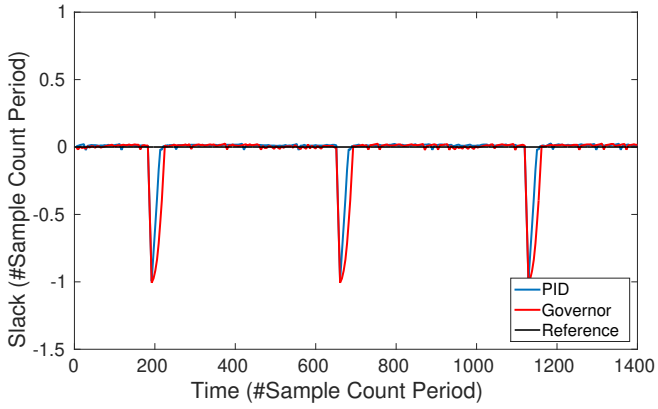


Figure 5.12: Slack versus time for PID and Governor.

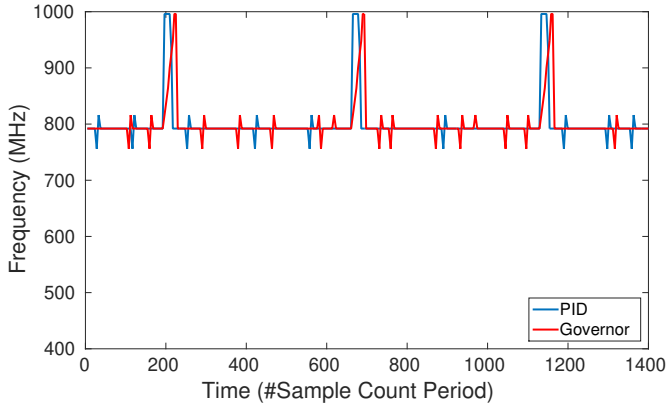


Figure 5.13: Frequency versus time for PID and Governor.

methodologies present similar energy costs however, the PID scheme enables smarter DVFS compared to the governor.

5.5 Temperature Management

The alarming increase in power consumption and circuit density has emphasized the importance of temperature management in VLSI circuits. As previously mentioned, temperature affects system reliability since it exacerbates failure

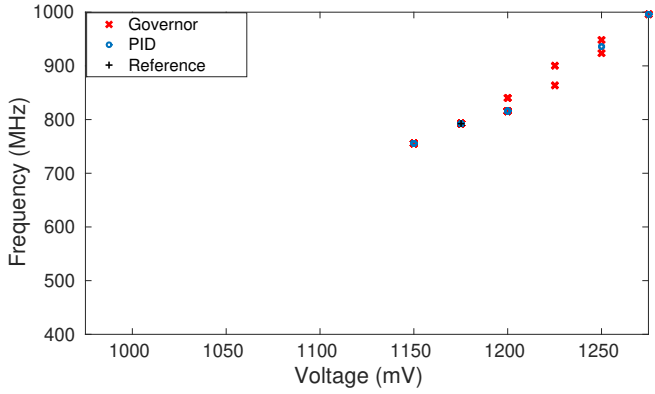


Figure 5.14: DVFS points with the PID and Governor.

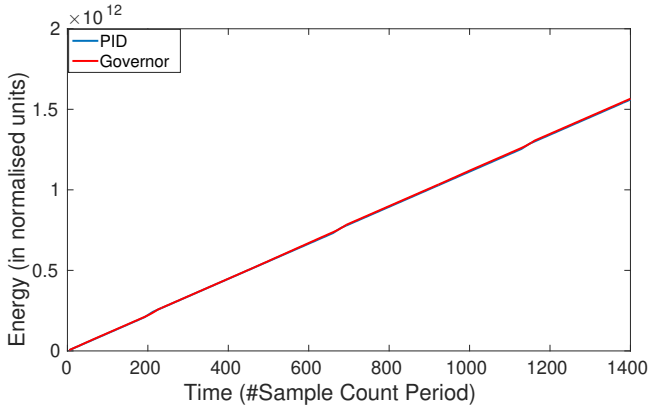


Figure 5.15: Energy Consumption for the PID and Governor.

mechanisms and amplifies current leakage. Transistor performance is degraded with the rise of temperature hence, designers are careful to avoid the generation of thermal hotspots in integrated circuit components. Interestingly, these hotspots can be created due to the unbalanced power consumption, even when average system temperature is below the thermal budget.

One way to deal with thermal hotspot generation and mitigate temperature related-issues is to use a heat sink and allow the heat convection. This is the case with our target board; the chip is in a metal lid configuration which enables the uniform distribution of temperature. Our board also provides an integrated on-chip Digital Thermal Sensor (DTS), calibrated and fused for temperatures up

to $125°C$, which is also the qualification point for most industry electronic chips [246]. By default, the thermal support of the platform is limited: it contains only an alarm function that raises an interrupt if the measured temperature exceeds a specified limit. In this Section, we will extend our PID methodology to also be utilized for thermal management purposes. Moreover, we will provide the experimental results that validate the efficiency of our scheme.

5.5.1 A PID controller for Thermal Management

After studying the controller's response to mitigate performance variation, we now realize a similar scheme to actuate DVFS for thermal management. In this version, key features of our controller are : a) the on-chip digital sensor that enables temperature monitoring of the processor and b) the DVFS configuration, the control the CPU's voltage, frequency and consequently its temperature.

We should underline that the ARM A9 quad-core of our target platform is designed with a single voltage/frequency island [15] therefore, all cores can operate at the same frequency and voltage level. Regarding our methodology, first, a target temperature T_{tar} is selected. *Only* when we measure T_{mon} higher than our target one, the controller actuates thermal management through proper DVFS. Temperature is monitored periodically, with a time step of approximately 10 seconds. A custom driver that allows a 13-DVFS-points configuration is used instead of the limited default DVFS support. We present this configuration in Table 5.3.

Table 5.3: DVFS points for thermal management

Available DVFS Points	
f (MHz)	V_{dd} (V)
396	**0.975**
444	1.00
492	1.025
544	1.050
588	1.075
636	1.100
696	1.125
744	1.150
792	**1.175**
840	1.200
888	1.225
936	1.250
996	**1.275**

The PID scheme presented in Figure 5.16, estimates the error $e[n] = T_{tar} - T_{mon}$ and afterwards voltage and frequency scaling decisions are decided based on the following mathematical formulation:

$$m[n] = \underbrace{k_p e[n]}_{\text{proportional}} + \underbrace{(e[n] - e[n-1])k_d}_{\text{differential}} + \underbrace{(m[n-1] + e[n]k_i)}_{\text{integral}}$$

where k_p,k_d,k_i are the proportional, differential and integral gains of the controller.

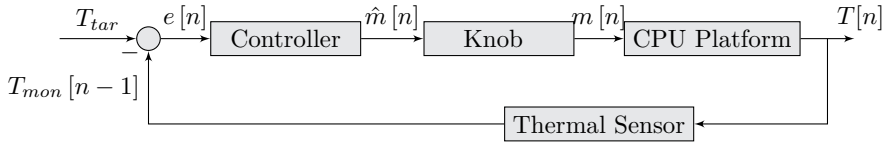


Figure 5.16: PID controller for Temperature Management

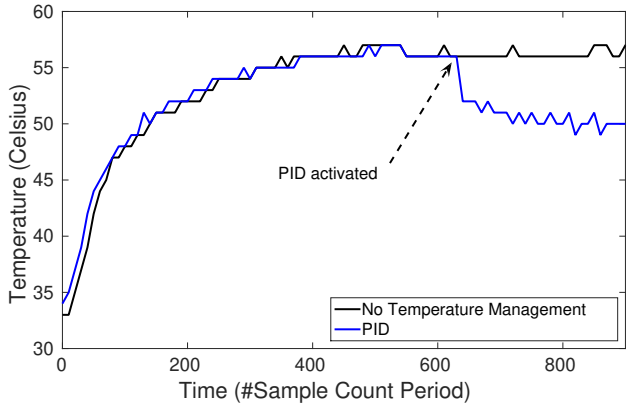


Figure 5.17: Chip Temperature using the PID.

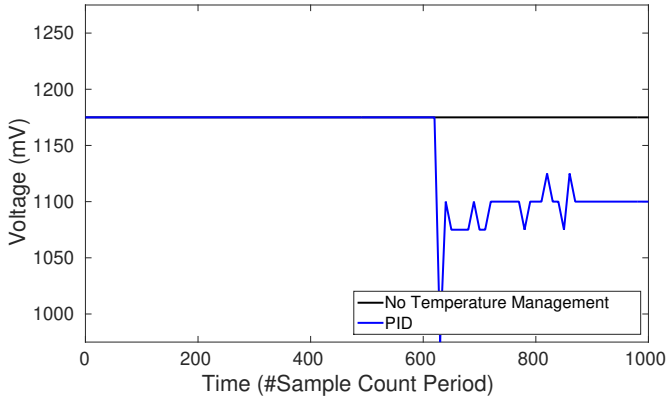


Figure 5.18: Voltage decision for PID.

Figure 5.17 presents an example of our PID controller as a thermal management solution while the target temperature is set to $50°C$. Again, the spectrum

sensing application is used, while the board is in room temperature conditions. This time, the workload stresses the CPU by running in parallel on all 4 cores of the chip. The platform fist functions without activating the thermal management scheme; when chip temperature reaches approximately $57°C$ and then (at 600#SampleCountPeriod), the PID controller is "turned on" to activate DVFS and lower the temperature, bringing it near the target value. From this Figure we can verify how the controller effectively reduces temperature down to T_{tar}. The voltage decisions of the PID scheme are shown in Figure 5.18.

5.6 Conclusions

In this Chapter, we have discussed a PID mechanism that mitigates performance variability by absorbing the temporal overhead from RAS interventions and dynamic workload. The scheme has been instantiated on a real platform, on top of an industry-related application with timing constraints. The RAS mechanism is a rollback technique that works along with a signature configuration for error detection. In Section 5.3, we have verified the effectiveness of the controller when RAS interventions and dynamic workload conditions impose performance variation. The hardware-related limitations of our scheme have also been outlined. Then, in Section 5.4, we have shown the comparison of our scheme versus the "*conservative*" frequency governor, that targets performance variability as well. The key feature of the governor's methodology is the graceful increment and decrement of CPU voltage and frequency. Results have shown that while both mechanisms manage dependability, our PID controller is more effective. Finally, we have moved forward and suggested an alternative version of our controller to perform thermal management. Again, we have instantiated this scheme on our board, which supports a digital, on-chip sensor to monitor the CPU's average temperature. For verification purposes, a realistic case study of such an approach has been presented.

Chapter 6

Mitigating Performance Variations Using System and Adaptive Scenarios

6.1 Introduction

Previously, we defined slack (see Definition 5.2.1) as the difference between the nominal and the actual execution time per task. This allowed us to mitigate performance variation in a task-level basis. In this Chapter, we will deploy this PID scheme on a finer workload granularity, applying DVFS actuations per basic workload chunks named as *Thread Nodes* (TNs); we believe that a TN-level mitigation of performance variation permits a more accurate and cost-efficient mitigation of performance variability. Nevertheless, while many modern application workloads can be split into a handful of tasks, their TN number increases significantly. For this end, we have to come up with a scheme which can handle this much larger amount of TNs for realistic applications. In Section 6.2 we will first define the related terminology and discuss the foundations of the *system scenario* concept that groups TNs with similar cost perspectives. In Section 6.3, we will apply a TN analysis and a system scenario clustering on our spectrum sensing application, while presenting related simulation results. The PID controller working in parallel with an *adaptive system scenario* mechanism, that allows run-time re-clustering decisions of the scenario hierarchy, will be studied later in Section 6.4; simulation results of Section 6.5 will show the benefit of the latter approach. Finally, Section 6.6 will conclude the Chapter.

6.2 System Scenarios: General Concepts and Related Terminology

Before elaborating on the system scenario methodology, we first need to discuss some basic concepts. For a more verbose definition of the following terms and extensive coverage, we refer readers to previous related works [107, 96]. The following work is also partly explained in our Chapter 7 of related book [51].

6.2.1 General Concepts

Following the background of previous works, we can divide system workload into a number of thread nodes; as mentioned earlier, thread node categorization constitutes a finer granularity compared to the task-level one. In short:

Definition 8. *Thread Node* is a set of successive instructions with a deterministic behavior that is executed in a single manner, like for example the body of a loop. In our study, it represents the basic computational block.

Therefore, our application workload can be split into a series of TNs with their sequence depending on the control plane of the platform.

Definition 9. *Run-Time Situation* (RTS) is a tag/label of a TN or of a sequence of TNs, with associated primary costs, like the execution time and the energy budget.

Evidently, workload execution is a sequence of numerous RTSs. In general, a TN can be executed under different power and timing budgets, generating a number of RTSs: a processor with several DVFS points for instance, can execute a TN in any of these steps. Hence, in the timing-energy cost plain, the RTS of a thread node can be visualized as in Figure 6.1. Obviously, TN execution in high DVFS levels results in increased energy costs and shorter execution times.

Definition 10. *Cycle Budget* (N) represents the actual number of cycles of a TN (or a RTS), in the absence of any error-recovery mechanism. In the same spirit, a *Timing Budget* (T_{ref}) can be defined, that is the reference execution time of the TN for nominal/reference frequency f_{ref}.

Moreover, we can express any kind of performance variation, and particularly variation caused by RAS interventions, in the form of *cycle noise*.

Definition 11. *Cycle Noise* (x) represents the extra clock cycles of a TN due to performance variation. The extra cycles lead to higher execution times ($T_{real} > T_{ref}$) of the TN when system is operating under f_{ref}.

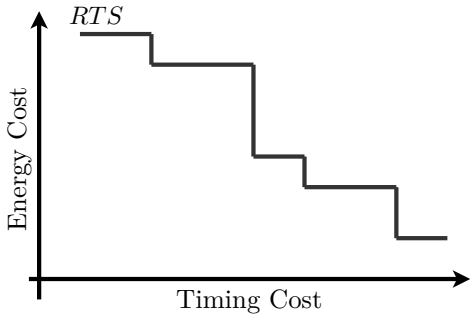


Figure 6.1: Illustration of a RTS in the energy-time cost plain.

We have highlighted how the non-determinism in the occurrence of hardware errors and error recovery interventions threatens timing deadlines and creates dynamic system operation. Cycle noise is injected based on system error rate (Λ) while another important parameter of x is its amplitude (M). This metric largely depends on the type of performance variation: in our case, we will focus on variation caused by RAS mechanisms. For example, an error-correcting code is a type of computer data storage technique that can detect and correct the most common kinds of internal data corruption using redundancy check codes that typically cost a few nanoseconds [11]. In contrast, a checkpoint-rollback mechanism returns the system to some correct, previous state after erroneous operations are performed and may require significantly longer timing penalties [52]. Since actual Λ and M values can be stochastic, we can also utilize λ, μ representing their respective averages over entire workload. Figure 6.2 shows a depiction of these metrics.

Definition 12. *Performance Vulnerability Factor* (PVF) represents the per unit increase in number of cycles for a specific workload due to cycle noise occurrence.

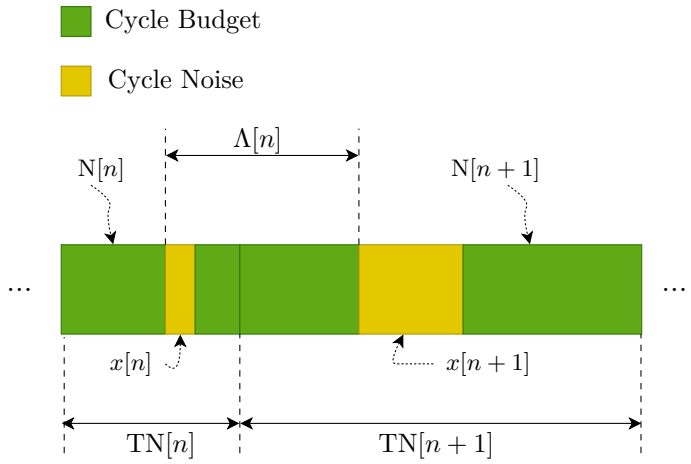


Figure 6.2: Depicture of cycle budgets and cycle noise after TNs execution.

In our case, we examine the workload in the thread node level hence, PVF after the n-th TN is formulated as follows:

$$PVF[n] = 1 - \frac{\sum_{i=1}^{n} N[i]}{\sum_{i=1}^{n} (N[i] + x[i])} \tag{6.1}$$

Definition 13. *Deadline Vulnerability Factor* (DVF) is the temporal equivalent to PVF and represents the per unit increase in execution time (T) for a specific workload because of performance variation: $\text{DVF} = 1 - T_{\text{ref}}/T_{\text{real}}$. After the execution of the n-th TN, DVF is estimated according to Equation 6.2.

$$\text{DVF}\,[n] = 1 - \frac{\sum_{i=0}^{n} \frac{N\,[i]}{f_{\text{ref}}}}{\sum_{i=0}^{n} \frac{N\,[i] + x\,[i]}{f\,[i]}} = 1 - \frac{\sum_{i=0}^{n} \frac{N\,[i]}{f_{\text{ref}}}}{\frac{\sum_{i=0}^{n-1} \frac{N\,[i]}{f_{\text{ref}}}}{1 - \text{DVF}\,[n-1]} + \frac{N\,[n] + x\,[n]}{f\,[n]}} \tag{6.2}$$

$f[i]$ is the operating frequency at step i. By definition, DVF and PVF are identical for constant frequency operation (i.e. $f[i] = f_{\text{ref}}, \forall i$). In practice, DVF tells us how far behind (or ahead) schedule the actual execution is, due to performance variation. An alternative metric to DVF is the slack (see Definition 5.2.1), which we now define in a more precise manner.

Definition 14. *Slack* (s) is the cumulative divergence from the cycle budget, assuming operation under reference (f_{ref}) and current ($f\,[n]$) frequency. We formulate slack according to Equation 6.3, where $n = 0, 1, 2, ...$, $s\,[0] = x\,[0] = 0$ and $f\,[0] = f_{\text{ref}}$.

$$s\,[n] = T_{\text{ref}}\,[n] - T_{\text{real}}\,[n] + s\,[n-1] = \frac{N\,[n]}{f_{ref}} - \frac{N\,[n] + x\,[n]}{f\,[n]} + s\,[n-1] \tag{6.3}$$

The interplay between slack and DVF metrics has been proven in our previous work, verifying the equivalence of $s[n] = 0 \Longleftrightarrow DVF[n] = 0$ [195].

6.2.2 Scenario Clustering Methodology

Modern applications typically consist of a surplus of RTSs hence, storing their information would demand considerable resources. Moreover, a real-time identification of the current TN and its characterization to a RTS can be quite complex and with a significant timing overhead. Therefore, we follow the system scenario approach [95] and manually cluster all application RTSs into scenarios to reduce RTS-based computational complexity. This procedure takes place offline, during the profiling of the workload. Specifically, each scenario represents a bin on the time-energy plain, containing a number of RTSs with similar behavior. Assuming that a workload is composed of k RTSs, the goal of the methodology is to limit this number into l scenarios where $l \ll k$ and now the l scenarios can easily be processed in real-time. An illustration of this clustering procedure in the energy vs. timing Pareto space is presented in Figure 6.3.

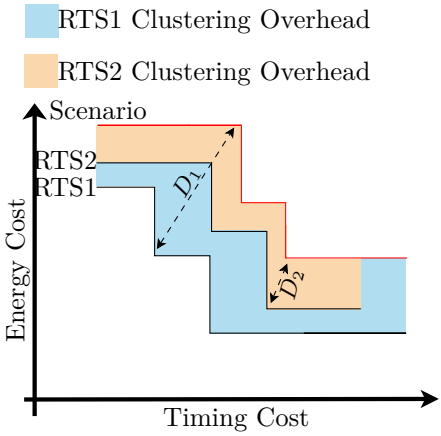


Figure 6.3: Clustering RTSs in a Scenario. D_1 and D_2 show the distance of the RTSs from the scenario and represent the clustering overhead.

The scenario classification however comes with a sacrifice in accuracy. Specifically, the costs of the RTSs in a cluster will not be identical with those of the scenario; on the contrary, the difference in the costs is referred to as *clustering overhead* and can be visualized as the distance of the RTS from the scenario identifier in the time-energy plain. It is important to note that as the number of the scenarios increases, the overhead is minimized and higher accuracy is obtained. However, we now pay a price in complexity, since more scenarios are now generated. It is up to the developer/user to find the best trade-off and proceed with the clustering of RTSs according to his demands.

6.2.3 A PID Controller Utilizing System Scenarios

In our approach, we will again utilize the PID controller with the mission to mitigate performance variations by applying suitable DVFS decisions. In contrast however to Chapter 5, the controller now operates at a TN-level: it first

uses a *scenario detector* to identify the relative scenario cluster in which the current RTS belongs and then estimates the time slack. Therefore a design-time RTS analysis of the application under study as well as a scenario classification precedes the deployment of our scheme. Specifically, after the scenario clustering we develop this detector in order to categorize at run-time the current RTS with its respective scenario.

The key point in the definition of the scenario clustering is to efficiently bound the operation cost (in this case energy and time) of each RTS. Considering that each RTS has a fixed cycle budget, we have explained how several DVFS schemes provide different cost trade-offs. In our PID implementation specifically, each scenario is represented by its RTS **with the minimum delay cost**; this is important considering that scenarios are used as reference to estimate the slack in the PID controller. Through this approach, the clustering overhead creates an overestimated impression of negative slack for the other RTSs, allowing an aggressive DVFS actuation of the controller to manage performance variation. This new version of our PID controller is illustrated in the diagram of Figure 6.4. The scenario detector receives the cycle budget of the executed RTS and decides for the relative scenario in which the RTS belongs to: $\hat{N}[n]$ represents the RTS parameter of the scenario while $\mathbf{p}[n]$ shows the actual cycle budget of the running RTS.

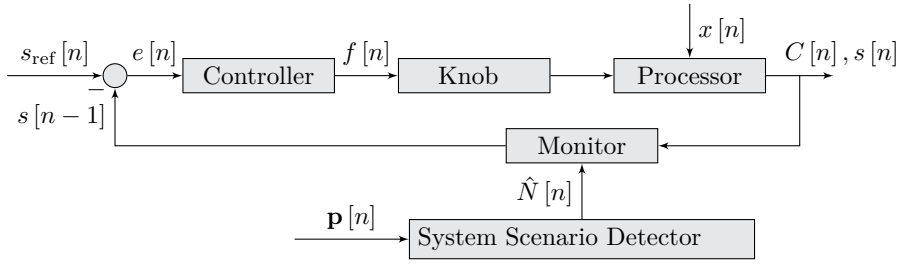


Figure 6.4: Block diagram of the PID scheme with system scenarios [228].

6.3 Simulation Results: The PID Controller With System Scenarios

6.3.1 System Scenario Analysis

To manage dependable performance, all thread nodes have to respect their timing deadlines which are defined through their cycle budgets and the reference

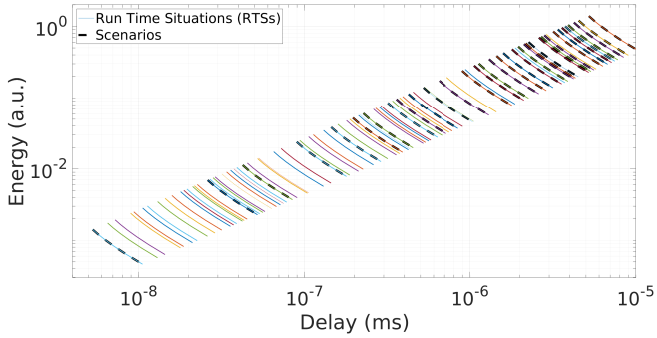


Figure 6.5: RTS clustering to scenarios on the spectrum sensing application. Energy is calculated using the square law for dynamic power, i.e. $P_{\text{dyn}} \propto fV_{dd}^2$ and numbers are normalized to the smallest RTS at $V_{dd} = 1.2$ V and $f = 2.588$ GHz.

frequency. Since the increased TN number of the workload as well as the several DVFS steps of the processor generate a variety of RTSs, it is rather sub-optimal to store RTS information of the workload and detect them at run-time; the number of RTSs encountered in the instruction stream of a processor is in fact significantly difficult to handle. Therefore we have highlighted the use of system scenarios to cluster at design-time the population of RTSs into cumulative representative cases. The clustering methodology for our PID controller was explained in Section 6.2.3.

The RTS analysis, along with a scenario clustering attempt of the spectrum sensing application, can be seen in Figure 6.5 where each RTS represents a certain cycle budget. The DVFS points allowing us to perform the RTS analysis are shown in Table 6.1 and are based on the Intel Core2 Duo E6850 processor; we have chosen this processor for our simulation purposes since, in contrast to our target platform, detailed power and timing data are documented in prior art [203]. The multitude of solid lines in Figure 6.5 corresponds to the application's complete analysis and already represents the complexity of handling TNs using the RTS abstraction. This amount of information, evidently, needs to be abstracted further. Hence, we utilize system scenarios, that can replace the many RTSs with a more manageable set of scenarios to be used as cycle budget estimators. A close look at this Figure shows how we have split the application into a pool of 768 RTSs (performing a fine-grained RTS analysis of the workload) and later grouped them into 32 scenario bins. In reality, the granularity of the RTS analysis depends on the user's/developer's demands: final goal is to achieve the optimal RTS and scenario clustering of the workload under study.

RTSs that require a similar number of cycles to execute are classified into the same scenario. The scenario is uniquely identified with its cycle budget estimator $\hat{N}_k$, which is the minimum (or maximum in some cases) cycle budget of the RTSs that have been grouped therein. In a similar manner, RTSs with similar costs are also falling under the same scenario when perceived at run-time[1]. For our controller's scheme, the fastest RTS is assigned as the cycle budget estimator in the scenario. While moving away from the origin of the axis, new scenarios are defined and certain RTSs are clustered to them. Eventually, the wealth of RTSs is produced by all possible combinations of the application's configuration, like for example accuracy settings and input size. Hence, the system scenario methodology allows us to reduce this huge set of RTSs to a more manageable set of scenarios. It is clear, however, that the complexity reduction detailed above introduces an accuracy overhead, which we have already defined as clustering overhead. There lies a trade-off between the clustering accuracy and the number of the derived scenarios: a fine grain clustering can decrease the gap between the budget estimators however, it increases the number of the scenarios and vice versa.

The assignment of the minimum cycle budget to a scenario forces the TNs to be more restricted, compared to their actual timing deadlines. This trend leads the PID controller to act in a more aggressive manner and boost the clock's frequency. This effect is essential for the purpose of our design: the controller's target is to ultimately absorb the interfering cycle noise of the RAS interventions. Therefore, a small additional frequency boosting operates preemptively towards the absorption of cycle noise. This "negative" clustering overhead (produced by the minimum cycle budget estimators) acts as an offset and enables the mitigation of cycle noise without a reactive response. As a result, this inevitable clustering overhead is not undesirable, provided that it does not force our scheme to operate for long time periods in the "worst case" mode, which is to force it actuate the maximum operating frequency.

6.3.2 Simulation Results

After the system scenario analysis of the spectrum sensing application, we submit this workload to the simple TN-level simulator that has been presented in previous work [233]. The simulator operates at TN-granularity: for each TN, the scenario detector provides the cycle budget estimator $\hat{N}[n]$. Moreover,

[1]Note that we notate the cycle budget estimator for the n-th executed TN as $\hat{N}[n]$. Assuming that this TN is binned to scenario k, then the following equation holds $\hat{N}[n] = \hat{N}_k$. Obviously, when we use system scenarios or any other similar cycle budget estimation techniques, the estimator $\hat{N}[n]$ is used for slack calculation (Equation 5.1), instead of the actual one $N[n]$ which in this case cannot be known at run time.

Table 6.1: DVFS points of the Intel Core2 Duo processor [203].

m	V_{dd} (V)	f (GHz)	m	V_{dd} (V)	f (GHz)
1.2	1.3	3.074	0.9	1.15	2.281
1.1	1.25	2.852	0.7	1.1	1.932
1.0	1.2	2.588	0.6	1.05	1.54

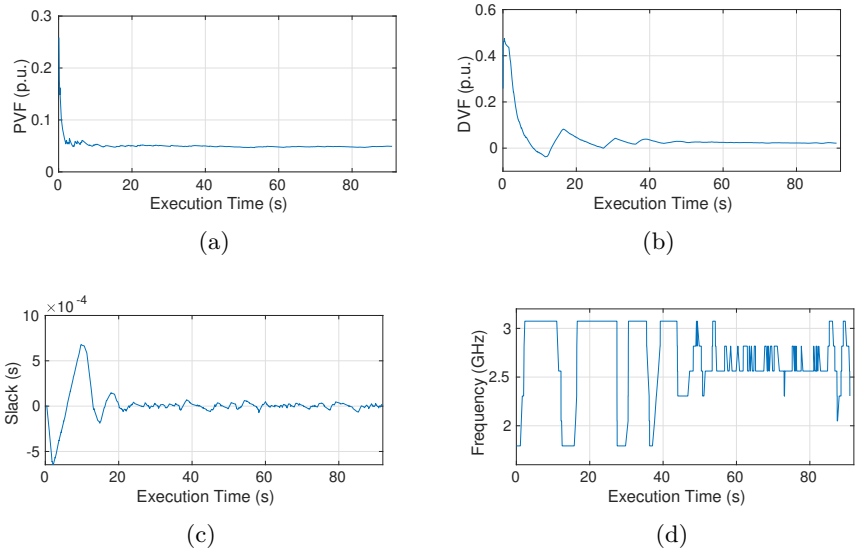


Figure 6.6: An instantiation of our simulator examining the PID's response utilizing system scenarios: a) PVF values ; b) DVF response ; c) Slack response ; d) Frequency decisions of the PID.

we inject cycle noise based on the average values for the noise's intensity and rate, μ and λ and their respective standard deviations $\sigma\{M\}$ and $\sigma\{\Lambda\}$. The timing and power models of the simulator create traces from which DVF and total energy consumption values can be deduced. It should be noted that power models, in this case, account for both dynamic and static power consumption [203]. It is also noteworthy that the DVFS energy and timing overhead is also included in these models. After the completion of an RTS, the PID controller selects the frequency of the next iteration from the values depicted in Table 6.1. Figure 6.6 shows an instantiation of our simulator, examining the controller scheme in terms of DVF,PVF, slack and frequency decisions.

RTSs and their cycle budget estimators are drawn from Figure 6.5. This time

we will inject cycle noise by sweeping μ and λ values and assuming $\sigma\{M\} = 0$ and $\sigma\{\Lambda\} = 0.1 \times \lambda$. This exploration will provide us with interesting results concerning the response of our control scheme across different cycle noise profiles. The simulated workload assumes equally probable RTSs. Each measurement corresponds to a stream of 10^5 TNs and, when enabled, PID controller gains are $k_p = 1.52 \times 10^{-7}$, $k_i = 5 \times 10^{-8}$, and $k_d = 4 \times 10^{-8}$. Figure 6.7 shows the PVF values for this set of noise setup.

In Figures 6.8 and 6.9 we see average values for the DVF and energy consumption after 10 simulation iterations. We can conclude that the controller manages to enable dependable performance since it absorbs the increment in PVF that cycle noise injection is causing; the challenge exists for the extreme cases of cycle noise profile (high μ, low λ). In addition, total energy consumption is also increasing as injected cycle noise moves to these profiles. This however

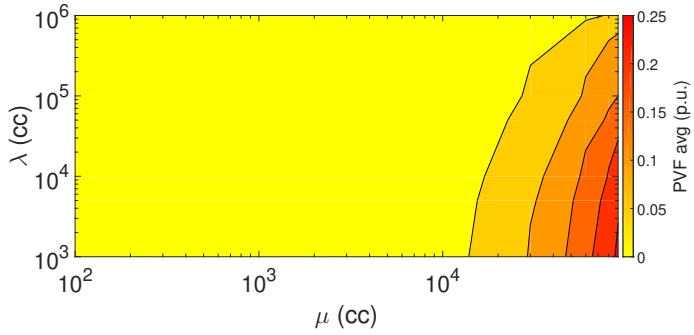


Figure 6.7: PVF values for the set of μ and λ values.

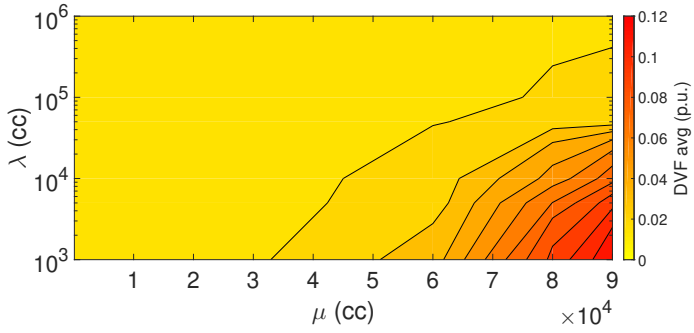


Figure 6.8: Average DVF estimations after 10 simulation iterations for the set of μ and λ values.

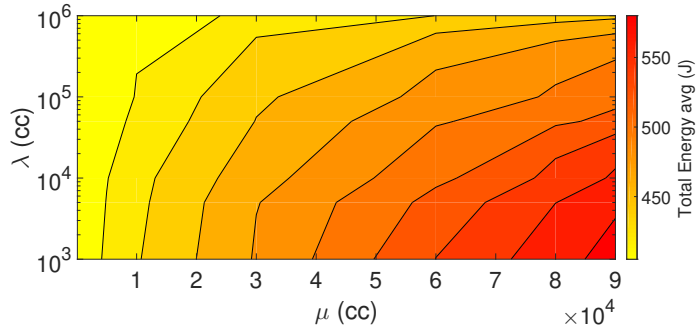


Figure 6.9: Average energy consumption in Joules after 10 simulation iterations for the set of μ and λ values.

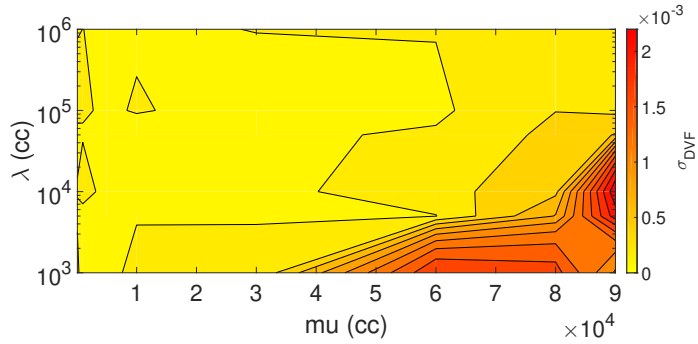


Figure 6.10: The σ_{DVF} after the 10 simulation iterations for the set of μ and λ values.

is expected, given that, in these cases, the controller needs to work overtime. Finally, the standard deviation values of Figure 6.10 provide insight into the statistical variability of our simulation results and indicate that the average DVF reported is substantially accurate.

6.4 Towards an Adaptive Scenario Approach

6.4.1 Scenario Adaptation: Methodology

The system scenario methodology can be easily applied to any application, given that an RTS analysis and the scenario classification are enabled at design-time.

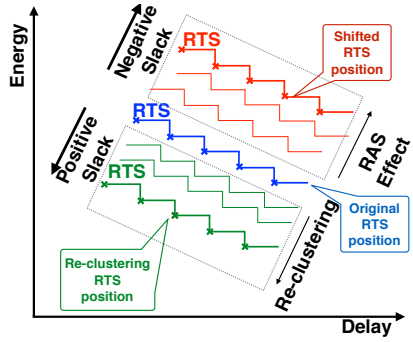


Figure 6.11: Slack impact in the Re-clustering Process.

On condition that the RTS positions into scenarios can be updated at run-time, the scenario detector can modify the RTS hierarchy by re-clustering the existing scenario distribution. This process provides a run-time adjustment of the RTS' costs estimations. If a RTS is correlated with a specific cycle noise, this RTS can change its scenario reference and move to a new scenario by removing cycles equal to the cycle noise. This enables the PID controller to "adapt" to run-time conditions and absorb performance overheads before their presence. Figure 6.11 presents the basic concept behind this methodology: the RAS interventions shift the RTS position far from the start of the axes. The re-clustering process creates a positive slack that works as an offset on the expected negative slack and equalizes the time lost. The goal here is to keep the RTS original position stable. For this reason, RTS slack history needs be recorded and updated on fixed time windows and be exploited at run-time. The re-clustering priority focuses first on the most frequent RTSs with high slack values. This leads the PID controller to act partially proactive and consider RAS events that are gradually becoming a norm on the target processor[2].

For each RTS re-positioning, an individual re-clustering decision is taken to equalize the performance degradation. The scenario detector contains a record of the design-time scenario budgets which is updated after each re-clustering. Regarding the re-clustering decisions, the key point is to distinguish between the transient and (quasi-)permanent cycle noise interference events. For example, a very brief cycle noise burst that rarely reoccurs should not be triggering a re-clustering process. The hardware degradation effects, manifesting during the operation life, cannot be predicted in advance. Thus, an updated trace history of the slack per RTS is needed to consider the error fluctuation in respect with

[2]The re-clustering process applies well when error behavior is statistical, allowing the controller to eventually adapt. On the condition that this behavior is highly random, the partial proactive nature of the scheme will be disabled.

Algorithm 1 Re-clustering

```
1: sc_rts=detect_scenario(rts) ; {Scenario identification}
2: sl_rts = pid_controller(sc_rts) ; {Slack Identification}
3: if sl_rts > 0 then
4:    if rts_list(rts) then
5:       update_rts_slack(rts,sl_rts) {update rts slack history}
6:       if check_slack_status(rts) then
7:          Np=new_rts_pos(rts,sl_rts) {new rts position}
8:          Sc_fit=Find_scenario(Np) {best fitting to scenario}
9:          Merge (rts,Sc_fit) {merge rts to the new scenario}
10:      end if
11:   else
12:      store_rts_list(rts,sl) {new rts slack registration}
13:   end if
14: end if
```

the time. The running slack $RTS\_Slack[i]$ for a moving average window of K_w RTS executions is given by Equation 6.4. After identifying a new trend in cycle noise events, a reformulation of the system scenarios through RTS re-clustering follows.

$$Slack_{RTS[i]} = \frac{\sum_{i=0}^{N} (RTS\_Slack[N - k_w + i])}{k_w} \tag{6.4}$$

Algorithm 1 provides a brief description of the re-clustering process. For each RTS delay, we record the original scenario cycle budget (sc_{rts}) and the time noise (sl_{rts}) cycles. The traced RTS's slack is registered to a list ($rts\_list(rts)$); if the RTS already exists, its record is updated ($update\_rts\_slack(rts, sl_{rts})$). A re-clustering is triggered if the status of the involved RTS shows a stable behavior for a specific time window ($check\_slack\_status(rts)$). Also, information about the distance from the neighboring scenarios ($Find\_scenario(Np)$) is exploited to recognize the optimal re-clustering decision, updating only the scenario data that have changed. Each re-clustering decision ($Merge(rts, Sc\_fit)$) is taken based on the new position of the RTS ($new\_rts\_pos(rts, sl\_rts)$) in the metric space, after reviewing the slack updated data and ensuring that the RTS deadlines in the new scenario will be respected with the minimum cost (Pareto optimal).

The re-clustering is a parallel process that does not interrupt the operation of the PID controller. In any case, the re-clustering time can easily be limited and

kept as a fraction of the RTS operation time. This is due to the simplicity of the algorithm to find the new shifted positions without reordering the whole RTSs by scratch. At the end of each re-clustering, the PID controller has updated information about the suitable DVFS schemes, minimizing the generated slack while paying a price in energy. A trade-off between the produced slack and the energy exists that will be explored later along with simulation results.

6.4.2 Re-clustering Decisions

The re-clustering clearly alters the initial, design-time RTS classification by removing the default RTS reference budgets and artificially creating positive slack that shifts the position of the RTSs into the scenarios (see Figure 6.11). This creates the impression of an extra equivalent delay to the PID controller, proportional to the removed cycles and triggers more aggressive DVFS schemes that absolve the negative slack. Examining the re-clustering decisions that have to be considered at run-time, we outline the following cases:

- In the first case, the scenario budget is restricted enough and no slack exists, hence there is no need for scenario change (see Figure 6.12a). The added noise (red shifted Pareto curve) triggers a revision of the RTS budget (green Pareto curve) that does not exceed the distance between the RTS and the scenario (SC_2), exploiting the clustering overhead as an offset. Thus, clustering overhead can be utilized as a proxy of cycle noise tolerance. The next four cases cover situations where the negative slack cannot be avoided.
- In the second case (see Figure 6.12b), in order to equalize the noise delay (red Pareto curve), the revised RTS (green Pareto curve) is located just under the scenario budget SC_1. Thus, Scenario #1 has a new more restricted case due to the RTS shifting that modifies the existing scenario budget.
- In the third case (see Figure 6.12c) the shifted RTS (green Pareto curve) is closer to a neighboring scenario (SC_2) compared with the original (SC_1). Thus, the RTS is assigned to the new scenario (SC_2) fitting better from distance perceptive.
- In the fourth case (see Figure 6.12d), a new distinct scenario budget is created. Two reasons lead to such a decision: 1) none of the existing scenario cycle budgets cover the new RTS constraint or 2) the creation of an individual scenario is more cost effective by merging with an existing scenario.

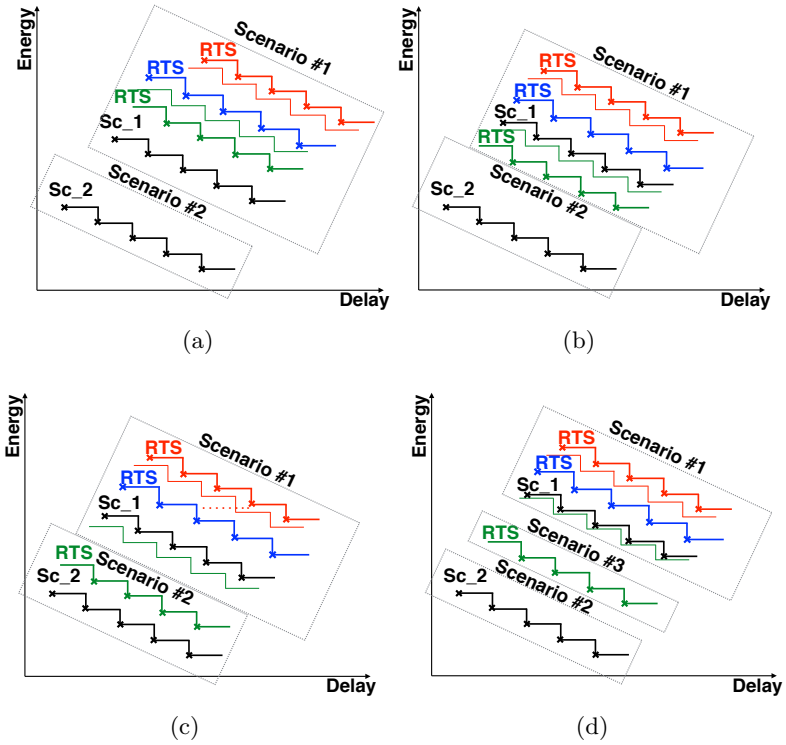


Figure 6.12: RTS shifted cases due to RAS interventions: a) RTS_1 remains in the scenario ; b) RTS_1 creates an individual scenario ; c) RTS_1 changes the scenario bounds ; d) RTS_1 is merged into another scenario.

- The fifth case is a combination of the second and the third aforementioned case. A neighboring scenario merges the revised RTS (as in Figure 6.12c). However, the position of the RTS in the new scenario is located just under the scenario budget (as the blue RTS in Figure 6.12b) changing the new scenario cycle budget as the most restricted case.

6.5 Evaluating Adative Scenarios: Simulation Results

Now, we will explore the PID scheme enhanced with the adaptive scenario methodology while examining the efficiency of the new concept; for simulation

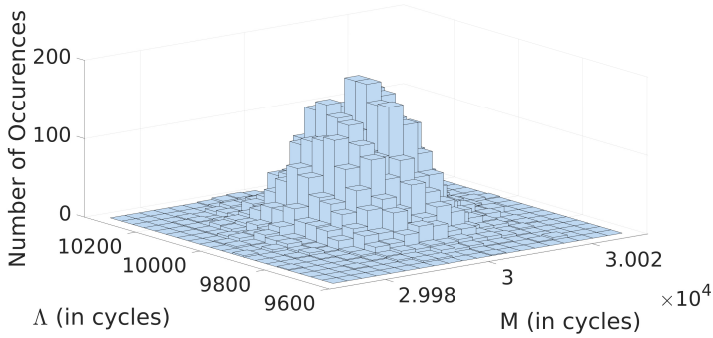


Figure 6.13: Histogram of the cycle noise injected in one RTS.

purposes, we utilize once more the spectrum sensing application (see Figure 6.5). Again, we inject cycle noise based on the values of Λ and M that represent the rate and amplitude of RAS events respectively in terms of clock cycles. It is assumed that cycle noise follows a bi-variate Gaussian distribution with mean Λ and M values (λ and μ) and σ ($\sigma\{\Lambda\}$, $\sigma\{M\}$). Again, the timing and power models of the simulator create traces from which DVF and total energy consumption values can be deduced. Also, in our simulations, all overheads of DVFS actuations are included in both the timing and power models. The simulation results presented herein correspond to a set of random streams of the target application RTS while the simulated workload assumes equally probable RTSs. To inject cycle noise, we once more sweep μ and λ values while $\sigma\{M\}$,$\sigma\{\Lambda\}$ are kept fixed.

The noise histogram for one stream of RTSs is shown in Fig 6.13. In fact, each measurement corresponds to a stream of RTSs and for each combination of μ and λ values, we show the average over 10 random measurements. This way, we can provide insight into the statistical variability of our simulation results. To underline the benefits of the scenario adaptivity, we compare this approach versus a) the DVFS scheme with system scenarios, presented in Section 6.2.3 and b) a (fixed) frequency guardband. A static, flat frequency usage is in fact widely employed to ensure timing deadlines [209]. To guarantee, for example, that timing constraints are not threatened by imposed error-recovery interventions or other sources of dynamism, the user can select to increase system speed in a manner that any overhead is absorbed and application finishes before its deadline. Such frequency guardband policies for example can be the ”performance” frequency governor of the Linux OS, where frequency is set at its highest level. From the simulation results, we can first of all conclude that, again, apart from extreme noise profiles (high μ and low λ

values), the controller with the adaptive scenarios methodology manages to enable dependable performance, especially considering the increment in PVF (as seen in Figure 6.14) which is the result of cycle noise injection.

6.5.1 Dependability Results

From Figures 6.15, 6.16 and 6.17, we clearly distinguish that the adaptive scenarios are evidently more effective in terms of eliminating negative slack compared to the system scenarios and competitive to the guardband. In fact, the timing performance gain of the adaptive scenario approach seems to be approximately 5% higher compared to the system scenarios while it is less effective than the gaurdband only for limited, extreme noise cases. More precisely, in Figure 6.18 we highlight how the adaptive scenarios clearly improve the performance dependability at a representative simulation instantiation. While system scenario DVF seems to balance to a value near zero, adaptive scenarios converge DVF closer to zero. For a certain transition period (0-2000 ms), the number of the adaptive scenarios is fluctuating to reach to a stable state. Throughout this time window, the re-clustering process creates new scenarios and gradually increases their number up to 38 (1000 ms). After this point, some of the existing scenarios are merged and reach a final number of 33 scenarios (2000 ms). Thus, the adaptive process creates and merges scenarios by applying the aforementioned re-clustering decisions and finds the optimal scenario clustering. After this process is completed (at 2000 ms), the scenarios remain stable and the DVF converges to zero without applying an over-boosting frequency policy (negative DVF) with unnecessary energy consumption results.

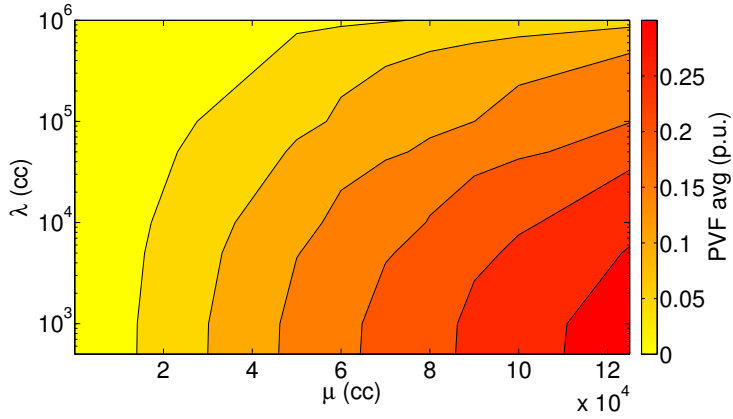


Figure 6.14: PVF average values of 10 identical iterations.

Thus, adaptive scenarios seem to ensure performance dependability without paying a remarkable energy penalty.

6.5.2 Energy Remarks

From Figures 6.19 and 6.20 we can notice that the energy gain of the two scenario methodologies is significant, compared to the guardband technology, and reaches up to 15%. This number is decreasing as the injected cycle noise profile becomes more intense (high μ and low λ). This however is to be expected, since, in this case, the scenarios attempt to mitigate the worst-case conditions that correspond to a state similar to the guardband. The energy consumption deviation between the system and adaptive scenario is negligible and does not exceed the 3%, as seen in Figure 6.21. The reason for this is that the extra energy, consumed from frequency boosting on the adaptive scenarios, is partially equalized by the less number of DVFS switches due to the scenario adjustment that has previously taken place. Hence, we can realize that a fundamental trade-off exists between the achieved performance as expressed by the DVF and the energy impact that will be examined next.

6.5.3 Design Trade-offs

To understand the different trade-offs between performance and energy regarding system and adaptive scenarios approaches we will first increase the reference cycle budget. This is completed by homogeneously applying homogeneously a percentage relaxation factor F_{rel} to the MRC of each scenario (Fig. 6.3). This, of course, leads to an equal clustering overhead increase in each scenario.

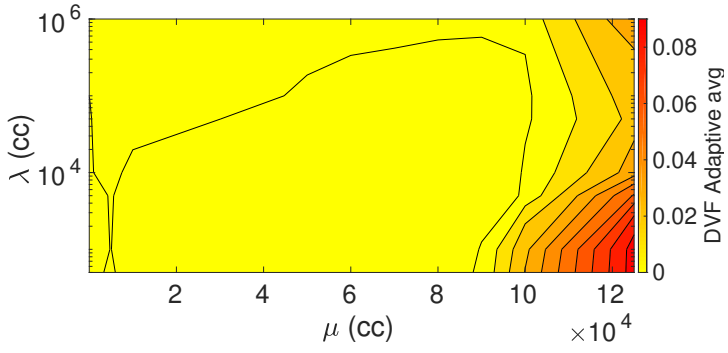


Figure 6.15: Average DVF for Adaptive Scenarios after 10 iterations.

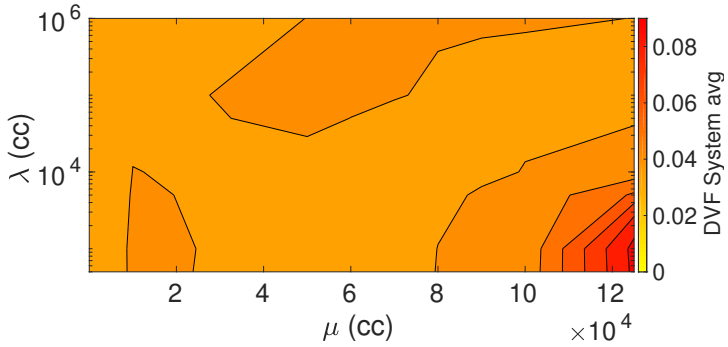


Figure 6.16: Average DVF for System Scenarios after 10 iterations.

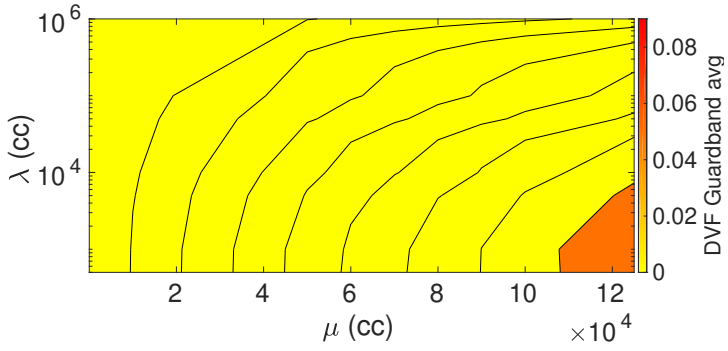


Figure 6.17: Average DVF for Frequency Guardband after 10 iterations.

The new clustering overhead is formulated in Equation 6.5. This relaxation of the reference scenario budgets provides better energy results due to the less accurate slack estimations in the PID controller. However it also presents worse performance results due to the increase of the clustering overhead. In Figure 6.22 we present relative trade-offs between system and adaptive scenarios for several relaxation factors (0-10%) and one noise instantiation (μ=5×10^4 and λ=5×10^3). We see that the Pareto curves of the two approaches seem to be very close and presenting similar cost trade-offs (delay-energy) for a specific constraint area ($0.04 < DVF < 0.08$). In fact, the adaptive scenarios approach appears just a little worse trend (longer distance from the start of the metric axes) because of the extra, required implementation cost. We can realize that the adaptive scenario approach is of high value when strict dependability constraints are essential; clearly, the static approach is unable to meet such demands. Nevertheless, when timing constraints relax ($DVF > 0.04$), the static

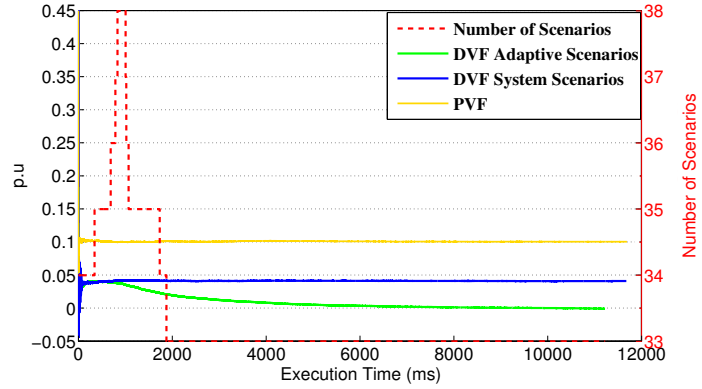


Figure 6.18: Performance adaptability instance.

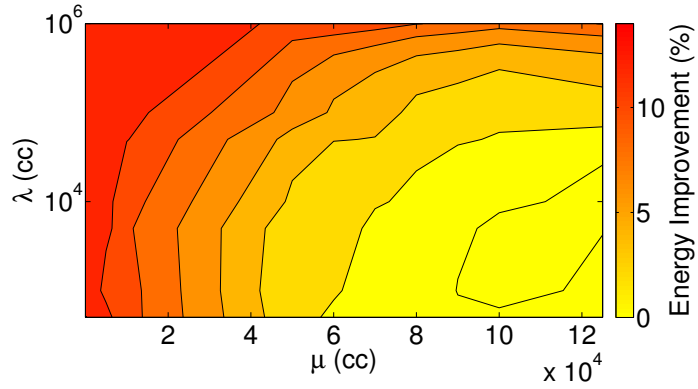


Figure 6.19: Energy gain (%): Adaptive scenarios vs frequency guardband.

scenarios scheme appears to be Pareto-optimal and should be utilized instead.

$$Cl'_{RTS[i]} = \sum_{i=0}^{n} (1 + F_{rel}) \times MRC_{RTS[i]} - OP_{RTS[i]} \tag{6.5}$$

6.6 Conclusions

In this Chapter, we have briefly presented the theoretical background on system scenarios while proposing, in Section 6.2, an enhanced version of the PID

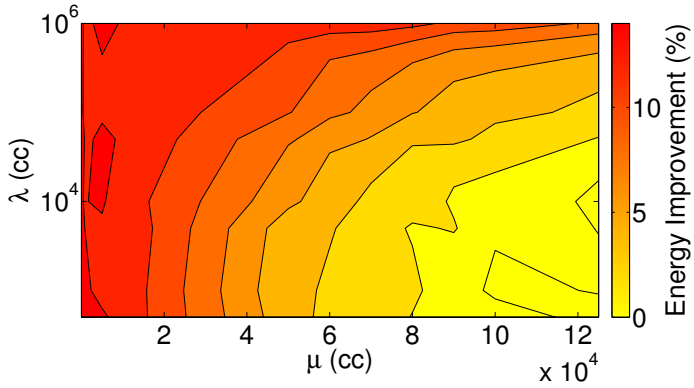


Figure 6.20: Energy gain (%): System scenarios vs frequency guardband.

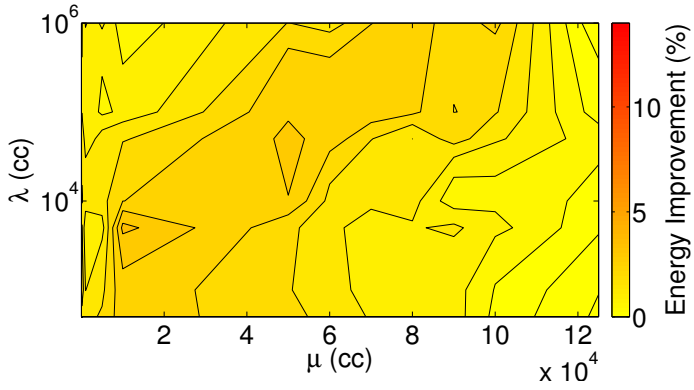


Figure 6.21: Energy gain (%): System vs Adaptive Scenarios.

controller, that actuates DVFS on thread node granularity. Next, in Section 6.3, we have verified the operation of our scheme using several cycle noise profiles. For simulation purposes, we use the Intel Core2 Duo E6850 processor which had sufficient and detailed data, documented in prior art. Section 6.4 discusses our adaptive scenario approach that allows the re- clustering of the scenario hierarchy at run-time. We elaborated on the re-clustering methodology while studying, in detail, relative re-clustering decisions. Finally, in Section 6.5, we have simulated a PID implementation working with adaptive scenarios and compared the results versus:

1. a controller working with system scenarios and

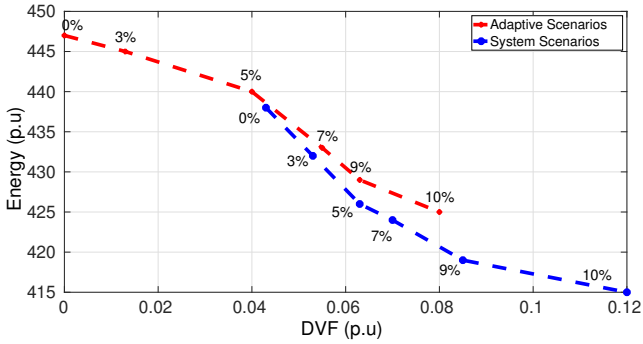


Figure 6.22: Performance (DVF) Vs Energy trade-offs.

2. a frequency guardband policy.

Simulation results have proven the efficiency of the adaptive scenarios scheme both in terms of dependability management and energy consumption.

Chapter 7

Timing Guarantees with Dynamic Scenarios

7.1 Introduction

A reactive scheme to manage dependable performance has been extensively studied in Chapter 5. In Chapter 6 we have shown an enhancement of this controller using system and adaptive scenarios. These versions of the scheme can operate at the thread node granularity and mitigate performance variation in a more efficient manner. Nevertheless, timing constraints are still not guaranteed, mainly because of the reactive nature of the controller: first, a negative slack value is monitored and afterwards, proper DVFS is actuated to converge slack to zero. The general methodology adopted by computer architects to guarantee dependable systems is the worst-case design approach [105]: reliability features are ensured at design-time, assuming worst-case values for system characteristics such as workload, process variation, ambient temperature etc. This approach is pessimistic and adds large safety margins therefore, has proven to be quite sub-optimal, especially for aggressively scaled technology nodes.

To remedy this, in this Chapter we will study a proactive DVFS scheme that operates on the thread node level and guarantees timelines in a cost-efficient manner. Specifically, by exploiting partial predictability of the application behavior, we will develop a *dynamic scenario* approach to enable cost-effective DVFS decisions. Section 7.2 will first mention related, proactive DVFS algorithms used in a task-level basis, along with an illustrative example. Section 7.3 will elaborate on our dynamic scenario methodology. For experimental

purposes, we will use two applications: i) the spectrum sensing app, which will be the main focus of Section 7.4 and ii) a decoder drawn from the audio domain, studied in Section 7.5. Finally, Section 7.6 will conclude our Chapter.

7.2 Guaranteeing Dependable Performance

7.2.1 Related Work and Task-Level DVFS Algorithms

To account for Process, Voltage and Temperature (PVT) variations, memory vendors assume worst-case conditions and demand substantial design margins [284]. While product yield is increased with this approach, memory access latency is kept slow therefore contributing to the so-called *memory-wall* problem [163]. Apart from memory components, most circuit designs are verified using worst-case conditions in order to guarantee performance constraints under variability threats [44, 76]. This analysis method has also proven to be quite efficient in terms of design effort and computational time. Nevertheless, major drawback of worst-case analysis is the pessimistic approach that leads to large safety margins and over-conservative designs. Taking into consideration the scaling of the voltage supply in the new technology nodes, one can conclude that worst-case designs can no longer be adopted by the electronic industry.

An interesting area of study towards a different direction is the technique of adaptive control [287]. The basic concept is to allow the integrated circuit to adapt to certain working conditions like ambient temperature, energy and workload. We have already studied in Chapters 5 and 6 a feedback controller that employs PID-based DVFS to ensure dependable performance under the presence of stochastic, error-recovery interventions. The controller is based on a reactive response and does not guarantee timely execution of relevant applications nevertheless, it ensures timing violations are minimized while meeting low-power operation demands. In order to provide time guarantees, Lampka et al. [150] discuss DVFS methodologies and scheduling algorithms in hard real-time systems. The work assumes offline profiling of the application tasks in order to account for worst-case time execution analysis. Actually, the general mechanism to handle the hard guarantees always relies on some form of guardbanding based on worst-case analysis at design-time. A survey of scheduling algorithms and solutions built upon feedback-based configurations can be found in prior art [3] while an interesting analysis of existing, hard real-time scheduling policies also exists [71]. A paper that discusses task scheduling and deadline guarantees under an error-recovery context can be found in prior art [276]. Voltage scaling techniques of pre-emptive systems have been recently

proposed [152], [212]. These works apply DVFS and scheduling decisions only in a task-granularity basis.

7.2.2 An Illustrative Example

We have aleady highlighted the importance of applying DVFS decisions on the thread node level. However, trying to deploy the algorithms of Subsection 7.2.1 "as-is" with TNs may be computationally prohibitive since, the conditional branches generate many paths on the data flow level control and create an explosive number of possible RTSs. Thus, it is not realistic to assume run-time identification of each individual RTS; even if such a case occurs, the scheduler would demand significant computation times to identify the RTS and perform DVFS and scheduling decisions. For the above reasons, we proceed and follow the scenario classification of Section 6.2. Therefore, while the methodologies mentioned earlier cannot be employed in the TN-level with their exact conditions, it is interesting to examine their DVFS algorithms.

Specifically, we now discuss the *Cycle-conserving RT-DVS* and *Look-Ahead RT-DVS* algorithms from the work of Pillai et al. [212] that guarantee timing deadlines for a sequence of tasks; we focus on their voltage scaling decisions and slack distribution techniques. Neglecting the fact that both algorithms work on task-level granularity (and also employ task scheduling policies), the *Look-Ahead RT-DVS* algorithm seems to be the most aggressive and cost-effective solution to guarantee deadlines. The concept of this approach is to "try to defer as much work as possible, and set the operating frequency to meet the minimum work to ensure all future deadlines are met". This inherently means that system operation may be forced to run at elevated frequencies later in order to ensure all timing constraints. The general idea is that, in most cases, a task is not completed in a worst-case execution time (WCET) window and workload dynamism is modest. In modern processors however, the difference between the maximum frequency and the nominal is not that broad, at least to an extent that would allow the algorithm to achieve significant energy gains.

Next, studying an online DVFS scheme presented earlier [152], we find a *Worst-Case Ready Queue* (WCRQ) algorithm. Authors manage to meet deadlines working in task-level granularity (future tasks reside in the scheduler's queue), while ensuring minimum power consumption. Operating frequency is set to the lowest point that guarantees timely execution of a task. If such a frequency is not found, system operates at f_{max} and speeds up earlier jobs, one job at a time. Then, the frequency is re-decided based on the algorithm. Finally, in terms of DVFS, the principle of *Cycle-conserving RT-DVS* is rather simple: assume WCET for all tasks and when one task is finished after *cc* cycles (<

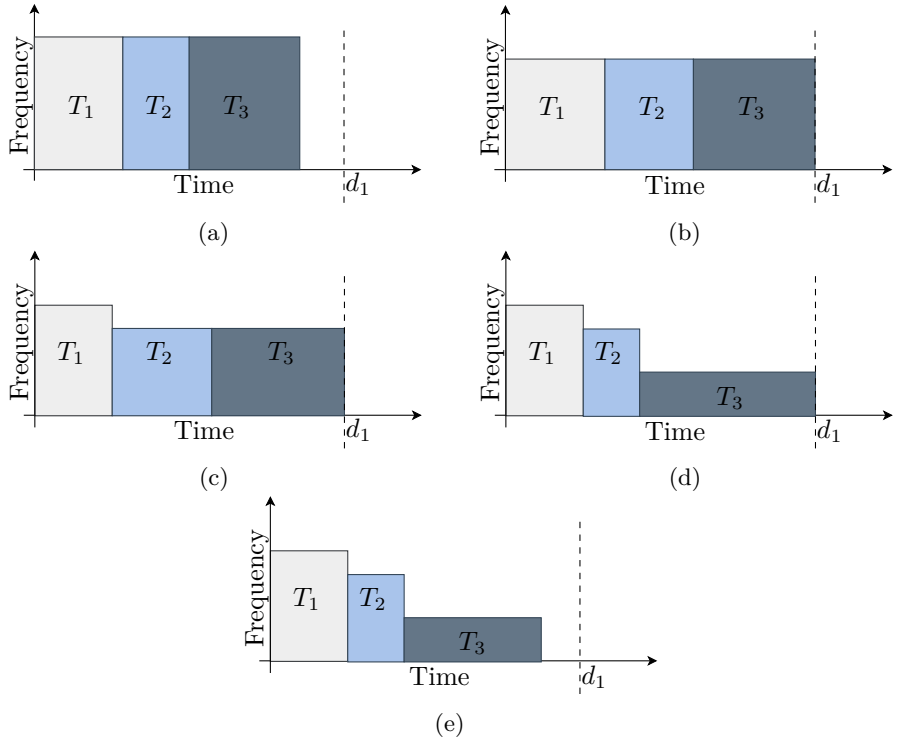


Figure 7.1: Example of the cycle-conserving RM algorithm: a) WCET is considered for all tasks and deadline is set; b) Optimal frequency is selected based on the deadline; c) After the execution of task 1, slack is redistributed equally to the upcoming tasks and frequency decreases accordingly; d) With the completion of task 2 enough slack exists to lower the frequency even further without violating the deadline d_1; e) all tasks have been completed and d_1 is guaranteed in every step.

WCE cycles), saved slack is averaged out until the deadline while actuating proper DVFS and relaxing the frequency. An example of this algorithm is presented in Figure 7.1. In our methodology, we will follow a similar approach mainly because of its simplicity nevertheless, other algorithms highlighted in this Section could also be integrated.

The PID-based, DVFS methodology that manages timing deadlines in the presence of error-recovery interventions has been thoroughly presented earlier in Chapters 5 and 6. This method adopts the scenarios concept and can work on the TN level, however it is built upon a reactive response and thus cannot

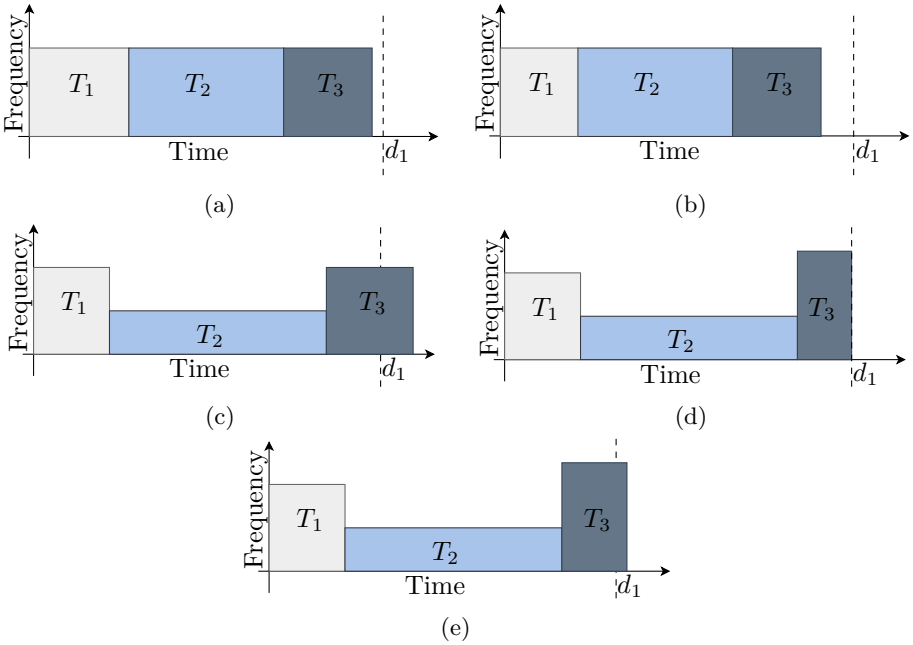


Figure 7.2: Example of the PID-based DVFS concept: a) WCET is considered for all TNs and deadline is set; b) The execution of the TN1 requires less time than WCET hence, slack increases; c) The controller relaxes the frequency and now TN2 requires a longer time window; TN3 is not guaranteed; d) With the completion of TN2, the controller now boosts frequency according to the PID formulation and deadline is managed for TN3; e) in a scenario that TN3 lasts longer, TN3 may miss the deadline; timing constraints are not guaranteed.

provide guarantees on timing deadlines. After estimating how far behind or ahead we are from our schedule, DVFS switches are actuated from a closed-loop, PID controller. An example is illustrated in Figure 7.2, presenting the technique's response in the task level. We can realize how timing dependability is managed while guarantees are not provided. The power penalty of this method, however, is minimal hence, it will be used in this Chapter as a baseline technique to compare with.

7.3 Dynamic Scenarios: A Proactive DVFS Scheme

To guarantee timing constraints, we need to ensure that application execution finishes before the deadline even at worst-case conditions. Since we are operating at the fine-grain level of TNs, we first consider WCET for all TNs, along with their respective WC execution cycles. Assuming that cycle noise is bounded, we can proceed with determining WC cycles per each TN in a sense that cycle budget along with cycle noise will never exceed WC cycles. The execution of such TNs under multiple DVFS levels generates a surplus of RTS points, clearly though, this kind of information needs to be abstracted further; the scenarios (see Section 6.2) group these RTSs into clusters with similar cycle budgets. System scenarios now compose a more manageable set of information. The assumption that cycle noise can be bounded is based upon the fact that device degradation models actually allow to put absolute bounds; the Gaussion distributions are only sufficient approximations and in reality, the tails have cut offs [104]. The same concept applies for models describing wire degradation and power supply noise. For the architecture layer of an embedded system, we can also assume that realistic workload simulations (on long workloads) are able to show the bounds on how much cycle overhead we can expect maximally for a given application. Hence, we can safely provide bounds on the cycle noise.

The label describing each scenario accounts for the RTS with the highest cycle budget among the cluster. Hence, assuming WC cycles of this RTS, we can make sure that there exists no RTS in a scenario where WC cycles of the RTS are higher than WC cycles of the scenario tag. On the contrary, due to the clustering overhead we now have that $WC_{SCE} = WC_{RTS} + D$ where D represents the overhead. Finally, we employ a DVFS scheme, the *Cycle-conserving RT-DVS* policy in our case, to guarantee timely RTS execution. Clearly, DVFS is now occurring in a TN granularity. Just as in the case of previous task-level DVFS algorithms, partial predictability of the system can occur through a buffer containing future TNs for example. Such predictors already exist in video image processing and computer vision applications. In addition, using machine learning [267] or probabilistic predictors [279] one can also estimate, to an extent, future system behaviour.

Assuming a $Buffer$ with size S, containing future TNs (and hence their respective cycle budgets N), we can present the DVFS Algorithm 2. Our algorithm contains three main procedures: at first, the running RTS is chosen from the start of the $Buffer$ while a scenario detector matches it with its cluster based on the cycle budget. Then, the $Buffer$ is updated by allowing a left shift (SAL) of the stored TNs and placing the incoming, future TN at the $Buffer$'s end position. After this, we estimate the slack based on Equation 6.3. Finally, we select the optimal frequency that can guarantee timing deadlines

Algorithm 2 DVFS Algorithm

1: Update_Buffer:
$N[n] = Buffer[start]$
$WC_{SCE}[n] =$ detect_scenario$(N[n])$
SAL $Buffer$ {Shift Buffer to the left}
$Buffer[end] = rts_{n+s}$ {Update Buffer with incoming RTS}

2: Estimate_slack:
$slack[n] = \frac{WC_{SCE}[n]}{f_{ref}} - \frac{N[n]+x[n]}{f[n]} + slack[n-1]$

3: select_frequency(f):
$d = \frac{\sum_{i=1}^{s} WC_{SCE}[i]}{f_{ref}} + slack[n]$
use lowest freq. $f \in \{f_1, ..., f_m | f_1 < ... < f_m\}$
such that $\frac{\sum_{i=1}^{s} WC_{SCE}[i]}{f} \leq d$

and at the same time ensure minimum energy budget. The slack is added to the respective deadline d and the minimum frequency is chosen so that timely execution is guaranteed, assuming WC_{SCE} cycles for future TNs. Again, we should emphasize the difference between WC_{RTS} which represents the worst-case cycle estimation for an RTS and WC_{SCE} that are the worst-case cycles of the scenario cluster. This buffer can be built through any workload prediction mechanism. It is also important to distinguish between system scenarios and dynamic scenarios: in both cases application RTSs are classified into scenarios however only the latter enables a proactive operation by exploiting workload prediction and future slack stealing [188].

7.4 Utilizing Dynamic Scenarios on the Spectrum Sensing App

To examine the efficiency of our dynamic scenarios approach we will now proceed with simulations and board-level experiments. The first application for our experiments is the spectrum sensing app which we have already seen in Chapters 5 and 6. It performs heavy digital signal computations and power analysis of GSM bands and is relevant for software defined and cognitive radios. Since the application contains a significant number of RTSs, which we have grouped in scenario bins, it can be considered as a representative case study. In addition, we choose to model the Intel Core2 Duo E6850 processor (see Table 6.1), following accurate data from previous works [203]. Energy consumption is estimated

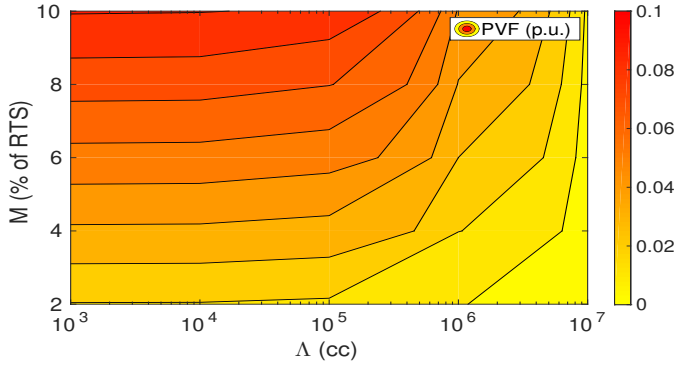


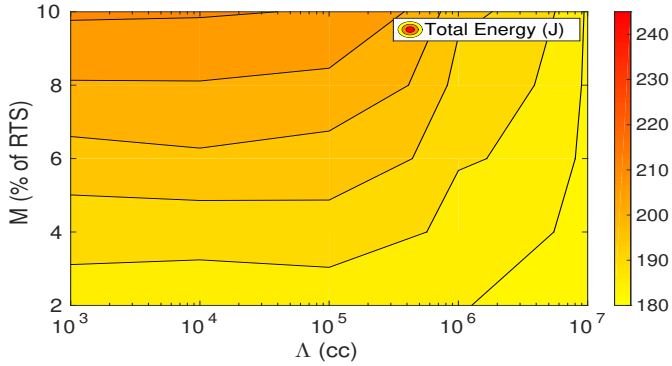
Figure 7.3: The PVF values for different cycle noise profiles.

taking into consideration both static and dynamic power as well as DVFS energy overheads; timing overhead of DVFS switches is also examined.

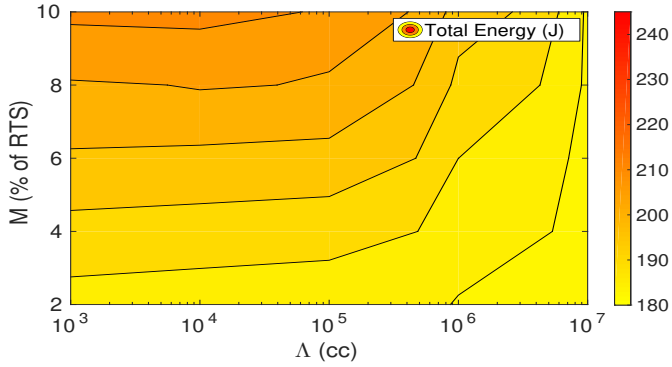
7.4.1 Simulation Results

For our simulation, we will compare our dynamic scenario scheme, against the PID-enabled DVFS and the frequency guardband methodologies. This comparison will be studied in terms of overall energy consumption and dependability; meeting, that is, timing constraints. Dependability will be viewed through the DVF term. We will examine our proactive approach and the other two methods for different cycle noise configurations, with $\Lambda \in [10^3, 10^7]$ cycles and M ranging from 1% to 10% of RTS cycles. For simulation purposes, we assume a dominant error-recovery mechanism hence, cycle noise is modeled to follow a Gaussian distribution with μ and λ values equal to M, Λ. Nevertheless, other noise profiles can also be simulated without changing the nature of the output. The sigma values of the distribution are equal to $\sigma_\mu = 200$ and $\sigma_\lambda = 100$ cycles respectively. Increased M and lower Λ values lead to more aggressive cycle noise profiles, as can be seen in Figure 7.3. In our graphs, we will present average results after 50 application runs per each cycle noise profile.

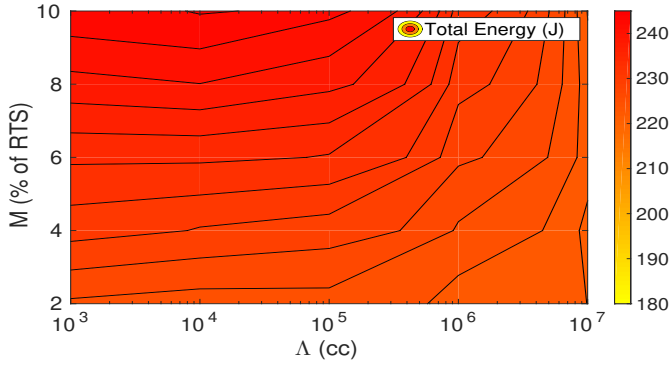
Next, in Figure 7.4, we present average total energy estimations for the above noise profiles. We can verify how the static frequency guardband is highly suboptimal in terms of energy consumption while our proactive DVFS scheme shows the same energy costs with the reactive controller-based DVFS configuration. In fact, both proactive and PID-enabled DVFS schemes present a 23% energy gain compared to the frequency guardband approach. Nevertheless,



(a)

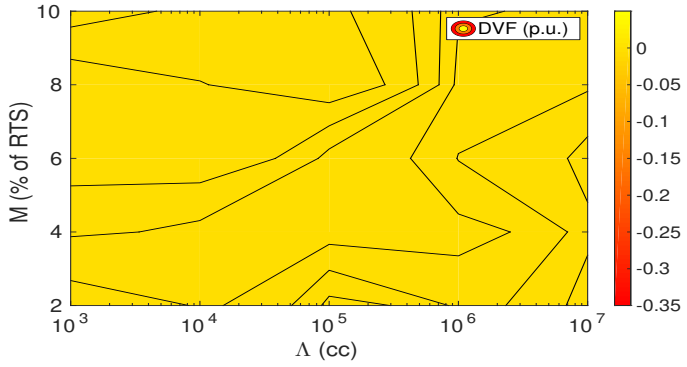


(b)

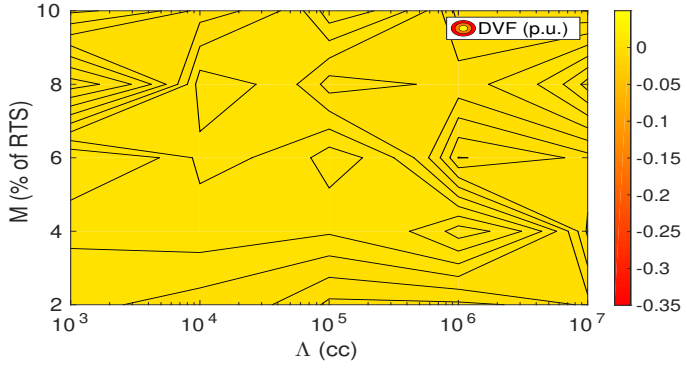


(c)

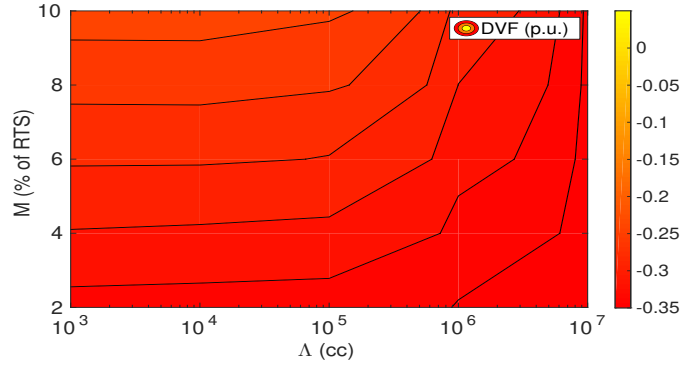
Figure 7.4: Total Energy estimations for the three schemes: a) dynamic scenarios; b) PID controller-based; c) frequency guardband.



(a)



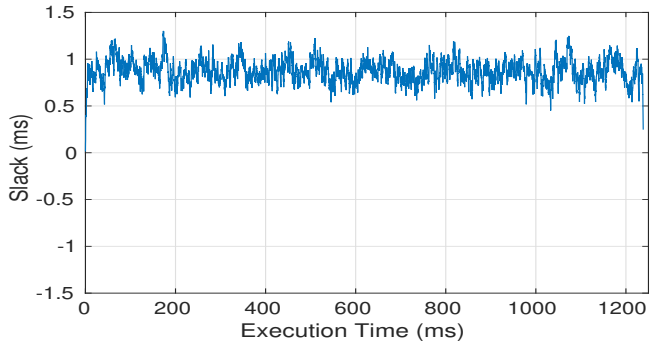
(b)



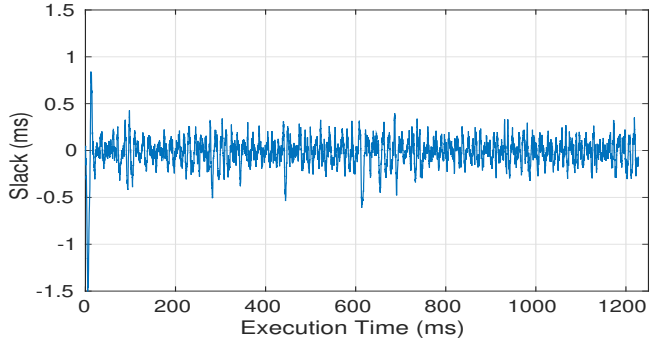
(c)

Figure 7.5: DVF estimations (per unit) for the three schemes: a) dynamic scenarios; b) PID controller-based; c) frequency guardband.

through our scheme we are also able to provide dependability guarantees. For the frequency guardbanding method, the static frequency was set at the processor's first DVFS level.



(a)



(b)

Figure 7.6: Slack progression for the two DVFS schemes: a) dynamic scenarios; b) PID controller-based DVFS. While in the proactive scheme slack never reaches negative values, in the controller-based DVFS configuration slack often drops to negative levels.

In Figure 7.5, we show the DVF values for the 3 different configurations. We understand how for the frequency guardband, DVF reaches significantly negative values. This translates to unnecessary energy losses since to reach our deadline in time we only need to be slightly below zero. In addition, both controller-based and proactive DVFS implementations manage to keep DVF value near zero for the complete noise configurations. While this is notable and shows the efficiency of both schemes, we should emphasize that only our proactive scheme is able to guarantee DVF being below zero throughout the application execution (and

hence we will never miss the deadline). In contrast, the PID controller often misses the deadline. Figure 7.6 presents the slack for one application run when the two schemes are deployed. While our proactive approach ensures that slack will never reach negative values (as this means that we are behind our schedule), in the PID controller's case, slack often drops to negative levels.

7.4.2 Buffer Size Impact

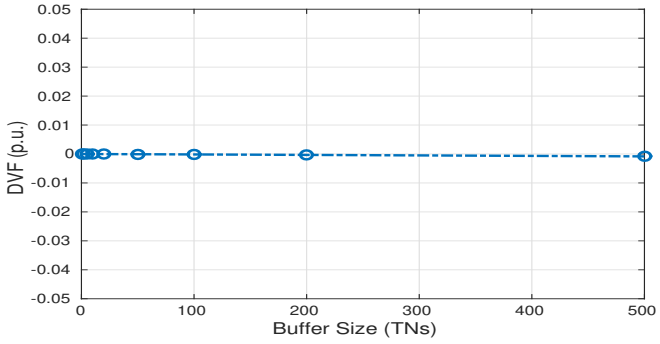
The buffer size plays an important role in this case study. It represents actually how far ahead the application's execution characteristics can be predicted. To quantify this effect, we choose to simulate a number of buffer sizes for our scheme and estimate the final DVF value as well as the total energy consumption. Obviously, larger buffer sizes can allow a more efficient management of the application's timing demands. This statement seems quite reasonable; the more information we have regarding the future operation, whether this may be through the scheduler's queue (as in our example) or through other predictor configurations, the easier it is to exploit this information and ensure/guarantee system dependability. Figure 7.7a presents the aforementioned results regarding the DVF. As can be seen from this graph, regardless the buffer size, a timing guarantee ($DVF < 0$) is always reached because of the characteristics of our algorithm.

Nevertheless, when examining the energy costs for the different buffer sizes we see how larger buffer sizes lead to lower total energy consumption. Specifically, in Figure 7.7b we see that when the buffer contains a small number of future TNs, energy cost can reach up to 9% compared to the optimal buffer size of 50. On the contrary, after a certain degree, in our case a size of 50, increasing the buffer translates to no further energy gains. This clearly demonstrates that we do not require large costly buffers to make our approach function efficiently.

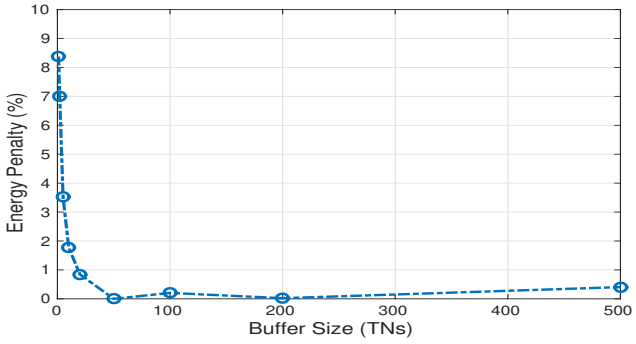
7.4.3 Board-Level Experimental Verification

To verify our approach on pure hardware, we utilize the NXP IMX6Q board (see Section 5.2). This commercial board is widely used by developers for multi-core processing purposes, low-power consumption and multimedia/graphics applications. The current DVFS configuration, enabling the usage of 14 operating points, is presented in Table 7.1[1]. Reference frequency is set at 792 MHz.

[1]Our technique can seamlessly be deployed on any processor with similar DVFS features.



(a)



(b)

Figure 7.7: Dependability and energy costs for different buffer sizes: a) DVF value; regardless buffer size, our scheme can guarantee timing dependability; b) Energy penalty (%) for different buffer sizes, compared to the optimal choice of a buffer size 50. Small sizes lead to increased energy costs up to 9%.

Table 7.1: DVFS Levels of the ARM Cortex A9 Processor for our Experiments.

DVFS Level	**Volt. (V)**	**Freq. (GHz)**	**DVFS Level**	**Volt (V)**	**Freq. (GHz)**
Level 1	0.975	0.396	Level 8	1.150	0.744
Level 2	1.000	0.444	Level 9	1.175	0.792
Level 3	1.025	0.492	Level 10	1.200	0.840
Level 4	1.050	0.544	Level 11	1.225	0.888
Level 5	1.075	0.588	Level 12	1.250	0.936
Level 6	1.100	0.636	Level 13	1.275	0.948
Level 7	1.125	0.696	Level 14	1.300	0.996

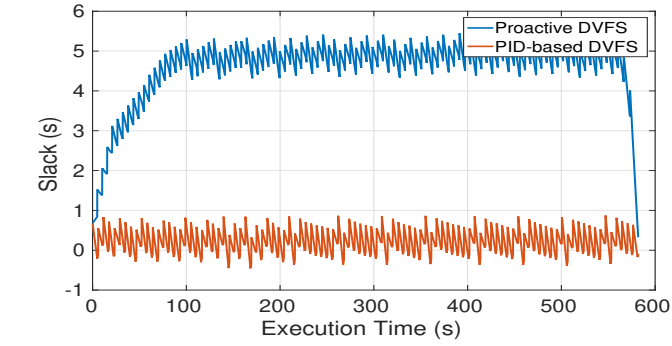
Regarding the application, we will utilize a part of the spectrum sensing app,

performing low-sensitivity, digital signal processing[2]. For this experiment, the application consists of 3 different RTSs, clustered in 2 scenarios. We build the buffer, containing up to 10 future RTSs, and run the application comparing our proactive DVFS technique versus the PID controller-based DVFS method. Cycle noise is injected through the `usleep` command of the Linux library. Noise is again following a Gaussian distribution with $M[n] \sim \mathbf{Norm}(\mu, 0.1 \times \mu)$ and $\Lambda[n] \sim \mathbf{Weib}(\lambda, 1)$. In this case study, $\mu = 0.05 \times N[n]$ and $\lambda = 10 \times N[n]$ where N represents the cycle budget of a defined TN. Again, cycle noise is bounded and $x[n] < 0.1 \times N[n]$. Concerning the controller, gains are tuned as $k_p = 25 \times 10^{-2}$, $k_i = 1 \times 10^{-3}$, $k_d = 3 \times 10^{-2}$.

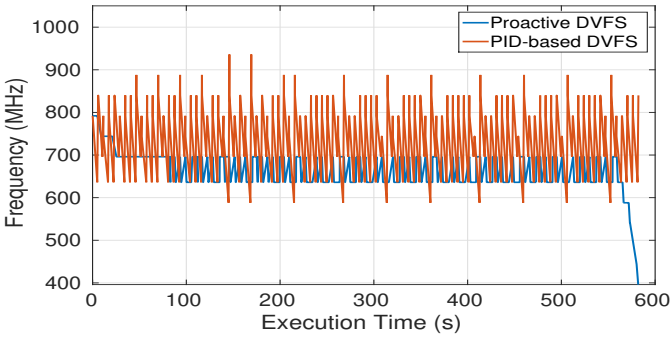
Slack progression for the two test cases is presented in Figure 7.8a. We can verify how for the reactive, controller-based, DVFS methodology timing slack converges to zero. However, it often drops to negative values meaning that, at certain instances, timing constraints are not met. Momentarily therefore, application execution is behind schedule and the controller boosts operating frequency and drives slack to positive values to meet the deadlines. On the contrary, the slack progression of the proactive DVFS technique is always positive meaning that timely application execution is accomplished at any moment during the run. This is expected of course since our proactive approach is guaranteeing the timing demands. Interestingly, the graphs of the slack progression for the two test cases generated from these experiments on the target board are similar in nature with the ones predicted though the simulation modelling in Subsection 7.4.1, at Figure 7.6. Frequency decisions for the two cases are shown in Figure 7.8b. Due to lack of information, energy estimations with these experiments are not presented[3].

[2]In that respect, we are not utilizing the complete application as presented in Subsection 7.4.1. On the contrary, we are deviating from these simulations and assume the existence of only 3 TNs clustered in two different scenarios. In this Subection, our purpose is to emphasize on the responsiveness of the proactive DVFS methodology using DVFS capabilities provided by a commercial platform running a standard Linux installation. The complexity reduction and the degree of optimality offered by scenario classification has been explained earlier and was also covered in Subsection 7.4.1.

[3]We were not able to perform energy estimations in this case, due to lack of technical information. Searching though the board's and processor's data sheets we did not find enclosed information relevant to power and energy data for our DVFS levels as well as the energy overhead of DVFS.



(a)



(b)

Figure 7.8: Experimental results on the NXP board for the Dynamic Scenarios and PID-based DVFS approaches: a) Slack progression; b) Frequency decisions.

7.5 Utiling Dynamic Scenarios on the MP2 Decoder

Occasionally, the use of a scheduler queue to foresee future workload and scenarios can be impractical; a simplistic operating system on an embedded platform for example, may not provide such a feature. In addition, even if this feature exists, sometimes the user may not have the administrative rights to read from the queue. In this case, usage of our proactive DVFS methodology alone seems somehow limited. To this end, we will now focus on another case study, a small MPEG-1 Audio Layer II (MP2) decoding library written in C that is widely used in several audio applications. Here, the predictor is not based on a scheduler's queue; workload prediction is enabled based on information

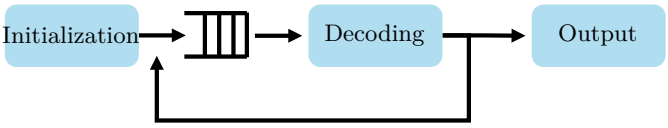


Figure 7.9: Flowchart of the application: the decoding procedure, is repeated in chunks until the entire bitstream, drawn from the buffer, is processed. The application reads the input file at run-time, updating the buffer.

extracted from the application profile and the input files.

7.5.1 Application 2: An MP2 Decoder

In this case study we choose a simple, MP2 decoding application available online [88]. MP2 decoding, in general, is part of numerous audio and video players. We chose this format since it is the simplest audio compression scheme still in broad use today; Layer III (MP3) and other formats are considerably more complex. Therefore, a TN analysis followed by a scenario clustering classification of the complete audio application is outside the scope of our work; Performing the scenario analysis for the aforementioned library we find 5 TNs clustered in 2 scenarios. MPEG Audio Layers I and II are built around a filter that transforms 32 consecutive time-domain samples into 32 frequency-domain values (subband samples). This process is performed over a 512-sample window. 36 of these 32-sample runs are packed together in one 1152-sample frame, which is the smallest atomic data unit in the stream that can be decoded independently.

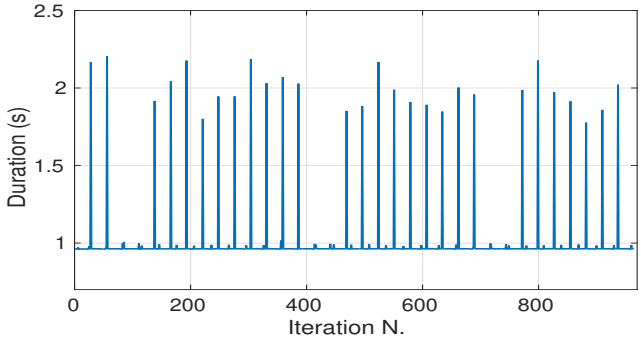


Figure 7.10: Dynamism in the execution time of the decoding process for different bitstream chunks of Vivaldi's "Four Seasons" Concerti.

The decoder is realized as a main loop that is executed iteratively, reading bitstream frames, decoding them and writing the output. With such an application we can partially predict incoming TNs without using the feature of a scheduler queue: after profiling the application, we understand how the decoding process is depending on the input bitstream and the sample window. A depicture of the decoder application flowchart is shown in Figure 7.9. It is also important to note the inherent dynamism of this application since different bitstream frames, need various computation times. This dynamism leads to the generation of a plethora of RTSs that cannot be taken individually into account; on the contrary, it highlights the need for scenario classification. An example of this dynamism for the “Four Seasons" violin concerti is shown in Figure 7.10. On top of this bitstream-dependent inherent dynamism, the error-correcting overhead will be added hence, scenario clusters are designed accordingly, to account for both sources of dynamism.

7.5.2 Simulation Results

We will simulate the application running on the Intel Core2 Duo E6850 processor, as we have already performed for the GSM spectrum sensing application in Section 7.4.1. Our proactive DVFS methodology will be compared to the frequency guardband approach and the reactive, PID-enabled DVFS framework in terms of total energy consumption and DVF estimations. In this case, regarding cycle noise configurations, $\Lambda \in [10^3, 10^9]$ cycles and M ranges from 1% to 10% of RTS cycles. A dominant error-recovery mechanism is also considered hence, cycle noise is modeled to follow a Gaussian distribution, as in the GSM

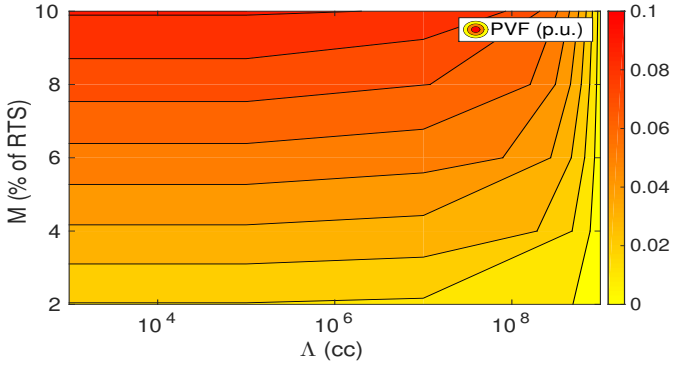


Figure 7.11: The PVF values for different cycle noise profiles.

application. PVF values for the noise profiles of our simulations are presented in Figure 7.11. Again, graphs show average values of 50 application runs.

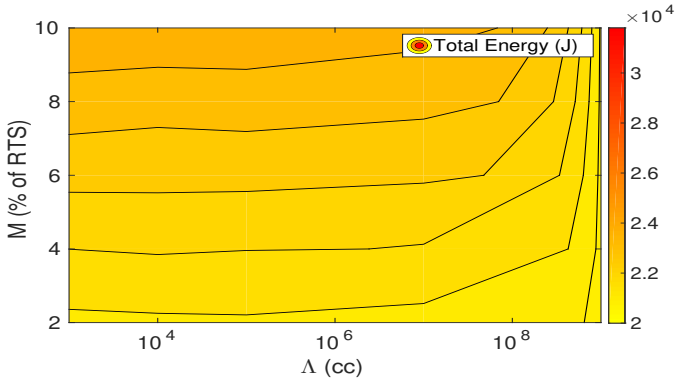
Total energy estimations for the 3 distinct technologies are shown in Figure 7.12. We observe how these results are analogous to the ones extracted from simulation runs of the spectrum sensing application. Specifically, the frequency guardband methodology is the most energy-expensive approach while the proactive and the PID-enabled DVFS techniques present equal energy costs. In this application, energy gains reach up to 28% for the proactive DVFS framework compared to the static, frequency guardband implementation. Again, for the latter approach, the highest DVFS step was selected as guardband.

Results of the DVF metric for the MP2 decoding application are highlighted in Figure 7.13. Once more, we confirm how our proactive DVFS scheme is able to guarantee timing deadlines without unnecessary frequency boosts and energy penalties, in contrast to the guardband approach. Slack progression for the proactive and reactive DVFS schemes is presented in Figures 7.14. As expected, these graphs present a similar trend to the ones obtained from the GSM application. We can observe how in the case of the proactive DVFS approach, slack never reaches negative values. On the contrary, when testing the reactive, PID-enabled DVFS implementation, deadline misses can be noticed since slack drops far below zero.

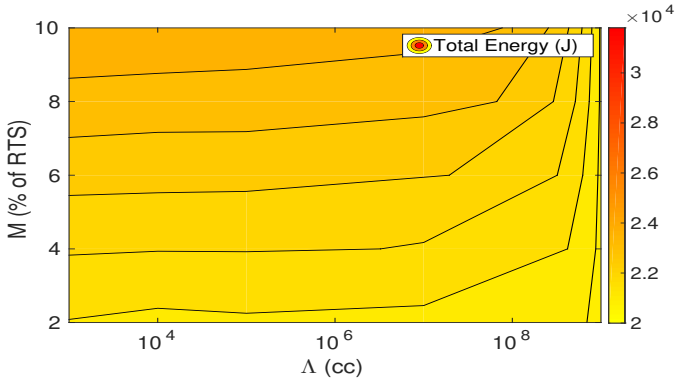
7.5.3 Buffer Size Impact

Through simulations, we will now examine the impact of different buffer sizes in the efficiency of our scheme using the decoding application. As seen in the flowchart of Figure 7.9, after the initialization phase, the decoder splits the audio bitstream into workload frames of a specific size. In our case, the decoder is used *statically* since the complete input stream is first stored into a file (an audio song for instance). However, for real-time decoding applications, these frames can be stored in a memory buffer, just after they are pre-fetched, waiting to be processed like for example in the H.264/AVC video decoder [96]. It is important to deploy our proactive DVFS framework so that timing deadlines are guaranteed regardless of the buffer size. This is shown in Figure 7.15a; we verify how for a buffer storing a number of frames (ranging from 1 up to 500) all deadlines are guaranteed and DVF never drops to negative values. These results are inline with the ones of Subsection 7.4.2, concerning the GSM spectrum sensing application.

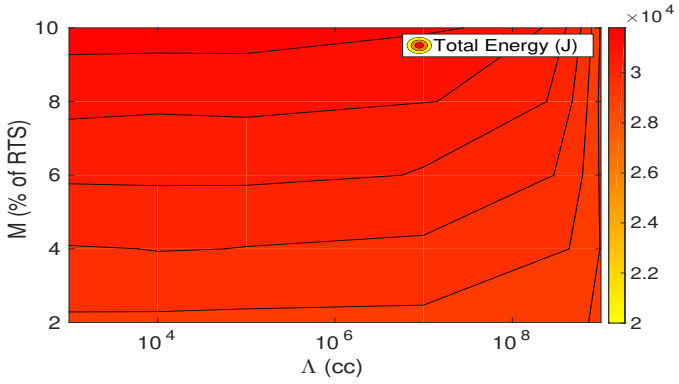
The size of the buffer as well as the size of the frame itself can vary, depending on QoS and latency requirements. While we have shown that for all buffer sizes deadlines are managed, it seems evident that more opportunities for



(a)

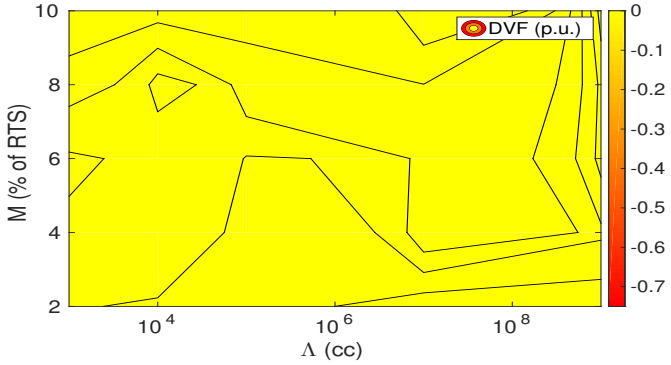


(b)

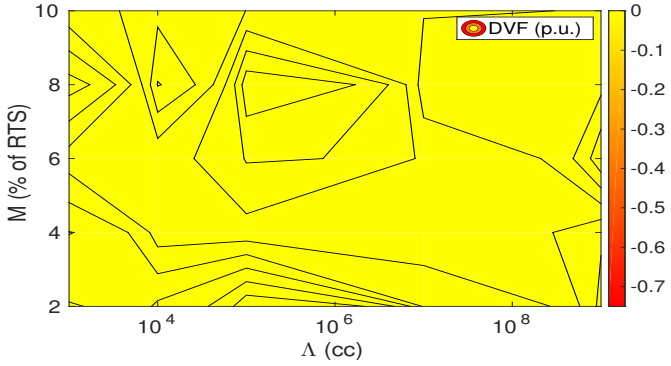


(c)

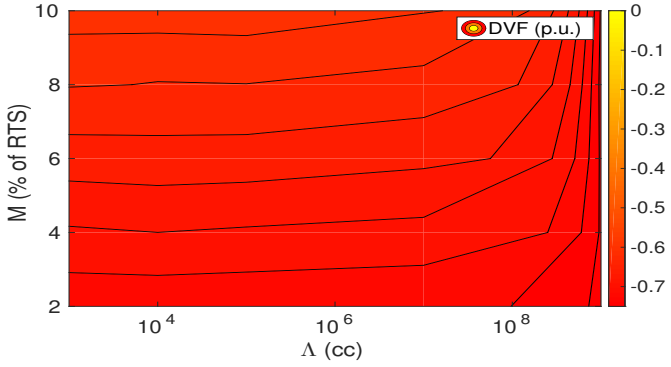
Figure 7.12: Total Energy estimations for MP2 decoding app: a) dynamic scenarios; b) PID controller-based; c) frequency guardband.



(a)

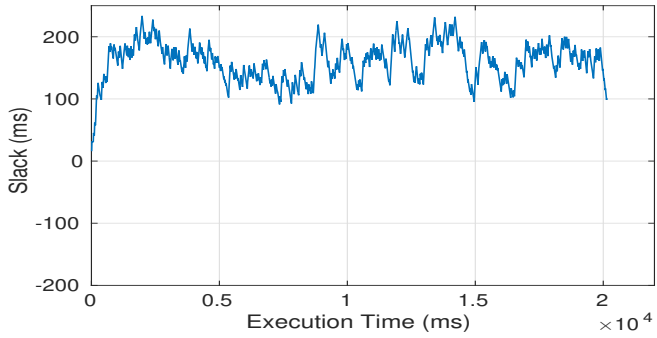


(b)

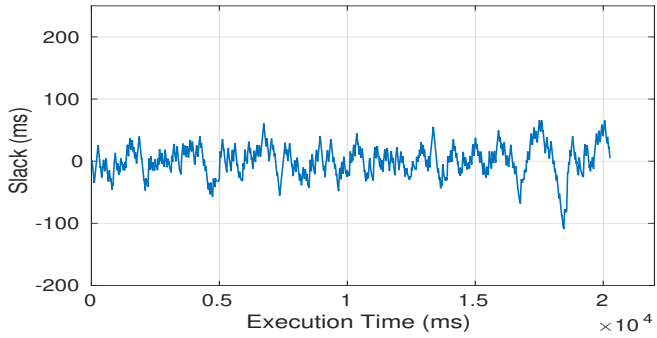


(c)

Figure 7.13: DVF estimations (per unit) for MP2 decoding app: a) dynamic scenarios; b) PID controller-based; c) frequency guardband.



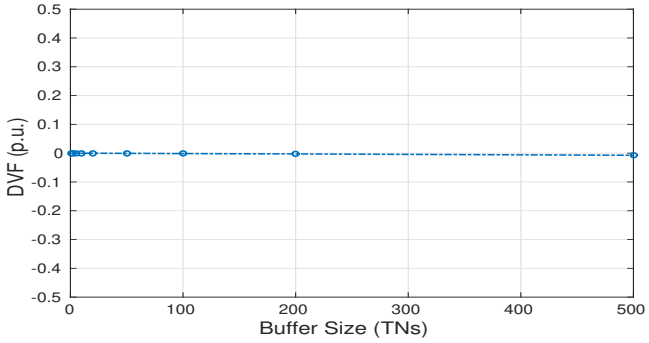
(a)



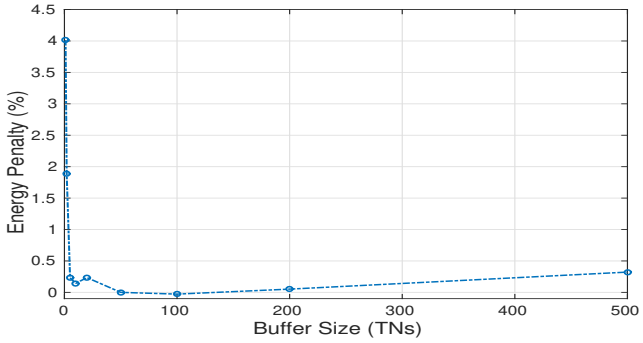
(b)

Figure 7.14: Slack progression for the two DVFS schemes in the case of the decoding app: a) dynamic scenarios; b) PID controller-based DVFS. While in the proactive scheme slack never reaches negative values, in the controller-based DVFS configuration slack often drops to negative levels.

optimal, proactive voltage and frequency scaling exist when higher buffer sizes are selected; specifically, higher energy costs should be expected as the buffer size shrinks since this would lead to a more aggressive voltage and frequency scaling policy. For the decoding application, energy costs for different buffer sizes are estimated via simulations and illustrated in Figure 7.15b. Results for the two applications in fact present a similar trend: for small buffer sizes energy costs to guarantee timing deadlines are elevated nevertheless, the energy gains after stretching the buffer size quickly saturate. An ideal buffer for the decoding application seems to be of a size of 50 frames. Nevertheless, when a



(a)



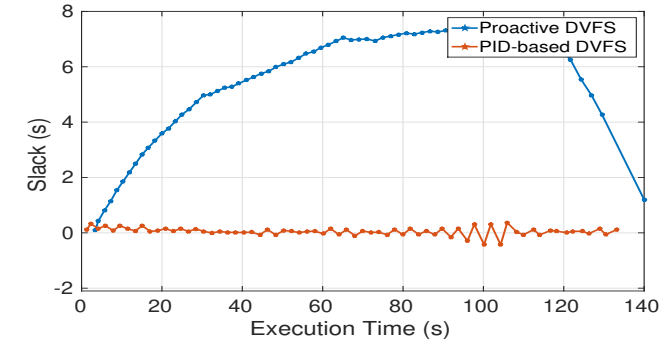
(b)

Figure 7.15: Dependability and energy costs for different buffer sizes in the case of the decoding application: a) DVF value; regardless buffer size, our scheme can guarantee timing dependability b) energy penalty (%) for different buffer sizes, compared to the optimal choice of a buffer size 50. Small sizes lead to increased energy costs up to 4%.

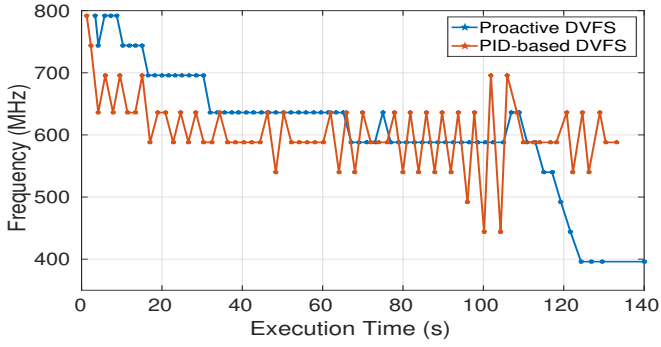
buffer contains nearly 10 future frames, the energy costs do not exceed 0.5%. In fact, when only the next frame to be decoded is known (buffer size is 1) the energy penalty to guarantee the deadlines is almost 4% compared to the energy cost for the ideal buffer of size 50.

7.5.4 Board-Level Experimental Verification for Application 2

We will deploy this technique on our NXP IMX6Q board, decoding the “Fur Elise" solo piano of Beethoven. During the *Initialization* phase of the decoder, future workload of the application is unknown at the moment and no proactive



(a)



(b)

Figure 7.16: Experimental results on the NXP board for the Dynamic Scenarios and PID-based DVFS approaches in the case of the decoding application: a) Slack progression b) Frequency decisions.

decision can be made. When the MP2 input file is selected, the application reads the bitstream, filling the buffer and updating it at run-time. During the *Decoding* stage, the proactive DVFS methodology is employed and timing constraints are guaranteed. Again, through the `usleep` command we inject cycle noise, emulating the overhead of any RAS-related intervention mechanism (similar to Subsection 7.4.3).

Slack progression during the application is presented in Figure 7.16a where the scheme is compared again versus the reactive, PID-enabled DVFS configuration. One can notice that slack is behaving in a manner similar to what we have already observed in previous Sections. During the *Decoding* phase, that accounts for most of the execution time, the proactive DVFS method is employed hence timing constraints are guaranteed. In Figure 7.16b we see the frequency decisions: at first, when the PID is enabled, frequency relaxes momentarily, since positive slack is measured. Then, the input file is selected allowing us to foresee future workload and utilize the proactive DVFS technology.

7.6 Conclusions

In this Chapter we elaborated on the dynamic scenarios methodology and implemented a proactive DVFS scheme that aims to guarantee application timing constraints. Firstly, in Section 7.2, we have mentioned several related works that attempt to ensure dependability in the task-level granularity. In addition, we have presented an illustrative example of such a DVFS algorithm. Then, in Section 7.3, we have explained in detail the dynamic scenarios approach that is the foundation of our proactive DVFS scheme. Another important feature of our scheme is the workload prediction mechanism that enables us to foresee and identify future scenarios. Two case studies have been examined to show how our methodology can be applied to a group of real-time applications drawn from different domains: the streaming spectrum sensing application, in Section 7.4, and an MP2 decoder, in Section 7.5. Both the extensive simulation results as well as the board-level experiments have verified the benefits of our scheme in terms of power consumption and timeline guarantees.

Chapter 8

Conclusion

8.1 Summary of the Main Contributions

After the technical work of the dissertation has been thoroughly presented, we can now summarize and reflect on our research while highlighting the main contributions. Firstly, we have defined performance variation as the unpredictable, nonstop changes of the workload in digital systems. This variation can be the result of modern, computationally-intensive software applications as well as failures and errors on the silicon layer; in the current text, we focus mainly on the latter. We have also elaborated on the utilization of RAS mechanisms and explained how they ensure correct functional operation at the cost however of parametric reliability.

Next, in Chapter 2, we have given a general review on the common hardware failure mechanisms. Voltage supply variations due to di/dt noise and IR drops can impact propagation delay of basic logic gates. Process variation phenomena such as random dopant fluctuations, oxide thickness variation and line-edge roughness also affect the performance of a digital system. On top of time-zero variability, workload-dependent oxide and interconnect wearout degrades the electrical characteristics of circuit elements, threatening functional and parametric reliability violations. Finally, reliability violations can also occur due to radiation-induced transient errors and power/ground line voltage variation.

The main goal of our research is the management of parametric reliability and in particular performance variation. Thus, in Chapter 3 we have studied in detail previous mitigation attempts drawn from the prior art [194]. These works cover, in fact, a wide range of research domains, from design and pre-fabrication

solutions to software schemes of single-die, multi-die systems and systems across packages. We have presented a classification methodology to cluster these works into their respective domains, generating a binary tree in an orthogonal, top-down approach. This classification enables the reader to identify research fields that have received limited attention as well as decompose any prior art samples to their primitives.

Before developing mitigation schemes of performance variation, an accurate description of dominant failure mechanisms is necessary. To this end, we have developed an efficient modeling framework using the Most Probable Failure Point technique to capture RDF and BTI and estimate failure probabilities of typical 6T SRAM cells (Chapter 4) [193]. The tool exploits the symmetry of the design to apply Spectre "in-the-loop", aggressively shrinking the simulation times (compared to widely-used Monte-Carlo experimentations [196]). Moving further, we have developed a complete simulation framework that captures RDF and BTI on a cycle-by-cycle basis and estimates failure probability over lifetime operation [192]. As a case study, we have focused on the SRAM memory of a Network on Chip router.

In Chapter 5 we introduce a closed-loop PID controller that actuates DVFS switches to manage performance variation [197]. We have deployed this scheme on the NPX iMX6Q target platform that contains an ARM Cortex A9 Quad processor. To generate performance variation, we have developed an error-recovery mechanism that detects functional errors through a signature-based approach and recovers correct system operation via task-level rollback. Slack is monitored periodically and when performance variation is observed, proper DVFS switches are decided. We have explored the hardware-related limitations of this scheme while comparing it against another DVFS approach in terms of performance and energy efficiency. Finally, we have briefly presented a similar, PID-based, DVFS scheme for thermal management purposes [198].

Then, we study an enhanced version of the PID scheme that operates on the RTS level (Chapter 6) [195]. Given that modern applications typically consist of numerous RTSs, we have utilized the system scenario approach to group these RTSs into a more manageable set of scenario clusters. Through simulations, we have verified the ability of our enhanced scheme. The scenario clustering takes place offline, during the profiling of the application; a run-time, re-clustering policy through a novel, adaptive scenario approach is also presented [307, 306]. Simulation results comparing this scheme to the system scenarios and to a static frequency guardband technology, highlight the benefit of the former methodology both in terms of performance and energy response.

Finally, we have introduced a proactive DVFS scheme that employs dynamic scenarios and manages to guarantee dependability in certain dynamic systems

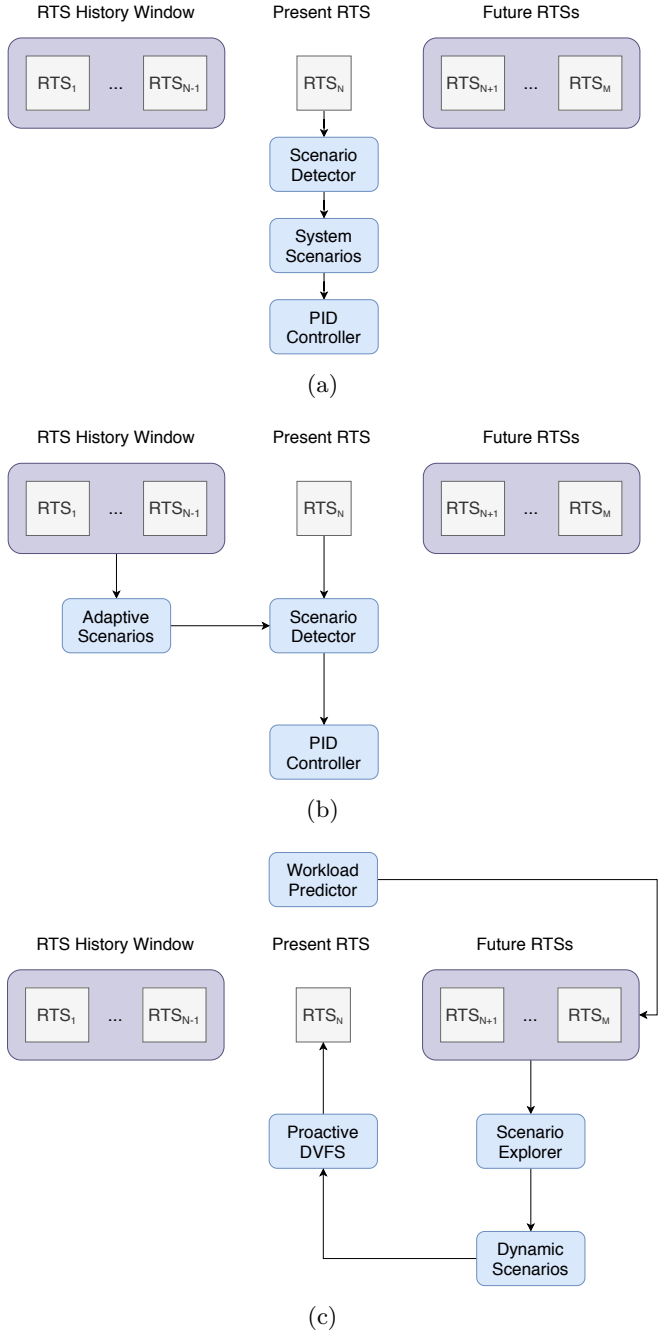


Figure 8.1: An illustration of the System, Adaptive and Dynamic Scenarios methodology: a) System Scenarios; b) Adaptive Scenarios; c) Dynamic Scenarios.

[191]. Application RTSs are first clustered offline into scenarios according to the system scenario methodology. Nevertheless, a workload predictor can now exploit the system's partial predictability and enables us foresee and also identify future scenarios. Simulation results, as well as experiments on the target board, verify our claims and show the efficiency of our proactive scheme against the system scenarios and the static frequency guardband approach. Figure 8.1 underlines the difference between the dynamic, the adaptive the system scenarios concept. In system scenarios, the scenario clustering takes place offline and at run-time, a scenario detector matches the current RTS to each related scenario while the controller decides for DVFS. In the adaptive scenarios, a run-time, scenario re-clustering procedure occurs which is based on the monitored RTS history. Finally, in the dynamic scenarios approach, the workload predictor allows us to foresee future RTSs and apply proactive DVFS actuations.

8.2 Extensions and Future Work

Extensions of the current research and topics for future work can now be discussed. These efforts could focus on the enhancement of our developed tools and schemes, on the update of certain algorithms or even on the use of different policies and technologies for simulation and experimental purposes. Specifically:

- In Chapter 4, one could study the application of the MPFP tool on more complex circuitry. In this work, we have analyzed the 6T SRAM cell while basic logic gates (inverters and NAND gates) have also been discussed in prior art; sequential circuit blocks however, that consist of latches and flip-flops have not been studied. Given that such designs are typically non-symmetric and also create feedback loops, it is the author's belief that such an analysis would require substantial effort.

- Another general extension of Chapter 4 is to develop reliability modeling frameworks that capture the majority of failure mechanisms (as mentioned in Chapter 2). In our attempt, we focus on the dominant mechanisms of deeply scaled nodes namely, the RDF concerning time-zero variability and BTI (that is the phenomenon causing time-dependent variation).

- In Chapter 5, we have presented a case study of our PID-based DVFS controller that manages system dependability. We also briefly discussed an implementation of this scheme for temperature management. An interesting topic for future work is the development of a relative scheme that focuses on the performance and temperature in parallel. These two

features are usually overlapping: thermal management requires a low voltage and frequency operation while dependability needs the boosting of the performance when necessary. Hence, a policy that manages both features and the development of such a controller is important. In addition, while Section 5.4 compares our PID scheme to a policy based on the "conservative" governor, other existing policies could also be compared. This comparison would produce interesting results regarding energy and performance response.

- Chapter 6 deals with PID implementation using the system and adaptive scenarios. In the former implementation, we used the GSM spectrum sensing application and performed an offline analysis that clustered its nearly 800 RTSs into 30 scenarios. An exploration of different clustering profiles (such as the use of 5,10 or 20 scenarios) and the study of the response of our scheme in these cases, is definitely interesting. Such a work could provide significant information regarding the trade-off between the complexity of the scenario analysis and the response of the PID controller.

- Finally, in Chapter 7, we could focus on probabilistic workload predictors and predictors based on machine learning (as mentioned in Section 7.3) to develop elaborate dynamic scenario concepts where the future look-ahead aspects are even stronger emphasized and can be applied to more complex application contexts.

Appendix A

Appendix

A.1 Reliability Metrics

To characterize reliability and availability of digital systems several metrics have been adopted by the community. *Time to Failure* (TTF) refers to the length of time passed before a system experiences a fault or an error. Typically, however, it is mostly used for failures. If, for instance, a system experiences a failure after 5 months of operation, then the system's TTF at that instance is 5 months. *Mean Time to Failure* (MTTF) represents the average time a hardware component or a digital system is expected to operate before an error occurs. Therefore a system where errors occur after 1 year of operation has an MTTF equal to 1 year. Similar to the MTTF metric, *Mean Time Between Failures* (MTBF) is also widely used amongst reliability engineers to show the mean elapsed time between two consecutive errors. Mean Time Between Failures includes, in fact, MTTF and *Mean Time to Repair (MTTR)*, which refers to the average time required for an error to be repaired from the moment it has been detected. Hence, while MTBF is mostly used for repairable products, MTTF describes the reliability of systems without this capability. A depicture of the above, drawn for relevant work [185], is shown in Figure A.1. *Failure Rate* is defined as the number of failures on a system during a specific time interval. The most common unit for the failure rate is *Failures in Time* (FIT), which shows the number of failures per 1 billion operation hours. A component having a failure rate of 1 FIT is equivalent to having a MTBF of 1 billion hours.

Other metrics focus on the quality of the semiconductor fabrication process. The ratio of the dice that are working correctly, to the total number of dice

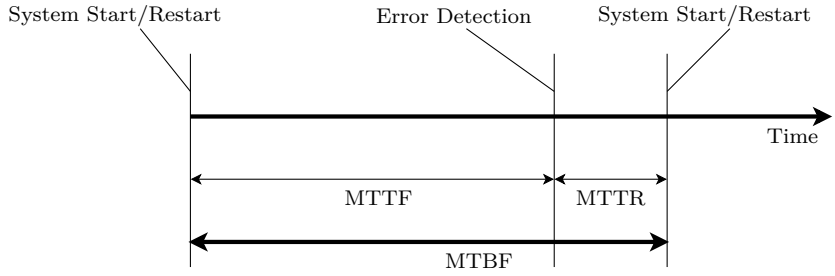


Figure A.1: Depicture of the correlation between MTTF,MTTR and MTBF

on the wafer is defined as *(die) yield*. Because of faults in the manufacturing process (like imperfections on the starting or the photomasking material) yield is typically far below 100% [292]. Evidently, yield is estimated before the products are shipped, through the manufacturing test procedure that screens out the defective parts. Another important metric that shows the density of chip-killing defects is *Defects per Million* (DPM) and it is actually defined as the number of defective parts per million units, chips in this case.

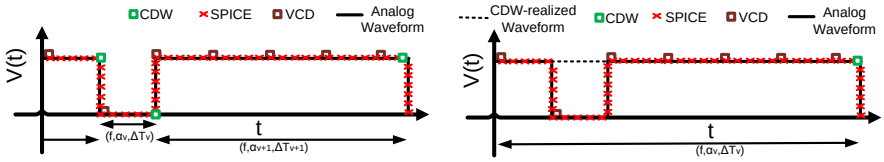
A.2 CDW Approximation

The computational overhead for a dynamic timing simulation of a circuit design is directly coupled to the granularity of the workload's signal representation. Typical signal representations can be:

1. SPICE: It is considered the standard tool for IC simulations while a broad array of SPICE simulators are available on the market. The signal is discretized over adjustable time intervals. SPICE signal representations are considered the most accurate amongst circuit simulators.

2. Value Change Dump (VCD): VCD is an ASCII-based signal format defined along with the Verilog hardware description language. The waveform's representation is achieved via a fitting of voltage to logic levels. The two logic levels correspond to high (V_{dd}) and low (V_{ss}) voltage.

Through our tool, signal activity is transformed to a Compact Digital Waveform (CDW). A detailed evaluation of the *CDW* signal representation is presented in prior art [233]. Briefly, "consecutive signal regions that feature similar f and α figures and occupy a duration Δt are represented by a single CDW point with coordinates (f,α,Δt)". In our case study, initial workload of the buffers is



(a) (b)

Figure A.2: Two epsilon scenarios for CDW: (a) Artist's impression of a waveform CDW compression to the respective VCD and SPICE points for an $\varepsilon < 3 \cdot T_{clock}$. Consecutive signal points with similar voltage levels are grouped in a single CDW point; (b) For an $\varepsilon \geq 3 \cdot T_{clock}$, the relaxation period lasting $3 \cdot T_{clock}$ is masked during the compression. CDW representation helps us alleviate extensive computational overhead of the simulations.

stored in VCD-like format and by utilizing `CDW approximation`, signal activity in compressed to CDW points. In detail, compression is achieved via a user-defined error margin ε (in clock cycles) for the time-duration of each consecutive point, as specified by the following formulation:

$$|\Delta T_{VCD} - \Delta T_{CDW}| < \varepsilon \cdot T_{clock} \tag{A.1}$$

where ΔT_{VCD} represents the initial, cycle-accurate signal activity of a device in VCD format and ΔT_{CDW} its CDW counterpart. $Tclock$ is simply one period of the processor's clock.When $\varepsilon \to 0$, $\Delta T_{CDW} \to \Delta T_{VCD}$ and the CDW representation is similar to the initial signal VCD activity. On the contrary, the crudest approximation is achieved when $\varepsilon \to \infty$ and all signal activity is compressed to a single CDW point. Figure A.2 presents a simplified compression example of the same waveform for two different ε scenarios.

A.3 Choosing the Operating Points for the PID Controller

The NXP IMX6Q CPU consists of an ARM Cortex-A9 quad core while the Linux DVFS driver normally provides the usage of only 3 different operating points, in the range of 396 MHz to 996 MHz [199]. However, a custom driver has been developed to enable the use of numerous DVFS points (up to 50). With this configuration, we can achieve a finer granularity division of the frequency and voltage space. To choose the actual number of DVFS points for our configuration,

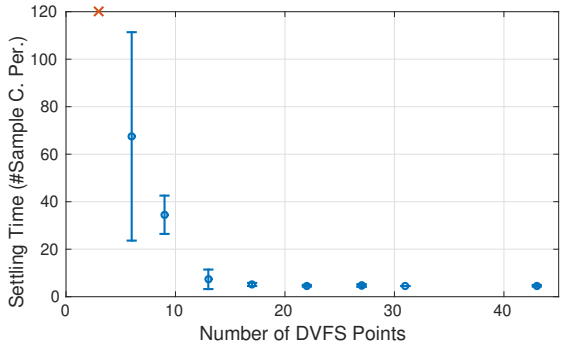


Figure A.3: Graph showing the dependence of settling time on number of DVFS points. Settling time per each configuration is the mean value as estimated after 10 iterations.

we performed experiments with the controller, instantiating different number of operating points (that were equally placed between maximum and minimum frequency limits). Each time we injected an error, a rollback event occurred and the *settling time* for all different configurations was measured. From the domain of control theory, we define *settling time* as:

Definition 15. *Settling Time* in our case, represents the time elapsed from the occurrence of a rollback event to the time at which the slack converges to zero, remaining within an error band of 10%.

In Figure A.3 we present our experimental results. We see the mean estimated values for settling time after 10 experiments per configuration while the standard error for a confidence interval of 95% is also highlighted. It can be realized that, when utilizing only 3 DVFS levels, we are unable to converge slack to zero. Therefore, settling time cannot be defined in this case. Interestingly, increasing the DVFS points greatly decreases settling time nevertheless, after a certain degree, further increment of these points hardly has any impact on the *settling time*. We must note that in practice, it is unusual for embedded processors to offer a large number of operating points.

A.4 Slack Ringing/Jittering

In real conditions, the PID controller's response usually deviates from the ideal operation and is subject to certain non-ideal side effects. An example of

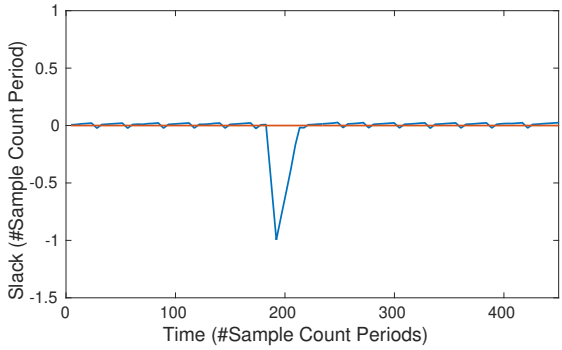


Figure A.4: Ringing of the Slack.

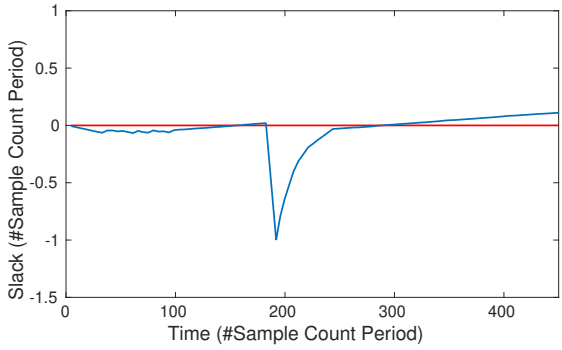


Figure A.5: No Ringing of the Slack.

such a "nuisance" is called *ringing/jittering* and hints to the oscillation of the controller's output between the set point value. This oscillation can be seen in numerous cases throughout our work, like for instance in Figures 5.2, 5.5 and 5.17. In the case of performance variability and slack mitigation, this oscillation is caused by random variations in the execution time of a certain task. For example, while nominal duration of a specific task at a frequency of 792 MHz may last t_{ref} seconds, while the system is running at the same speed and at error-free conditions, actual time delay of the task might may slightly diverge from its reference value.

Thus ringing of the output can be partly tackled with appropriate tuning of the controller's gains. Tuning the controller gains in a manner that allows the scheme to be less aggressive and sensitive to minor slack deviations would restrain the ringing. In this case however, the settling time would also be

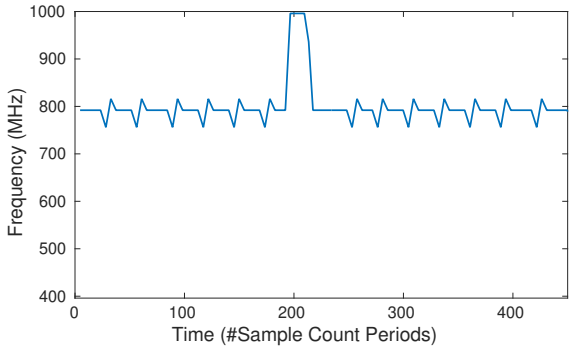


Figure A.6: Frequency Decisions: Ringing Scenario.

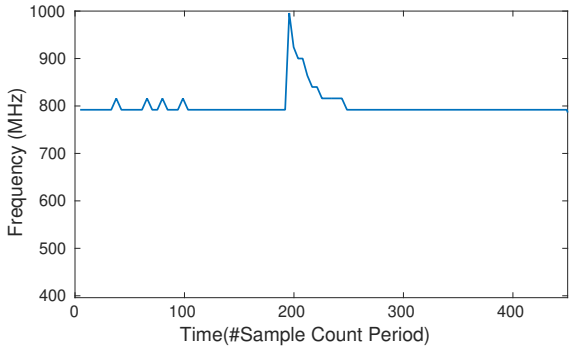


Figure A.7: Frequency Decisions: Non-Ringing Scenario.

extended. This can be visualized in Figures A.4 and A.5. In the first Figure, after a rollback event, frequent jittering of the output is observed. In the second case, the ringing is negligible. Nevertheless, settling time is considerably increased now. Frequency actuations of the two examples, that portray different PID gains, are shown in Figures A.6 and A.7.

A.5 Tuning Controller's Gains – Overshoot Criterion

A proper tuning the gains of our controller is highly important for an effective PID controller scheme. Actually, aside from the *ringing* of the output and the

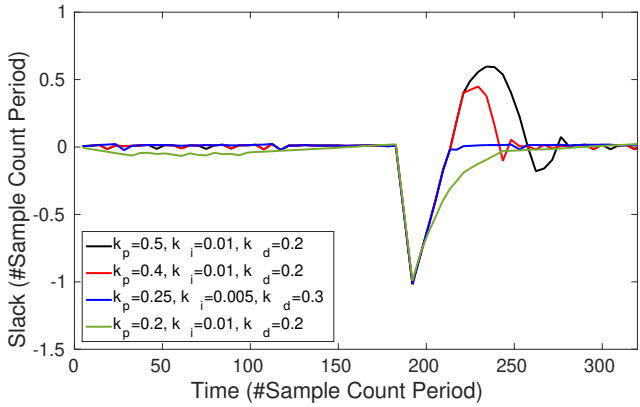


Figure A.8: Transient Response of the Controller for Different Gain Configurations.

settling time, other metrics are also used to characterize the scheme's response.

Definition 16. *Overshoot* describes the amount the output exceeds the final, steady-state value during the transient response.

Typically, PID gains are tuned in a way that *overshoots* are limited while, at the same time, *settling time* is minimized. In our case, for instance, *overshoot* means an unnecessary energy penalty. These two goals however, can conflict. Tuning the controller to minimize *overshoot* usually translates into a sacrifice of its aggressiveness; the controller now needs to progressively slow the control effort as it approaches the setpoint and this implies *settling time* is stretched. Ideally, the controller would be aggressive enough to quickly set the output to the desired value while avoiding an *overshoot*. Therefore tuning of the PID gains needs to carefully take place. When setting the gains of our PID scheme, we mainly targeted on minimizing *settling time* and *jittering* while also avoiding *overshoots*[1]. Figure A.8 shows the controller's transient responses (slack) after a rollback event for several gain setups.

[1]In reality, an overshoot can occur due to aggressive k_i values and can be limited by proper k_d tuning. Nevertheless, this study goes deep in control theory and is beyond the scope of the current work.

Bibliography

[1] SPARC Enterprise M3000, M4000, M5000, M8000, and M9000 Server Architecture Flexible, Mainframe-Class Compute Power. White paper, Oracle, April 2011.

[2] Abdelzaher, T., Stankovic, J., Lu, C., Zhang, R., and Lu, Y. Feedback performance control in software services. *Control Systems, IEEE, 23*, 4 (May 2003), 74–90.

[3] Abdelzaher, T., Stankovic, J., Lu, C., Zhang, R., and Lu, Y. Feedback performance control in software services. *Control Systems, IEEE, 23*, 4 (May 2003), 74–90.

[4] Abella, J., Vera, X., and Gonzalez, A. Penelope: The nbti-aware processor. In *40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007)* (Dec 2007), pp. 85–96.

[5] Achanta, R. S., Plawsky, J. L., and Gill, W. N. Copper ion transport induced dielectric failure: Inclusion of elastic drift and consequences for reliability. *Journal of Vacuum Science & Technology A 26*, 6 (2008), 1497–1500.

[6] Agarwal, A., Blaauw, D., Zolotov, V., and Vrudhula, S. Computation and refinement of statistical bounds on circuit delay. In *Proceedings 2003. Design Automation Conference (IEEE Cat. No.03CH37451)* (June 2003), pp. 348–353.

[7] Agarwal, A., Kang, K., Bhunia, S., Gallagher, J. D., and Roy, K. Device-aware yield-centric dual-vt design under parameter variations in nanoscale technologies. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems 15*, 6 (June 2007), 660–671.

[8] Ajami, A. H., Banerjee, K., and Pedram, M. Modeling and analysis of nonuniform substrate temperature effects on global ulsi interconnects.

IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 24, 6 (June 2005), 849–861.

[9] Alam, M., and Mahapatra, S. A comprehensive model of pmos nbti degradation. *Microelectronics Reliability 45*, 1 (2005), 71 – 81.

[10] Alewine, N. J., Shyh-Kwei Chen, Kent Fuchs, W., and Hwu, W. . W. Compiler-assisted multiple instruction rollback recovery using a read buffer. *IEEE Transactions on Computers 44*, 9 (Sep. 1995), 1096–1107.

[11] Amato, P., Bellini, S., Ferrari, M., Laurent, C., Sforzin, M., and Tomasoni, A. Fast decoding ecc for future memories. *IEEE Journal on Selected Areas in Communications 34*, 9 (Sep. 2016), 2486–2497.

[12] AMD. Amd fx processors unleashed. `https://www.amd.com/Documents/AMD_FX_Performance_Tuning_Guide.pdf`, 2011. a Guide to Performance Tuning with AMD OverDrive and the new AMD FX Processors.

[13] AMD Embedded Solutions. Understanding Power Management and Processor Performance Determinism. Tech. rep., 2018.

[14] Amin, C. S., Menezes, N., Killpack, K., Dartu, F., Choudhury, U., Hakim, N., and Ismail, Y. I. Statistical static timing analysis: How simple can we get? In *Proceedings of the 42Nd Annual Design Automation Conference* (New York, NY, USA, 2005), DAC '05, ACM, pp. 652–657.

[15] ARM. Cortex-a9 processor specifications. `https://www.arm.com/products/processors/cortex-a/cortex-a9.php?tab=Specifications`, 2015.

[16] Arrobo, G. E., and Gitlin, R. D. Improving the reliability of wireless body area networks. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society* (Aug 2011), pp. 2192–2195.

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

[18] Asenov, A., Kaya, S., and Brown, A. R. Intrinsic parameter fluctuations in decananometer mosfets introduced by gate line edge roughness. *IEEE Transactions on Electron Devices 50*, 5 (May 2003), 1254–1260.

[19] Avelar, V. Comparing availability of various rack power redundancy configurations. `https://download.schneider-electric.com/files?p_Doc_Ref=SPD_SADE-5TNRKC_EN`, 2003. White Paper.

[20] Avizienis, A., Laprie, J., Randell, B., and Landwehr, C. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing 1*, 1 (Jan 2004), 11–33.

[21] Azeem, B. A., and Helal, M. Performance evaluation of checkpoint/restart techniques: For mpi applications on amazon cloud. In *2014 9th International Conference on Informatics and Systems* (Dec 2014), pp. PDC–49–PDC–57.

[22] Bae, H.-C., Bae, H.-E., Jeon, S.-J., Jung, K.-H., Eom, Y.-S., and Choi, K.-S. 3d sip module using tsv and novel low-volume solder-on-pad(sop) process. *2012 4th Electronic System-Integration Technology Conference* (2012), 1–4.

[23] Baek, W., and Chilimbi, T. M. Green: A framework for supporting energy-conscious programming using controlled approximation. *SIGPLAN Not. 45*, 6 (June 2010), 198–209.

[24] Balouchestani, M., Raahemifar, K., and Krishnan, S. Increasing the reliability of wireless sensor network with a new testing approach based on compressed sensing theory. In *2011 Eighth International Conference on Wireless and Optical Communications Networks* (May 2011), pp. 1–4.

[25] Bambagini, M., Marinoni, M., Aydin, H., and Buttazzo, G. Energy-aware scheduling for real-time systems: A survey. *ACM Trans. Embed. Comput. Syst. 15*, 1 (Jan. 2016), 7:1–7:34.

[26] Ban, Y., Sundareswaran, S., and Pan, D. Z. Electrical impact of line-edge roughness on sub-45-nm node standard cells. *Journal of Micro/Nanolithography, MEMS, and MOEMS 9*, 4 (2010), 1–10.

[27] Ban, Y., and Yang, J. Layout aware line-edge roughness modeling and poly optimization for leakage minimization. In *2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC)* (June 2011), pp. 447–452.

[28] Banijamali, B., Ramalingam, S., Liu, H., and Kim, M. Outstanding and innovative reliability study of 3d tsv interposer and fine pitch solder micro-bumps. In *2012 IEEE 62nd Electronic Components and Technology Conference* (May 2012), pp. 309–314.

[29] Bathen, L., Dutt, N., Nicolau, A., and Gupta, P. Vamv: Variability-aware memory virtualization. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2012* (March 2012), pp. 284–287.

[30] Baumann, R. C. Radiation-induced soft errors in advanced semiconductor technologies. *IEEE Transactions on Device and Materials Reliability 5*, 3 (Sep. 2005), 305–316.

[31] Beichl, I., and Sullivan, F. The importance of importance sampling. *Computing in Science Engineering 1*, 2 (March 1999), 71–73.

[32] Bellasi, P., Massari, G., and Fornaciari, W. Effective runtime resource management using linux control groups with the barbequertrm framework. *ACM Trans. Embed. Comput. Syst. 14*, 2 (Mar. 2015), 39:1–39:17.

[33] Bergamaschi, R., , Buyuktosunoglu, A., Patel, H., Nair, I., Dittmann, G., Janssen, G., Dhanwada, N., , Bose, P., and Darringer, J. Exploring power management in multi-core systems. In *2008 Asia and South Pacific Design Automation Conference* (March 2008), pp. 708–713.

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

[35] Bertossi, A. A., Mancini, L. V., and Rossini, F. Fault-tolerant rate-monotonic first-fit scheduling in hard-real-time systems. *IEEE Transactions on Parallel and Distributed Systems 10*, 9 (Sep. 1999), 934–945.

[36] Bhasker, J., and Chadha, R. *Static Timing Analysis for Nanometer Designs: A Practical Approach*, 1st ed. Springer Publishing Company, Incorporated, 2009.

[37] Bickford, J. P., and Foreman, E. A. Selective voltage binning within a three-dimensional integrated chip stack. `https://patents.google.com/patent/US20140229909`, 2013. US Patent, US20140229909A1.

[38] Bina, M., Tyaginov, S., Franco, J., Rupp, K., Wimmer, Y., Osintsev, D., Kaczer, B., and Grasser, T. Predictive hot-carrier modeling of n-channel mosfets. *IEEE Transactions on Electron Devices 61*, 9 (Sep. 2014), 3103–3110.

[39] Binkert, N., Beckmann, B., Black, G., Reinhardt, S. K., Saidi, A., Basu, A., Hestness, J., Hower, D. R., Krishna, T., Sardashti, S., and et al. The gem5 simulator. *SIGARCH Comput. Archit. News 39*, 2 (Aug. 2011), 1—–7.

[40] Blaauw, D., Chopra, K., Srivastava, A., and Scheffer, L. Statistical timing analysis: From basic principles to state of the art. *Trans. Comp.-Aided Des. Integ. Cir. Sys. 27*, 4 (Apr. 2008), 589–607.

[41] Black, B., Annavaram, M., Brekelbaum, N., DeVale, J., Jiang, L., Loh, G. H., McCaule, D., Morrow, P., Nelson, D. W., Pantuso, D., Reed, P., Rupley, J., Shankar, S., Shen, J., and Webb, C. Die stacking (3d) microarchitecture. In *2006 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06)* (Dec 2006), pp. 469–479.

[42] Black, J. R. Electromigration—a brief survey and some recent results. *IEEE Transactions on Electron Devices 16*, 4 (April 1969), 338–347.

[43] Bowen, N. S., and Pradham, D. K. Processor-and memory-based checkpoint and rollback recovery. *Computer 26*, 2 (Feb 1993), 22–31.

[44] Brayton, R. K., Hachtel, G. D., and Sangiovanni-Vincentelli, A. L. A survey of optimization techniques for integrated-circuit design. *Proceedings of the IEEE 69*, 10 (Oct 1981), 1334–1362.

[45] Brodowski, D., Wysocki, R. J., and Kumar, V. Linux cpufreq governors. `https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt`, 2017. CPU frequency and voltage scaling code in the Linux(TM) kernel.

[46] Busby, J., Hawken, D., Perfecto, E., Dang, B., Shah, J., Ruhmer, K., Gruber, P., Weisman, R., and Buchwalter, S. C4np lead free solder bumping and 3d micro bumping. In *2008 IEEE/SEMI Advanced Semiconductor Manufacturing Conference* (May 2008), pp. 333–339.

[47] Cadence. In *Spectre Circuit Simulator Datasheet*.

[48] Cadence. Virtuoso XL Layout Editor User Guide. Tech. rep., 2006.

[49] Camarota, R. C. Method and apparatus for self-annealing multi-die interconnect redundancy control. `https://patents.google.com/patent/EP2730028A1/no`, 2011. US Patent, US8539420B2.

[50] Cao, Y. K. What is predictive technology model (ptm)? *SIGDA Newsl. 39*, 3 (Mar. 2009).

[51] Catthoor, F., Basten, T., Zompakis, N., Geilen, M., and Kjeldsberg, P. *System-Scenario-based Design Principles and Applications*. Springer, 01 2020.

[52] Chan, C., Schwartz-Narbonne, D., Sethi, D., and Malik, S. Specification and synthesis of hardware checkpointing and rollback mechanisms. In *DAC Design Automation Conference 2012* (June 2012), pp. 1222–1228.

[53] Chandy, K. M., and Ramamoorthy, C. V. Rollback and recovery strategies for computer programs. *IEEE Transactions on Computers C-21*, 6 (June 1972), 546–556.

[54] Chang, L., Fried, D. M., Hergenrother, J., Sleight, J. W., Dennard, R. H., Montoye, R. K., Sekaric, L., McNab, S. J., Topol, A. W., Adams, C. D., Guarini, K. W., and Haensch, W. Stable sram cell design for the 32 nm node and beyond. In *Digest of Technical Papers. 2005 Symposium on VLSI Technology, 2005.* (June 2005), pp. 128–129.

[55] Charron, F. Air flow management system for an internet data center. `https://patents.google.com/patent/US6672955B2/en`, 2002. US Patent, US6672955B2.

[56] Chen, C. L. Symbol error correcting codes for memory applications. In *Proceedings of Annual Symposium on Fault Tolerant Computing* (June 1996), pp. 200–207.

[57] Chen, C. L., and Hsiao, M. Y. Error-correcting codes for semiconductor memory applications: A state-of-the-art review. *IBM Journal of Research and Development 28*, 2 (March 1984), 124–134.

[58] Chen, F., Bravo, O., Chanda, K., McLaughlin, P., Sullivan, T., Gill, J., Lloyd, J., Kontra, R., and Aitken, J. A comprehensive study of low-k sicoh tddb phenomena and its reliability lifetime model development. In *2006 IEEE International Reliability Physics Symposium Proceedings* (March 2006), pp. 46–53.

[59] Chen, J., and Kuo, C. Energy-efficient scheduling for real-time systems on dynamic voltage scaling (dvs) platforms. In *13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2007)* (Aug 2007), pp. 28–38.

[60] Chen, X., Wang, Y., Cao, Y., Xie, Y., and Yang, H. Assessment of circuit optimization techniques under nbti. *IEEE Design Test 30*, 6 (Dec 2013), 40–49.

[61] Chenming Hu, Simon C. Tam, Fu-Chieh Hsu, Ping-Keung Ko, Tung-Yi Chan, and Terrill, K. W. Hot-electron-induced mosfet degradation - model, monitor, and improvement. *IEEE Journal of Solid-State Circuits 20*, 1 (Feb 1985), 295–305.

[62] Choi, M., and Milor, L. Diagnosis of optical lithography faults with product test sets. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 27*, 9 (Sep. 2008), 1657–1669.

[63] Communication, T., and Security. Spectrum monitoring and homeland security. `https://www.thalesgroup.com/en/worldwide/security`, 2015.

[64] Cong, J., Gururaj, K., Jiang, W., Liu, B., Minkovich, K., Yuan, B., and Zou, Y. Accelerating monte carlo based ssta using fpga. In *Proceedings of the 18th Annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays* (New York, NY, USA, 2010), FPGA '10, Association for Computing Machinery, pp. 111—-114.

[65] Corbetta, S., Meeus, W., Rodopoulos, D., Cappe, E., Catthoor, F., and Fritsch, A. System-wide reliability analysis on real processor and application under vdd and t stress. In *SELSE– Silicon Errors in Logic –System Effects* (2016).

[66] Cosemans, S., Dehaene, W., and Catthoor, F. A 3.6 pj/access 480 mhz, 128 kb on-chip sram with 850 mhz boost mode in 90 nm cmos with tunable sense amplifiers. *IEEE JSSC 44*, 7 (July 2009), 2065–2077.

[67] Croon, J. A., Storms, G., Winkelmeier, S., Pollentier, I., Ercken, M., Decoutere, S., Sansen, W., and Maes, H. E. Line edge roughness: characterization, modeling and impact on device behavior. In *Digest. International Electron Devices Meeting,* (Dec 2002), pp. 307–310.

[68] Das, A., Kumar, A., Veeravalli, B., Shafik, R., Merrett, G., and Al-Hashimi, B. Workload uncertainty characterization and adaptive frequency scaling for energy minimization of embedded systems. In *2015 Design, Automation Test in Europe Conference Exhibition (DATE)* (March 2015), pp. 43–48.

[69] Data Center Group – Intel Corporation. Managing Process Variation in Intel's 45nm CMOS Technology. Tech. rep., 2008.

[70] Data Center Group – Intel Corporation. Intel® Xeon® Processor E7 Family: Reliability, Availability, and Serviceability. Tech. rep., 2011.

[71] Davis, R. I., and Burns, A. A survey of hard real-time scheduling for multiprocessor systems. *ACM Comput. Surv. 43*, 4 (Oct. 2011), 35:1–35:44.

[72] Dawoud, W., Takouna, I., and Meinel, C. Elastic vm for cloud resources provisioning optimization. In *Advances in Computing and*

Communications (Berlin, Heidelberg, 2011), A. Abraham, J. Lloret Mauri, J. F. Buford, J. Suzuki, and S. M. Thampi, Eds., Springer Berlin Heidelberg, pp. 431–445.

[73] Deal, B. E., Sklar, M., Grove, A. S., and Snow, E. H. Characteristics of the surface-state charge (qss) of thermally oxidized silicon. *J. Electrochem. Soc 114*, 3 (Mar. 1967), 266–274.

[74] Dell, and Intel. Advanced reliability for intel®xeon®processorson dell™ poweredge™ servers. `https://www.dell.com/downloads/global/products/pedge/en/poweredge-advanced-reliability-intel-nehalem-processor.pdf`, 2010. A Dell Technical White Paper.

[75] Developer, A. Cortex-a9 technical reference manual. `http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0388e/BEHEDIHI.html`, 2009. About the Performance Monitoring Unit.

[76] Dharchoudhury, A., and Kang, S. M. Worst-case analysis and optimization of vlsi circuit performances. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 14*, 4 (April 1995), 481–492.

[77] Di Wu, Venkataraman, G., Jiang Hu, Quiyang Li, and Mahapatra, R. Dicer: distributed and cost-effective redundancy for variation tolerance. In *ICCAD-2005. IEEE/ACM International Conference on Computer-Aided Design, 2005.* (Nov 2005), pp. 393–397.

[78] Döbel, B., Härtig, H., and Engel, M. Operating system support for redundant multithreading. In *Proceedings of the Tenth ACM International Conference on Embedded Software* (New York, NY, USA, 2012), EMSOFT '12, ACM, pp. 83–92.

[79] Donald, J., and Martonosi, M. Techniques for multicore thermal management: Classification and new exploration. In *33rd International Symposium on Computer Architecture (ISCA'06)* (June 2006), pp. 78–88.

[80] Duell, J., Hargrove, P., and Roman, E. The design and implementation of berkeley lab"s linux checkpoint/restart.

[81] El-Sayed, N., and Schroeder, B. To checkpoint or not to checkpoint: Understanding energy-performance-i/o tradeoffs in hpc checkpointing. In *2014 IEEE International Conference on Cluster Computing (CLUSTER)* (Sep. 2014), pp. 93–102.

[82] Ernst, D., Das, S., Lee, S., Blaauw, D., Austin, T., Mudge, T., Nam Sung Kim, and Flautner, K. Razor: circuit-level correction of timing errors for low-power operation. *IEEE Micro 24*, 6 (Nov 2004), 10–20.

[83] Fang, J., Gupta, S., Kumar, S. V., Marella, S. K., Mishra, V., Zhou, P., and Sapatnekar, S. S. Circuit reliability: From physics to architectures: Embedded tutorial paper. In *2012 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (Nov 2012), pp. 243–246.

[84] Feldbaumer, D., Babcock, J., Mercier, V., and Chun, C. Pulse current trimming of polysilicon resistors. *IEEE TED 42*, 4 (Apr 1995), 689–696.

[85] Feldbaumer, D. W., Babcock, J. A., Mercier, V. M., and Chun, C. K. Y. Pulse current trimming of polysilicon resistors. *IEEE Transactions on Electron Devices 42*, 4 (April 1995), 689–696.

[86] Ferri, C., Sherief Reda, and R. Iris Bahar. Strategies for improving the parametric yield and profits of 3d ics. In *2007 IEEE/ACM International Conference on Computer-Aided Design* (Nov 2007), pp. 220–226.

[87] Fiala, D., Mueller, F., Engelmann, C., Riesen, R., Ferreira, K., and Brightwell, R. Detection and correction of silent data corruption for large-scale high-performance computing. In *SC '12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis* (Nov 2012), pp. 1–12.

[88] Fiedler, M. J. A minimal mpeg-1/2 audio layer ii decoder library. `https://github.com/technosaurus/PDMP2/blob/master/kjmp2.c`, 2013. Github.

[89] Forzan, C., and Pandini, D. Why we need statistical static timing analysis. In *2007 25th International Conference on Computer Design* (Oct 2007), pp. 91–96.

[90] Fraternali, F., Bartolini, A., Cavazzoni, C., and Benini, L. Quantifying the impact of variability and heterogeneity on the energy efficiency for a next-generation ultra-green supercomputer. *IEEE Transactions on Parallel and Distributed Systems 29*, 7 (July 2018), 1575–1588.

[91] Gao, S., He, B., and Xu, J. Real-time in-memory checkpointing for future hybrid memory systems. In *Proceedings of the 29th ACM on*

International Conference on Supercomputing (New York, NY, USA, 2015), ICS '15, ACM, pp. 263–272.

[92] Gaonkar, S., Rozier, E., Tong, A., and Sanders, W. H. Scaling file systems to support petascale clusters: A dependability analysis to support informed design choices. In *2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN)* (June 2008), pp. 386–391.

[93] Garg, R., and Khatri, S. *Analysis and design of resilient VLSI circuits: Mitigating soft errors and process variations*. 01 2010.

[94] Gerard, J. Repair machine for integrated circuits using laser-induced microchemistry. In *IEEE/CHMT IEMTS* (Oct 1988).

[95] Gheorghita, S. V., Basten, T., and Corporaal, H. Application scenarios in streaming-oriented embedded-system design. *IEEE Design Test of Computers 25*, 6 (Nov 2008), 581–589.

[96] Gheorghita, S. V., Palkovic, M., Hamers, J., Vandecappelle, A., Mamagkakis, S., Basten, T., Eeckhout, L., Corporaal, H., Catthoor, F., Vandeputte, F., and Bosschere, K. D. System-scenario-based design of dynamic embedded systems. *ACM Trans. Des. Autom. Electron. Syst. 14*, 1 (Jan. 2009), 3:1–3:45.

[97] Ghosh, S., Melhem, R., and Mosse, D. Fault-tolerant scheduling on a hard real-time multiprocessor system. In *Proceedings of 8th International Parallel Processing Symposium* (April 1994), pp. 775–782.

[98] Giannopoulou, G., Stoimenov, N., Huang, P., and Thiele, L. Scheduling of mixed-criticality applications on resource-sharing multicore systems. In *Proceedings of the Eleventh ACM International Conference on Embedded Software* (October 2013).

[99] Gleixner, T., McKenney, P. E., and Guittot, V. Cleaning up linux's cpu hotplug for real time and energy management. *SIGBED Rev. 9*, 4 (Nov. 2012), 49–52.

[100] Gnad, D., Shafique, M., Kriebel, F., Rehman, S., Duo Sun, and Henkel, J. Hayat: Harnessing dark silicon and variability for aging deceleration and balancing. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)* (June 2015), pp. 1–6.

[101] Goel, N., Naphade, T., and Mahapatra, S. Combined trap generation and transient trap occupancy model for time evolution of nbti during dc multi-cycle and ac stress. In *2015 IEEE International Reliability Physics Symposium* (April 2015), pp. 4A.3.1–4A.3.7.

[102] Gonzalez, R., and Horowitz, M. Energy dissipation in general purpose microprocessors. *IEEE Journal of Solid-State Circuits 31*, 9 (Sep. 1996), 1277–1284.

[103] Goudarzi, M., and Ishihara, T. Sram leakage reduction by row/column redundancy under random within-die delay variation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems 18*, 12 (Dec 2010), 1660–1671.

[104] Grasser, T., Kaczer, B., Goes, W., Reisinger, H., Aichinger, T., Hehenberger, P., Wagner, P., Schanovsky, F., Franco, J., Luque, M. T., and Nelhiebel, M. The paradigm shift in understanding the bias temperature instability: From reaction–diffusion to switching oxide traps. *IEEE Transactions on Electron Devices 58*, 11 (Nov 2011), 3652–3666.

[105] Groeseneken, G., Degraeve, R., Kaczer, B., and Roussel, P. Recent trends in reliability assessment of advanced cmos technologies. In *Proceedings of the 2005 International Conference on Microelectronic Test Structures, 2005. ICMTS 2005.* (April 2005), pp. 81–88.

[106] Gu, C., Guan, N., Deng, Q., and Yi, W. Partitioned mixed-criticality scheduling on multiprocessor platforms. In *Design, Automation and Test in Europe Conference and Exhibition (DATE)* (March 2014).

[107] Hardy, D., Puaut, I., and Sazeides, Y. Probabilistic wcet estimation in presence of hardware for mitigating the impact of permanent faults. In *2016 Design, Automation Test in Europe Conference Exhibition (DATE)* (March 2016), pp. 91–96.

[108] Hasofer, A. M., and Lind, N. C. Exact and invariant second moment code format. *Journal of the Engineering Mechanics Division 100* (1974), 111–121.

[109] Hazelwood, K., and Brooks, D. Eliminating voltage emergencies via microarchitectural voltage control feedback and dynamic optimization. In *Proceedings of the 2004 International Symposium on Low Power Electronics and Design (IEEE Cat. No.04TH8758)* (Aug 2004), pp. 326–331.

[110] Hazucha, P., Svensson, C., and Wender, S. A. Cosmic-ray soft error rate characterization of a standard 0.6-/spl mu/m cmos process. *IEEE Journal of Solid-State Circuits 35*, 10 (Oct 2000), 1422–1429.

[111] Herbert, S., and Marculescu, D. Variation-aware dynamic voltage/frequency scaling. In *IEEE HPCA* (Feb 2009), pp. 301–312.

[112] Heremans, P., Bellens, R., Groeseneken, G., and Maes, H. E. Consistent model for the hot-carrier degradation in n-channel and p-channel mosfets. *IEEE Transactions on Electron Devices 35*, 12 (Dec 1988), 2194–2209.

[113] Hongliang Chang, and Sapatnekar, S. S. Statistical timing analysis considering spatial correlations using a single pert-like traversal. In *ICCAD-2003. International Conference on Computer Aided Design (IEEE Cat. No.03CH37486)* (Nov 2003), pp. 621–625.

[114] Hsieh, A., and Hwang, T. Tsv redundancy: Architecture and design issues in 3-d ic. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems 20*, 4 (April 2012), 711–722.

[115] Hu, C. ., Rodbell, K. P., Sullivan, T. D., Lee, K. Y., and Bouldin, D. P. Electromigration and stress-induced voiding in fine al and al-alloy thin-film lines. *IBM Journal of Research and Development 39*, 4 (July 1995), 465–497.

[116] Hyman, R., Bhattacharya, K., and Ranganathan, N. Redundancy mining for soft error detection in multicore processors. *IEEE Transactions on Computers 60*, 8 (Aug 2011), 1114–1125.

[117] Intel. Intel turbo boost technology 2.0. `https://www.intel.com/content/www/us/en/architecture-and-technology/turbo-boost/turbo-boost-technology.html`, 2008. Higher Performance When You Need It Most.

[118] Ishigami, T., Kurokawa, T., Kakuhara, Y., Withers, B., Jacobs, J., Kolics, A., Ivanov, I., Sekine, M., and Ueno, K. High reliability cu interconnection utilizing a low contamination cowp capping layer. In *IEEE IITC* (June 2004).

[119] Islam, A., Gupta, G., Ahmed, K., Mahapatra, S., and Alam, M. Optimization of gate leakage and nbti for plasma-nitrided gate oxides by numerical and analytical models. *IEEE TED 55*, 5 (May 2008), 1143–1152.

[120] Islam, A. E., Kufluoglu, H., Varghese, D., Mahapatra, S., and Alam, M. A. Recent issues in negative-bias temperature instability: Initial degradation, field dependence of interface trap generation, hole trapping effects, and relaxation. *IEEE Transactions on Electron Devices 54*, 9 (Sep. 2007), 2143–2154.

[121] James, N., Restle, P., Friedrich, J., Huott, B., and McCredie, B. Comparison of split-versus connected-core supplies in the power6 microprocessor. In *IEEE International Solid-State Circuits Conference* (February 2007).

[122] Jan, H., Paul, A., Minhas, A., Ahmad, A., Jabbar, S., and Kim, M. Dependability and reliability analysis of intra cluster routing technique. *Peer-to-Peer Networking and Applications 8* (09 2015).

[123] Januszewski, R. Mos device aging analysis with hspice and customsim. `https://www.synopsys.com/content/dam/synopsys/verification/white-papers/mosra-wp.pdf`, 2011. White Paper.

[124] Jeffery, C. M., and Figueiredo, R. J. O. A flexible approach to improving system reliability with virtual lockstep. *IEEE Transactions on Dependable and Secure Computing 9*, 1 (Jan 2012), 2–15.

[125] Jess, J. A. G., Kalafala, K., Naidu, S. R., Otten, R. H. J. M., and Visweswariah, C. Statistical timing for parametric yield prediction of digital integrated circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 25*, 11 (Nov 2006), 2376–2392.

[126] Jiang, A., Li, Y., Gad, E. E., Langberg, M., and Bruck, J. Error correcting code for flash memories. In *2013 Information Theory and Applications Workshop (ITA)* (Feb 2013), pp. 1–4.

[127] Joshi, V., Agarwal, K., Blaauw, D., and Sylvester, D. Analysis and optimization of sram robustness for double patterning lithography. In *IEEE/ACM ICCAD* (Nov 2010), pp. 25–31.

[128] Juan, D., Garg, S., and Marculescu, D. Impact of manufacturing process variations on performance and thermal characteristics of 3d ics: Emerging challenges and new solutions. In *2013 IEEE International Symposium on Circuits and Systems (ISCAS)* (May 2013), pp. 541–544.

[129] Kaczer, B., Grasser, T., Roussel, J., Martin-Martinez, J., O'Connor, R., O'Sullivan, B. J., and Groeseneken, G. Ubiquitous relaxation in bti stressing—new evaluation and insights. In *2008 IEEE International Reliability Physics Symposium* (April 2008), pp. 20–27.

[130] Kang, K., Kufluoglu, H., Alain, M., and Roy, K. Efficient transistor-level sizing technique under temporal performance degradation due to nbti. In *ICCD* (Oct 2006).

[131] Kashyap, S., Dhillon, J. S., and Purini, S. Rlc - a reliable approach to fast and efficient live migration of virtual machines in the clouds. In *Proceedings of the 2014 IEEE International Conference on Cloud Computing* (Washington, DC, USA, 2014), CLOUD '14, IEEE Computer Society, pp. 360–367.

[132] Khalil, D., Khellah, M., Kim, N., Ismail, Y., Karnik, T., and De, V. K. Accurate estimation of sram dynamic stability. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems 16*, 12 (Dec 2008), 1639–1647.

[133] Khalil, D., Khellah, M., Kim, N.-S., Ismail, Y., Karnik, T., and De, V. Sram dynamic stability estimation using mpfp and its applications. *Microelectronics Journal 40*, 11 (2009), 1523–1530. International Conference on Microelectronics Digital and Mixed-Signal Circuits and Systems.

[134] Khan, S., and Hamdioui, S. Modeling and mitigating nbti in nanoscale circuits. In *2011 IEEE 17th International On-Line Testing Symposium* (July 2011), pp. 1–6.

[135] Khellah, M., Ye, Y., Kim, N., Somasekhar, D., Pandya, G., Farhang, A., Zhang, K., Webb, C., and De, V. Wordline bitline pulsing schemes for improving sram cell stability in low-vcc 65nm cmos designs. In *2006 Symposium on VLSI Circuits, 2006. Digest of Technical Papers.* (June 2006), pp. 9–10.

[136] Kim, J., Wang, F., and Nowak, M. Method and apparatus for providing through silicon via (tsv) redundancy. `https://patents.google.com/patent/US20100295600`, 2009. US Patent, US20100295600A1.

[137] Kim, Y. G., Kong, J., and Chung, S. W. A survey on recent os-level energy management techniques for mobile processing units. *IEEE Transactions on Parallel and Distributed Systems 29*, 10 (Oct 2018), 2388–2401.

[138] Koh, M., Mizubayashi, W., Iwamoto, K., Murakami, H., Ono, T., Tsuno, M., Mihara, T., Shibahara, K., Miyazaki, S., and Hirose, M. Limit of gate oxide thickness scaling in mosfets due to apparent threshold voltage fluctuation induced by tunnel leakage current. *IEEE Transactions on Electron Devices 48*, 2 (Feb 2001), 259–264.

[139] Kong, J., Chung, S. W., and Skadron, K. Recent thermal management techniques for microprocessors. *ACM Comput. Surv. 44*, 3 (June 2012), 13:1–13:42.

[140] Konstanteli, K., Cucinotta, T., Psychas, K., and Varvarigou, T. Admission control for elastic cloud services. In *Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing* (Washington, DC, USA, 2012), CLOUD '12, IEEE Computer Society, pp. 41–48.

[141] Kriebel, F., Shafique, M., Rehman, S., Henkel, J., and Garg, S. Variability and reliability awareness in the age of dark silicon. *IEEE Design Test 33*, 2 (April 2016), 59–67.

[142] Kritikakou, A., Catthoor, F., Kelefouras, V., and Goutis, C. A systematic approach to classify design-time global scheduling techniques. *ACM Comput. Surv. 45*, 2 (Mar. 2013), 14:1–14:30.

[143] Kroese, D. P., Brereton, T., Taimre, T., and Botev, Z. I. Why the monte carlo method is so important today. *WIREs Comput. Stat. 6*, 6 (Nov. 2014), 386—392.

[144] Kueing-Long Chen, Saller, S. A., Groves, I. A., and Scott, D. B. Reliability effects on mos transistors due to hot-carrier injection. *IEEE Transactions on Electron Devices 32*, 2 (Feb 1985), 386–393.

[145] Kuhn, K. J., Giles, M. D., Becher, D., Kolar, P., Kornfeld, A., Kotlyar, R., Ma, S. T., Maheshwari, A., and Mudanai, S. Process technology variation. *IEEE Transactions on Electron Devices 58*, 8 (Aug 2011), 2197–2208.

[146] Kukner, H., Weckx, P., Franco, J., Toledano-Luque, M., Cho, M., Kaczer, B., Raghavan, P., Doyoung Jang, Miyaguchi, K., Bardon, M. G., Catthoor, F., Van der Perre, L., Lauwereins, R., and Groeseneken, G. Scaling of bti reliability in presence of time-zero variability. In *2014 IEEE International Reliability Physics Symposium* (June 2014), pp. CA.5.1–CA.5.7.

[147] Kumar, S. K., Warn, S. C., and Lee, R. Led thermal management of an automotive electronic control module with display. In *2012 IEEE 14th Electronics Packaging Technology Conference (EPTC)* (Dec 2012), pp. 764–769.

[148] Kunhyuk Kang, Saakshi Gangwal, Park, S. P., and Kaushik, R. Nbti induced performance degradation in logic and memory circuits: how effectively can we approach a reliability solution? In *2008 Asia and South Pacific Design Automation Conference* (March 2008), pp. 726–731.

[149] Lampka, K., and Forsberg, B. Keep it slow and in time: Online dvfs with hard real-time workloads. In *Proc. DATE* (March 2016).

[150] Lampka, K., and Forsberg, B. Keep it slow and in time: Online dvfs with hard real-time workloads. In *2016 Design, Automation Test in Europe Conference Exhibition (DATE)* (March 2016), pp. 385–390.

[151] Lau, J. The most cost-effective integrator (tsv interposer) for 3d ic integration system-in-package (sip). In *International Electronic Packaging Technical Conference and Exhibition* (01 2011).

[152] Leung, L.-F., Tsui, C.-Y., and Hu, X. S. Exploiting dynamic workload variation in low energy preemptive task scheduling. In *Design, Automation and Test in Europe* (March 2005), pp. 634–639.

[153] Li, X., and Qiu, Z. A most probable point-based univariate method for reliability evaluation of composite laminates with random and interval parameters. *IEEE Transactions on Reliability* (2019), 1–14.

[154] Lichtensteiger, S., and Bickford, J. P. Using selective voltage binning to maximize yield. *IEEE Transactions on Semiconductor Manufacturing 26*, 4 (Nov 2013), 436–441.

[155] Liew, B. ., Cheung, N. W., and Hu, C. Projecting interconnect electromigration lifetime for arbitrary current waveforms. *IEEE Transactions on Electron Devices 37*, 5 (May 1990), 1343–1351.

[156] Lii, W. Modeling of delay variability due to random dopant fluctuation in nano-scale cmos inverter. In *2014 International Conference on Information Science, Electronics and Electrical Engineering* (April 2014), vol. 1, pp. 168–171.

[157] Lin, X., Wang, Y., Xie, Q., and Pedram, M. Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment. *IEEE Transactions on Services Computing 8*, 2 (March 2015), 175–186.

[158] Lloyd, J. R., Liniger, E., and Chen, S. T. Time dependent dielectric breakdown in a low-k interlevel dielectric. *Microelectronics Reliability 44* (2004), 1861–1865.

[159] Lofstead, J., Zheng, F., Liu, Q., Klasky, S., Oldfield, R., Kordenbrock, T., Schwan, K., and Wolf, M. Managing variability in the io performance of petascale storage systems. In *SC '10: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis* (Nov 2010), pp. 1–12.

[160] Lu, C., Stankovic, J. A., Son, S. H., and Tao, G. Feedback control real-time scheduling: Framework, modeling, and algorithms*. *The International Journal of Time-Critical Computing Systems 23*, 1–2 (July 2002), 85–126.

[161] Lu, C., Stankovic, J. A., Son, S. H., and Tao, G. Feedback control real-time scheduling: Framework, modeling, and algorithms*. *The International Journal of Time-Critical Computing Systems 23*, 1–2 (July 2002), 85–126.

[162] Lu, C., Stankovic, J. A., Tao, G., and Son, S. H. Design and evaluation of a feedback control edf scheduling algorithm. In *IEEE, 20th Real-Time Systems Symposium* (December 1999).

[163] Lu, S., Karnik, T., Srinivasa, G., Chao, K., Carmean, D., and Held, J. Scaling the "memory wall": Designer track. In *2012 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (Nov 2012), pp. 271–272.

[164] Luo, H., Tao, H., Ma, H., and Das, S. K. Data fusion with desired reliability in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems 22*, 3 (March 2011), 501–513.

[165] Lyons, R. E., and Vanderkulk, W. The use of triple-modular redundancy to improve computer reliability. *IBM J. Res. Dev. 6*, 2 (Apr. 1962), 200–209.

[166] M.A., T., R., D., and D.M, M. *Spectral Techniques in VLSI CAD*. Springer, Boston, USA, 2001.

[167] Ma, Z., Marchal, P., Scarpazza, D. P., Yang, P., Wong, C., Gómez, J. I., Himpe, S., Ykman-Couvreur, C., and Catthoor, F. Systematic methodology for real-time cost-effective mapping of dynamic concurrent task-based systems on heterogenous platforms. Springer, Netherlands, 2007.

[168] Madden, L., Wu, E., Kim, N., Banijamali, B., Abugharbieh, K., Ramalingam, S., and Wu, X. Advancing high performance heterogeneous integration through die stacking. In *2012 Proceedings of the ESSCIRC (ESSCIRC)* (Sep. 2012), pp. 18–24.

[169] Magnone, P., Crupi, F., Wils, N., Jain, R., Tuinhout, H., Andricciola, P., Giusi, G., and Fiegna, C. Impact of hot carriers on nmosfet variability in 45- and 65-nm cmos technologies. *IEEE Transactions on Electron Devices 58*, 8 (Aug 2011), 2347–2353.

[170] Malhotra, S., Narkhede, P., Shah, K., Makaraju, S., and Muniandi, S. A review of fault tolerant scheduling in multicore systems. *International Journal of Scientific & Technology Research 4* (05 2015), 132–136.

[171] Manhaeve, H., Harrod, P., Singh, A., Patel, C., Arnolc, R., and Appello, D. Current testing: Dead or alive? In *2013 18th IEEE European Test Symposium (ETS)* (May 2013), pp. 1–1.

[172] Manolache, S., Eles, P., and Peng, Z. Optimization of soft real-time systems with deadline miss ratio constraints. In *Proceedings. RTAS 2004. 10th IEEE Real-Time and Embedded Technology and Applications Symposium, 2004.* (May 2004), pp. 562–570.

[173] May, T. C., and Woods, M. H. Alpha-particle-induced soft errors in dynamic memories. *IEEE Transactions on Electron Devices 26*, 1 (Jan 1979), 2–9.

[174] Mcconaghy, T. Analog behavior in custom ic variation-aware design. pp. 146–148.

[175] Mehrotra, V. *Modeling the Effects of Systematic Process Variation on Circuit Performance*. PhD thesis, Cambridge, MA, USA, 2001. AAI0803331.

[176] Mentor Graphics. Calibre Verification User's Manua. Tech. rep., 2006.

[177] Mentor Graphics. QuestaSIM User's Manual. Tech. rep., 2011.

[178] Mentor Graphics. ModelSim User's Manual. Tech. rep., 2012.

[179] Merino, J., Bota, S., Picos, R., and Segura, J. Alternate characterization technique for static random-access memory static noise margin determination. *International Journal of Circuit Theory and Applications 41* (01 2012).

[180] Mintarno, E., Skaf, J., Zheng, R., Velamala, J. B., Cao, Y., Boyd, S., Dutton, R. W., and Mitra, S. Self-tuning for maximized lifetime energy-efficiency in the presence of circuit aging. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 30*, 5 (May 2011), 760–773.

[181] Misaka, A., Goda, A., Matsuoka, K., Umimoto, H., and Odanaka, S. A statistical critical dimension control at cmos cell level. In *International Electron Devices Meeting. Technical Digest* (Dec 1996), pp. 631–634.

[182] Mitchell, J., Henderson, D., Ahrens, G., and Villareal, J. IBM Power Platform Reliability, Availability and Serviceability (RAS). Tech. rep., IBM Corporation, June 2009.

[183] Mittal, K., Joshi, A., and Mutyam, M. Timing variation-aware scheduling and resource binding in high-level synthesis. *ACM Trans. Des. Autom. Electron. Syst. 16*, 4 (Oct. 2011), 40:1–40:19.

[184] Moreno, G., Narumanchi, S., Bennion, K., Waye, S., and DeVoto, D. Gaining traction: Thermal management and reliability of automotive electric traction-drive systems. *IEEE Electrification Magazine 2*, 2 (June 2014), 42–49.

[185] Mukherjee, S. *Architecture Design for Soft Errors*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008.

[186] Mukundan, J., Hunter, H., Kim, K.-h., Stuecheli, J., and Martínez, J. F. Understanding and mitigating refresh overheads in high-density ddr4 dram systems. *SIGARCH Comput. Archit. News 41*, 3 (June 2013), 48–59.

[187] Muller, S., Koal, T., Scharoba, S., Vierhaus, H. T., and Schölzel, M. A multi-layer software-based fault-tolerance approach for heterogenous multi-core systems. In *2015 16th Latin-American Test Symposium (LATS)* (March 2015), pp. 1–6.

[188] Munaga, S., and Catthoor, F. Systematic design principles for cost-effective hard constraint management in dynamic nonlinear systems. *Int. J. Adapt. Resilient Auton. Syst. 2*, 1 (Jan. 2011), 18–45.

[189] Narendra, S., Antoniadis, D., and De, V. Impact of using adaptive body bias to compensate die-to-die vt variation on within-die vt variation. In *Proceedings. 1999 International Symposium on Low Power Electronics and Design (Cat. No.99TH8477)* (Aug 1999), pp. 229–232.

[190] Nasir, S. B., and Raychowdhury, A. A Model Study of an All-Digital, Discrete-Time and Embedded Linear Regulator. *ArXiv e-prints* (2015).

[191] Noltsis, M., Catthoor, F., and Soudris, D. Proactive dvfs using dynamic scenarios to guarantee deadlines under error-recovery interventions. *Under Review*.

[192] Noltsis, M., Englezakis, P., Maragkoudaki, E., Nicopoulos, C., Rodopoulos, D., Catthoor, F., Sazeides, Y., Zoni, D., and Soudris, D. Fast estimations of failure probability over long time spans. In *Proceedings of the 14th IEEE/ACM International Symposium on Nanoscale Architectures* (New York, NY, USA, 2018), NANOARCH '18, Association for Computing Machinery, pp. 1—-6.

[193] Noltsis, M., Maragkoudaki, E., Rodopoulos, D., Catthoor, F., and Soudris, D. Failure probability of a finfet-based sram cell utilizing the most probable failure point. *Integration 69* (2019), 111 – 119.

[194] Noltsis, M., Rodopoulos, D., Catthoor, F., and Soudris, D. Classification of prior art on parametric reliability techniques for digital systems. *To be submitted*.

[195] Noltsis, M., Rodopoulos, D., Zompakis, N., Catthoor, F., and Soudris, D. Runtime slack creation for processor performance variability using system scenarios. *ACM Trans. Des. Autom. Electron. Syst. 23*, 2 (Dec. 2017), 24:1–24:23.

[196] Noltsis, M., Weckx, P., Rodopoulos, D., Cathoor, F., and Soudris, D. Accuracy of quasi-monte carlo technique in failure probability estimations. In *2016 International Conference on IC Design and Technology (ICICDT)* (June 2016), pp. 1–4.

[197] Noltsis, M., Zambelis, N., Catthoor, F., and Soudris, D. A closed-loop controller to ensure performance and temperature constraints for dynamic applications. *ACM Trans. Embed. Comput. Syst. 18*, 5 (Oct. 2019).

[198] Noltsis, M., Zambelis, N., Catthoor, F., and Soudris, D. A synergy of a closed-loop dvfs controller and cpu hot-plug for run-time thermal management in multicore systems. In *2019 29th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)* (July 2019), pp. 49–56.

[199] NXP. i.mx 6dual/6quad automotive and infotainment applications processors data sheet, document no. imx6dqaec 3/2014. Tech. rep., 2014.

[200] Orailoglu, A., and Karri, R. Coactive scheduling and checkpoint determination during high level synthesis of self-recovering microarchitectures. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems 2*, 3 (Sep. 1994), 304–311.

[201] Oral, S., Wang, F., Dillow, D., Miller, R., Shipman, G., Maxwell, D., Becklehimer, J., Larkin, J., and Henseler, D. Reducing application runtime variability on jaguar xt5. In *2010 Cray Users Group Conference (CUG)* (May 2010), pp. 1–7.

[202] Ouyang, X., Marcarelli, S., and Panda, D. K. Enhancing checkpoint performance with staging io and ssd. In *2010 International Workshop on Storage Network Architecture and Parallel I/Os* (May 2010), pp. 13–20.

[203] Park, S., Park, J., Shin, D., Wang, Y., Xie, Q., Pedram, M., and Chang, N. Accurate modeling of the delay and energy overhead of dynamic voltage and frequency scaling in modern microprocessors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 32*, 5 (May 2013), 695–708.

[204] Partridge, J. P., Hussey, B., Chen, J., and Gupta, A. Repair of circuits by laser seeding and constriction induced plating. *IEEE Transactions on Components, Hybrids, and Manufacturing Technology 15*, 2 (April 1992), 252–257.

[205] Partridge, S. L. Silicon-on-insulator technology. *IEE Proceedings I - Solid-State and Electron Devices 133*, 3 (June 1986), 66–76.

[206] Paul, B. C., Kunhyuk Kang, Kufluoglu, H., Alam, M. A., and Roy, K. Impact of nbti on the temporal performance degradation of digital circuits. *IEEE Electron Device Letters 26*, 8 (Aug 2005), 560–562.

[207] Paul, B. C., Kunhyuk Kang, Kufluoglu, H., Alam, M. A., and Roy, K. Temporal performance degradation under nbti: Estimation and design for improved reliability of nanoscale circuits. In *Proceedings of the Design Automation Test in Europe Conference* (March 2006), vol. 1, pp. 1–6.

[208] Pelgrom, M. J. M., Duinmaijer, A. C. J., and Welbers, A. P. G. Matching properties of mos transistors. *IEEE Journal of Solid-State Circuits 24*, 5 (Oct 1989), 1433–1439.

[209] Pering, T., Burd, T., and Brodersen, R. The simulation and evaluation of dynamic voltage scaling algorithms. In *Proceedings. 1998 International Symposium on Low Power Electronics and Design (IEEE Cat. No.98TH8379)* (Aug 1998), pp. 76–81.

[210] Pham, C., Estrada, Z., Cao, P., Kalbarczyk, Z., and Iyer, R. K. Reliability and security monitoring of virtual machines using hardware architectural invariants. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks* (June 2014), pp. 13–24.

[211] Pillai, P., and Shin, K. G. Real-time dynamic voltage scaling for low-power embedded operating systems. *SIGOPS Oper. Syst. Rev. 35*, 5 (Oct. 2001), 89–102.

[212] Pillai, P., and Shin, K. G. Real-time dynamic voltage scaling for low-power embedded operating systems. *SIGOPS Oper. Syst. Rev. 35*, 5 (Oct. 2001), 89–102.

[213] Plank, J. S., Beck, M., Kingsley, G., and Li, K. Libckpt: Transparent checkpointing under unix. Tech. rep., Knoxville, TN, USA, 1994.

[214] Poliakov, P., Blomme, P., Pret, A. V., Corbalan, M. M., Gronheid, R., Wiaux, V., Versluijs, J., Verkest, D., Van Houdt, J., and Dehaene, W. Spacer-defined euv lithography reducing variability of 12nm nand flash memories. In *2012 4th IEEE International Memory Workshop* (May 2012), pp. 1–4.

[215] Polze, A., Tröger, P., and Salfner, F. Timely virtual machine migration for pro-active fault tolerance. In *2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops* (March 2011), pp. 234–243.

[216] Porterfield, A., Fowler, R., Bhalachandra, S., Rountree, B., Deb, D., and Lewis, R. Application runtime variability and power optimization for exascale computers. In *ROSS* (2015).

[217] Powell, M. D., Biswas, A., Gupta, S., and Mukherjee, S. S. Architectural core salvaging in a multi-core processor for hard-error tolerance. *SIGARCH Comput. Archit. News 37*, 3 (June 2009), 93–104.

[218] Psychou, G., Rodopoulos, D., Sabry, M. M., Gemmeke, T., Atienza, D., Noll, T. G., and Catthoor, F. Classification of resilience techniques against functional errors at higher abstraction layers of digital systems. *ACM Comput. Surv. 50*, 4 (Oct. 2017), 50:1–50:38.

[219] Pusukuri, K. K., Gupta, R., and Bhuyan, L. N. Thread tranquilizer: Dynamically reducing performance variation. *ACM Trans. Archit. Code Optim. 8*, 4 (Jan. 2012), 46:1–46:21.

[220] Qureshi, M., Kim, D. H., Khan, S., Nair, P., and Mutlu, O. Avatar: A variable-retention-time (vrt) aware refresh for dram systems. In *DSN* (2015).

[221] Rahimi, A., Benini, L., and Gupta, R. K. Aging-aware compiler-directed vliw assignment for gpgpu architectures. In *Proceedings of the 50th Annual Design Automation Conference* (New York, NY, USA, 2013), DAC '13, ACM, pp. 16:1–16:6.

[222] Rana, M., and Canal, R. Ssfb: A highly-efficient and scalable simulation reduction technique for sram yield analysis. In *2014 Design, Automation Test in Europe Conference Exhibition (DATE)* (March 2014), pp. 1–6.

[223] Rasmussen, N. Improving rack cooling performance using airflow management blanking panels. `https://download.schneider-electric.com/files?p_Doc_Ref=SPD_SADE-5TPLKQ_EN`, 2005. White Paper.

[224] Raychaudhuri, S. Introduction to monte carlo simulation. In *2008 Winter Simulation Conference* (Dec 2008), pp. 91–100.

[225] Reddi, V. J., Kanev, S., Kim, W., Campanoni, S., Smith, M. D., Wei, G., and Brooks, D. Voltage smoothing: Characterizing and mitigating voltage noise in production processors via software-guided thread scheduling. In *2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture* (Dec 2010), pp. 77–88.

[226] Reddi, V. J., Kanev, S., Kim, W., Campanoni, S., Smith, M. D., Wei, G.-Y., and Brooks, D. Voltage smoothing: Characterizing and mitigating voltage noise in production processors via software-guided thread scheduling. In *43rd IEEE/ACM International Symposium on Microarchitecture* (December 2010).

[227] Rehman, S., Shafique, M., Kriebel, F., and Henkel, J. Raise: Reliability-aware instruction scheduling for unreliable hardware. In *17th Asia and South Pacific Design Automation Conference* (Jan 2012), pp. 671–676.

[228] Rodopoulos, D. *Modeling and Mitigation of Parametric Time-Dependent Variability in Digital Systems*. KU Leuven.Faculteit ingenieurswetenschappen, Leuven, 2016.

[229] Rodopoulos, D., Catthoor, F., and Soudris, D. Tackling performance variability due to ras mechanisms with pid-controlled dvfs. *IEEE CAL* (2014).

[230] Rodopoulos, D., Mahato, S. B., de Almeida Camargo, V. V., Kaczer, B., Catthoor, F., Cosemans, S., Groeseneken, G., Papanikolaou, A., and Soudris, D. Time and workload dependent device variability in circuit simulations. In *2011 IEEE International Conference on IC Design Technology* (May 2011), pp. 1–4.

[231] Rodopoulos, D., Sazeides, Y., Catthoor, F., Nicopoulos, C., and Soudris, D. Sensitivity of sram cell most probable snm failure point to time-dependent variability. In *IEEE SELSE Workshop, Austin–Texas* (2015).

[232] Rodopoulos, D., Sazeides, Y., Catthoor, F., Nicopoulos, C., and Soudris, D. Approximating standard cell delay distributions using the most probable failure point. In *Workshop on Early Reliability Modeling*

for Aging and Variability in Silicon Systems, Dresden, Germany (March 2016).

[233] Rodopoulos, D., Weckx, P., Noltsis, M., Catthoor, F., and Soudris, D. Atomistic pseudo-transient bti simulation with inherent workload memory. *IEEE Transactions on Device and Materials Reliability 14*, 2 (June 2014), 704–714.

[234] Rosedahl, T. On chip controller(occ) overview. `http://on-demand.gputechconf.com/gtc/2015/presentation/S5699-Todd-Rosedahl-OpenPOWER.pdf`, 2015. San Jose, California.

[235] Rzepka, S., Banerjee, K., Meusel, E., and Chenming Hu. Characterization of self-heating in advanced vlsi interconnect lines based on thermal finite element simulation. *IEEE Transactions on Components, Packaging, and Manufacturing Technology: Part A 21*, 3 (Sep. 1998), 406–411.

[236] Sabry, M. M., Atienza, D., and Catthoor, F. A hybrid hw-sw approach for intermittent error mitigation in streaming-based embedded systems. In *2012 Design, Automation Test in Europe Conference Exhibition (DATE)* (March 2012), pp. 1110–1113.

[237] Sabry, M. M., Coskun, A. K., Atienza, D., Rosing, T. x., and Brunschwiler, T. Energy-efficient multiobjective thermal control for liquid-cooled 3-d stacked architectures. *Trans. Comp.-Aided Des. Integ. Cir. Sys. 30*, 12 (Dec. 2011), 1883–1896.

[238] Sakata, A., Kato, S., Yano, Y., Toyoda, H., Kawanoue, T., Hatano, M., Wada, J., Yamada, N., Oki, T., Yamaguchi, H., Nakamura, N., Higashi, K., Yamada, M., Fujimaki, T., and Hasunuma, M. Copper line resistance control and reliability improvement by surface nitridation of ti barrier metal. In *2008 International Interconnect Technology Conference* (June 2008), pp. 165–167.

[239] Sakuma, K., Andry, P. S., Tsang, C. K., Sueoka, K., Oyama, Y., Patel, C., Dang, B., Wright, S. L., Webb, B. C., Sprogis, E., Polastre, R., Horton, R., and Knickerbocker, J. U. Characterization of stacked die using die-to-wafer integration for high yield and throughput. In *2008 58th Electronic Components and Technology Conference* (May 2008), pp. 18–23.

[240] Sanjay Pant, Blaauw, D., Zolotov, V., Sundareswaran, S., and Panda, R. Vectorless analysis of supply noise induced delay variation. In *ICCAD-2003. International Conference on Computer Aided Design (IEEE Cat. No.03CH37486)* (Nov 2003), pp. 184–191.

[241] Sanz, C., Prieto, M., Gómez, J. I., Papanikolaou, A., Miranda, M., and Catthoor, F. Combining system scenarios and configurable memories to tolerate unpredictability. *ACM Trans. Des. Autom. Electron. Syst. 13*, 3 (July 2008), 49:1–49:7.

[242] Sarno, C., and Tantolin, C. Integration, cooling and packaging issues for aerospace equipments. In *2010 Design, Automation Test in Europe Conference Exhibition (DATE 2010)* (March 2010), pp. 1225–1230.

[243] Schroder, D. K., and Babcock, J. A. Negative bias temperature instability: Road to cross in deep submicron silicon semiconductor manufacturing. *Journal of Applied Physics 94*, 1 (2003), 1–18.

[244] Seevinck, E., List, F. J., and Lohstroh, J. Static-noise margin analysis of mos sram cells. *IEEE Journal of Solid-State Circuits 22*, 5 (Oct 1987), 748–754.

[245] Semiconductors, N. i.mx 6dual/6quad applications processor reference manual. `http://www.nxp.com/assets/documents/data/en/reference-manuals/IMX6DQRM.pdf`, 2015. Rev 3.

[246] Semiconductors, N. i.mx 6dual/6quad automotive and infotainment applications processors. `https://www.nxp.com/docs/en/data-sheet/IMX6DQAEC.pdf`, 2018. Data Sheet: Technical Data.

[247] Sharma, P., Tyaginov, S., Rauch, S. E., Franco, J., Makarov, A., Vexler, M. I., Kaczer, B., and Grasser, T. Hot-carrier degradation modeling of decananometer nmosfets using the drift-diffusion approach. *IEEE Electron Device Letters 38*, 2 (Feb 2017), 160–163.

[248] Shen, R., Tan, S. X. ., and Liu, X. A new voltage binning technique for yield improvement based on graph theory. In *Thirteenth International Symposium on Quality Electronic Design (ISQED)* (March 2012), pp. 243–248.

[249] Shin, C. *Variation-Aware Advanced CMOS Devices and SRAM*, vol. 56. Springer, Dordrecht, 01 2016.

[250] Shin, C., Sun, X., and Liu, T. K. Study of random-dopant-fluctuation (rdf) effects for the trigate bulk mosfet. *IEEE Transactions on Electron Devices 56*, 7 (July 2009), 1538–1542.

[251] Shin, J., Zyuban, V., Bose, P., and Pinkston, T. A proactive wearout recovery approach for exploiting microarchitectural redundancy to extend cache sram lifetime. In *ISCA* (June 2008), pp. 353–362.

[252] Shyue-Kung Lu, Yu-Chen Tsai, Hsu, C. ., Kuo-Hua Wang, and Cheng-Wen Wu. Efficient built-in redundancy analysis for embedded memories with 2-d redundancy. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems 14*, 1 (Jan 2006), 34–42.

[253] Siewiorek, D. P., and Swarz, R. S. *Reliable Computer Systems (3rd Ed.): Design and Evaluation*. A. K. Peters, Ltd., Natick, MA, USA, 1998.

[254] Singh, I., Qin, I., Xu, H., Huynh, C., Low, S., Clauberg, H., Chylak, B., and Acoff, V. L. Pd-coated cu wire bonding technology: Chip design, process optimization, production qualification and reliability test for high reliability semiconductor devices. In *2012 IEEE 62nd Electronic Components and Technology Conference* (May 2012), pp. 1089–1096.

[255] Slye, J. H., and Elnozahy, E. N. Supporting nondeterministic execution in fault-tolerant systems. In *Proceedings of Annual Symposium on Fault Tolerant Computing* (June 1996), pp. 250–259.

[256] Song, J., Kim, S., Bae, B., Kim, H., Kim, J. J., Kim, D. B., and Kim, J. Analysis of wireless power transfer system design on active silicon interposer for low voltage applications in 3d-ic. In *2014 IEEE 23rd Conference on Electrical Performance of Electronic Packaging and Systems* (Oct 2014), pp. 223–226.

[257] Stathis, J. H. Reliability limits for the gate insulator in cmos technology. *IBM Journal of Research and Development 46*, 2.3 (March 2002), 265–286.

[258] Stok, L., and Koehl, J. Structured cad: technology closure for modern asics [tutorial]. In *Proceedings Design, Automation and Test in Europe Conference and Exhibition* (Feb 2004), vol. 1, pp. xxxi–xxxi.

[259] Stolk, P. A., Widdershoven, F. P., and Klaassen, D. B. M. Modeling statistical dopant fluctuations in mos transistors. *IEEE Transactions on Electron Devices 45*, 9 (Sep. 1998), 1960–1971.

[260] Sukharev, V., and Najm, F. N. Electromigration check: Where the design and reliability methodologies meet. *IEEE Transactions on Device and Materials Reliability 18*, 4 (Dec 2018), 498–507.

[261] Suresh, V. B., and Kundu, S. On analyzing and mitigating sram ber due to random thermal noise. In *2013 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)* (Aug 2013), pp. 159–164.

[262] Swaminathan, V., and Chakrabarty, K. Pruning-based energy-optimal device scheduling for hard real-time systems. In *Proceedings*

of the Tenth International Symposium on Hardware/Software Codesign. CODES 2002 (IEEE Cat. No.02TH8627) (May 2002), pp. 175–180.

[263] Synopsys. PrimeTimeGolden Timing Signoff Solution and Environment. Tech. rep., 2017. Datasheet.

[264] Synopsys. Transistor Level Static Timing Analysis with NanoTime. Tech. rep., 2019. White Paper.

[265] Tang, H. H. K. Nuclear physics of cosmic ray interaction with semiconductor materials: Particle-induced soft errors from a physicist's perspective. *IBM Journal of Research and Development 40*, 1 (Jan 1996), 91–108.

[266] Tao, J., Cheung, N. W., and Hu, C. Electromigration characteristics of copper interconnects. *IEEE Electron Device Letters 14*, 5 (May 1993), 249–251.

[267] Tarsa, S. J., Kumar, A. P., and Kung, H. T. Workload prediction for adaptive power scaling using deep learning. In *2014 IEEE International Conference on IC Design Technology* (May 2014), pp. 1–5.

[268] Thorpe, T. W. *Computerized Circuit Analysis with SPICE (1st Ed.): A Complete Guide to SPICE with Applications*. Wiley-Interscience, USA, 1992.

[269] Tirumurti, C., Kundu, S., Sur-Kolay, S., and Yi-Shing Chang. A modeling approach for addressing power supply switching noise related failures of integrated circuits. In *Proceedings Design, Automation and Test in Europe Conference and Exhibition* (Feb 2004), vol. 2, pp. 1078–1083 Vol.2.

[270] Toledano-Luque, M., Kaczer, B., Roussel, P. J., Grasser, T., Wirth, G. I., Franco, J., Vrancken, C., Horiguchi, N., and Groeseneken, G. Response of a single trap to ac negative bias temperature stress. In *2011 International Reliability Physics Symposium* (April 2011), pp. 4A.2.1–4A.2.8.

[271] Torii, K., Kitajima, H., Arikado, T., Shiraishi, K., Miyazaki, S., Yamabe, K., Boero, M., Chikyow, T., and Yamada, K. Physical model of bti, tddb and silc in hfo/sub 2/-based high-k gate dielectrics. In *IEDM Technical Digest. IEEE International Electron Devices Meeting, 2004.* (Dec 2004), pp. 129–132.

[272] Tschanz, J., Bowman, K., Lu, S.-L., Aseron, P., Khellah, M., Raychowdhury, A., Geuskens, B., Tokunaga, C., Wilkerson, C.,

Karnik, T., and De, V. A 45nm resilient and adaptive microprocessor core for dynamic variation tolerance. In *IEEE International Solid-State Circuits Conference* (February 2010).

[273] Tschanz, J. W., Kao, J. T., Narendra, S. G., Nair, R., Antoniadis, D. A., Chandrakasan, A. P., and De, V. 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 37*, 11 (Nov 2002), 1396–1402.

[274] Tschanz, J. W., Narendra, S., Nair, R., and De, V. Effectiveness of adaptive supply voltage and body bias for reducing impact of parameter variations in low power and high performance microprocessors. *IEEE Journal of Solid-State Circuits 38*, 5 (May 2003), 826–829.

[275] Tuck-Boon Chan, Kahng, A. B., and Jiajia Li. Reliability-constrained die stacking order in 3dics under manufacturing variability. In *International Symposium on Quality Electronic Design (ISQED)* (March 2013), pp. 16–23.

[276] v. d. Brüggen, G., Chen, K., Huang, W., and Chen, J. Systems with dynamic real-time guarantees in uncertain and faulty execution environments. In *2016 IEEE Real-Time Systems Symposium (RTSS)* (Nov 2016), pp. 303–314.

[277] Van Gils. A triple modular redundancy technique providing multiple-bit error protection without using extra redundancy. *IEEE Transactions on Computers C-35*, 7 (July 1986), 623–631.

[278] van Santen, V. M., Amrouch, H., Martin-Martinez, J., Nafria, M., and Henkel, J. Designing guardbands for instantaneous aging effects. In *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)* (June 2016), pp. 1–6.

[279] Vandeputte, F., Eeckhout, L., and De Bosschere, K. A detailed study on phase predictors. In *Euro-Par 2005 Parallel Processing* (Berlin, Heidelberg, 2005), J. C. Cunha and P. D. Medeiros, Eds., Springer Berlin Heidelberg, pp. 571–581.

[280] Veetil, V., Chang, Y., Sylvester, D., and Blaauw, D. Efficient smart monte carlo based ssta on graphics processing units with improved resource utilization. In *Design Automation Conference* (June 2010), pp. 793–798.

[281] Velamala, J. B., Sutaria, K. B., Shimizu, H., Awano, H., Sato, T., Wirth, G., and Cao, Y. Compact modeling of statistical bti under

trapping/detrapping. *IEEE Transactions on Electron Devices 60*, 11 (Nov 2013), 3645–3654.

[282] Venkatachalam, V., and Franz, M. Power reduction techniques for microprocessor systems. *ACM Comput. Surv. 37*, 3 (Sept. 2005), 195–237.

[283] Venkataraman, S., Santos, R., Kumar, A., and Kuijsten, J. Hardware task migration module for improved fault tolerance and predictability. In *2015 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)* (July 2015), pp. 197–202.

[284] Vollrath, J. Signal margin analysis for dram sense amplifiers. In *Proceedings First IEEE International Workshop on Electronic Design, Test and Applications '2002* (Jan 2002), pp. 123–127.

[285] von Tils, V. Design requirements for automotive reliability. `http://www-g.eng.cam.ac.uk/robuspic/pub_present/ESSDERC06/6-ROBUSPIC-Workshop-ESSDERC06-VVonTils.pdf`, 2006. Montreux, Switzerland.

[286] Wakerly, J. F. Microcomputer reliability improvement using triple-modular redundancy. *Proceedings of the IEEE 64*, 6 (June 1976), 889–895.

[287] Wang, S., and Bettati, R. Reactive speed control in temperature-constrained real-time systems. *Real-Time Syst. 39*, 1-3 (Aug. 2008), 73–95.

[288] Wang, W., Yang, S., Bhardwaj, S., Vrudhula, S., Liu, F., and Cao, Y. The impact of nbti effect on combinational circuit: Modeling, simulation, and analysis. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems 18*, 2 (Feb 2010), 173–183.

[289] Wang, X., Cheng, B., Kuang, J. B., Nassif, S., Brown, A. R., Millar, C., and Asenov, A. Statistical variability and reliability and the impact on corresponding 6t-sram cell design for a 14-nm node soi finfet technology. *IEEE Design Test 30*, 6 (Dec 2013), 18–28.

[290] Weckx, P., Kaczer, B., Toledano-Luque, M., Raghavan, P., Franco, J., Roussel, P. J., Groeseneken, G., and Catthoor, F. Implications of bti-induced time-dependent statistics on yield estimation of digital circuits. *IEEE Transactions on Electron Devices 61*, 3 (March 2014), 666–673.

[291] Wen Wu, Xiaodong Duan, and Yuan, J. S. Modeling of time-dependent dielectric breakdown in copper metallization. *IEEE Transactions on Device and Materials Reliability 3*, 2 (June 2003), 26–30.

[292] Weste, N., and Harris, D. *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. Addison-Wesley Publishing Company, USA, 2010.

[293] Won, J., Chen, X., Gratz, P., Hu, J., and Soteriou, V. Up by their bootstraps: Online learning in artificial neural networks for cmp uncore power management. In *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)* (Feb 2014), pp. 308–319.

[294] Wu, Y.-T. Methods for efficient probabilistic analysis of systems with large number of random variables. In *Symposium on Multidisciplinary Analysis Optimization* (1998).

[295] Wurm, S., Seidel, P., Peski, C. V., He, L., Han, H., Kearney, P., and Cho, W. Euv mask infrastructure challenges. In *23rd European Mask and Lithography Conference* (Jan 2007), pp. 1–12.

[296] Xu, X., Cline, B., Yeric, G., Yu, B., and Pan, D. Pd14 self-aligned double patterning aware pin access and standard cell layout co-optimization. vol. 34, pp. 101–108.

[297] Xuejue Huang, Wen-Chin Lee, Charles Kuo, Hisamoto, D., Leland Chang, Kedzierski, J., Anderson, E., Takeuchi, H., Yang-Kyu Choi, Asano, K., Subramanian, V., Tsu-Jae King, Bokor, J., and Chenming Hu. Sub 50-nm finfet: Pmos. In *International Electron Devices Meeting 1999. Technical Digest (Cat. No.99CH36318)* (Dec 1999), pp. 67–70.

[298] Yan, J., and Zhang, W. Compiler-guided register reliability improvement against soft errors. In *Proceedings of the 5th ACM International Conference on Embedded Software* (New York, NY, USA, 2005), EMSOFT '05, ACM, pp. 203–209.

[299] Yang, C. H., Chen, S. C., Tsai, Y. S., Lu, R., and Lee, Y. . The physical explanation of tddb power law lifetime model through oxygen vacancy trap investigations in hkmg nmos finfet devices. In *2017 IEEE International Reliability Physics Symposium (IRPS)* (April 2017), pp. 3C–4.1–3C–4.6.

[300] Yang, P., Wong, C., Marchal, P., Catthoor, F., Desmet, D., Verkest, D., and Lauwereins, R. Energy-aware runtime scheduling for embedded-multiprocessor socs. *IEEE Design Test of Computers 18*, 5 (Sep. 2001), 46–58.

[301] Youngsoo Shin, and Kiyoung Choi. Power conscious fixed priority scheduling for hard real-time systems. In *Proceedings 1999 Design*

Automation Conference (Cat. No. 99CH36361) (June 1999), pp. 134–139.

[302] Zanini, F., Atienza, D., Jones, C. N., Benini, L., and De Micheli, G. Online thermal control methods for multiprocessor systems. *ACM Trans. Des. Autom. Electron. Syst. 18*, 1 (Jan. 2013), 6:1–6:26.

[303] Ziegler, J., and Nichols, N. Optimum setting for automatic controllers. *Transactions of ASME* (06 1993), 759–768.

[304] Ziegler, J. F. Terrestrial cosmic rays. *IBM Journal of Research and Development 40*, 1 (Jan 1996), 19–39.

[305] Zolotov, V., Visweswariah, C., and Xiong, J. Voltage binning under process variation. In *2009 IEEE/ACM International Conference on Computer-Aided Design - Digest of Technical Papers* (Nov 2009), pp. 425–432.

[306] Zompakis, N., Noltsis, M., Ndreu, L., Hadjilambrou, Z., Englezakis, P., Nikolaou, P., Portero, A., Libutti, S., Massari, G., Sassi, F., Bacchini, A., Nicopoulos, C., Sazeides, Y., Vavrik, R., Golasowski, M., Sevcik, J., Vondrak, V., Catthoor, F., Fornaciari, W., and Soudris, D. Harpa: Tackling physically induced performance variability. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2017* (March 2017), pp. 97–102.

[307] Zompakis, N., Noltsis, M., Rodopoulos, D., Catthoor, F., and Soudris, D. Energy efficient adaptive approach for dependable performance in the presence of timing interference. In *Proceedings of the on Great Lakes Symposium on VLSI 2017* (New York, NY, USA, 2017), GLSVLSI '17, Association for Computing Machinery, pp. 209—-214.

[308] Zoni, D., Canidio, A., Fornaciari, W., Englezakis, P., Nicopoulos, C., and Sazeides, Y. Blackout: Enabling fine-grained power gating of buffers in network-on-chip routers. *Journal of Parallel and Distributed Computing 104* (2017), 130 – 145.

[309] Zoni, D., and Fornaciari, W. Sensor-wise methodology to face nbti stress of noc buffers. In *DATE* (March 2013), pp. 1038–1043.

Curriculum Vitae

Michail Noltsis was born in Naousa, Greece, in 1990. He graduated from the School of Electrical and Computer Engineering (ECE) of the National Technical University of Athens (NTUA) in 2014, following the specialization of communication, electronics and information systems. Until 2020, he is serving as a dual PhD student at MicroLab-ECE-NTUA and Katholieke Universiteit Leuven (KUL) while he is also a researcher for the Institute of Communication and Computer Systems (ICCS) of ECE-NTUA. Michail has co-authored more than 10 papers, published in international peer reviewed conferences and journals and he is also the co-author of 5 book Chapters. During his PhD studies, he supervised 2 NTUA Thesis for undergraduate students and has been involved in the FP7-612029-HARPA project. His research interests include: functional and parametric processor reliability, and circuit variability and aging for deca-nanometer nodes.

List of publications

Journals:

JP-1 Dimitrios. Rodopoulos, P. Weckx, **M. Noltsis**, F. Catthoor and D. Soudris, "Atomistic Pseudo-Transient BTI Simulation With Inherent Workload Memory", in *IEEE Transactions on Device and Materials Reliability*, vol. 14, June 2014.

JP-2 **Michail Noltsis**, D. Rodopoulos, N. Zompakis, F. Catthoor, and D. Soudris. 2017. "Runtime Slack Creation for Processor Performance Variability using System Scenarios", in *ACM Trans. Des. Autom. Electron Syst. (TODAES)*, Article 24, December 2017.

JP-3 **Michail Noltsis**, E. Maragkoudaki, D. Rodopoulos, F. Catthoor and D. Soudris, "Failure probability of a FinFET-based SRAM cell utilizing the most probable failure point", in *Integration, the VLSI Journal*, Volume 69, June 2019.

JP-4 **Michail Noltsis**, N. Zambelis, F. Catthoor, and D. Soudris, "A Closed-Loop Controller to Ensure Performance and Temperature Constraints for Dynamic Applications", in *ACM Trans. Embed. Comput. Syst. (TECS)*, Article 40, October 2019.

JP-5 **Michail Noltsis**, F. Catthoor and D. Soudris, "Proactive DVFS Using Dynamic Scenarios to Guarantee Deadlines under Error-Recovery Interventions", *Submitted in ACM Trans. Embed. Comput. Syst. (TECS)*.

JP-6 **Michail Noltsis**, D. Rodopoulos, F. Catthoor and D. Soudris, "Classification of prior art on parametric reliability techniques for digital systems", *To Be Submitted*

Conferences:

CP-1 **Michail Noltsis**, P. Weckx, D. Rodopoulos, F. Cathoor and D. Soudris, "Accuracy of Quasi-Monte Carlo technique in failure probability estimations", in *International Conference on IC Design and Technology (ICICDT)*, Ho Chi Minh City, 2016.

CP-2 Nikolaos Zompakis, **M. Noltsis**, D. Rodopoulos, F. Catthoor, and D. Soudris, "Energy Efficient Adaptive Approach for Dependable Performance in the presence of Timing Interference", in *Proceedings of the on Great Lakes Symposium on VLSI (GLSVLSI)*, Alberta, 2017.

CP-3 Nikolaos Zompakis, **M. Noltsis** et al., "HARPA: Tackling physically induced performance variability", in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Lausanne, 2017.

CP-4 **Michail Noltsis**, E. Maragkoudaki, D. Rodopoulos, F. Catthoor and D. Soudris, "Failure probability of a FinFET-based SRAM cell utilizing the most probable failure point", in *27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, Thessaloniki, 2017.

CP-5 **Michail Noltsis** et al., "Fast Estimations of Failure Probability Over Long Time Spans", in *Proceedings of the 14th IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, Athens, 2018.

CP-6 **Michail Noltsis**, N. Zambelis, F. Catthoor and D. Soudris, "A Synergy of a Closed-Loop DVFS Controller and CPU Hot-Plug For Run-Time Thermal Management in Multicore Systems", in *29th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, Rhodes, 2019.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
ESAT
10 Kasteelpark Arenberg
B-3001 Leuven

NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF EL. AND COMPUTER ENGINEERING
9 HEROON POLITECHNIOU, ATHENS