HARPA

Harnessing Performance Variability

HARPA

Harnessing Performance Variability

Project ref.	FP7-612069
Call ref.	FP7-ICT-2013-10
Activity	ICT-10-3.4

SotA RT Mitigation of Performance Variability

Dimitrios Rodopoulos and Dimitrios Soudris
ICCS, GR

Report Number:	D2.1
Version:	4.14
Date:	April 2, 2014

Revisions List

Date	Version	Author(s)	Description
April 2, 2014	4.14	Dimitrios Rodopoulos	First Draft
April 14, 2014	4.14b	Dimitrios Rodopoulos	Including feedback from internal Reviewer

Executive Summary

Modern digital systems have to fulfill a plethora of reliability specifications, regarding their functionality, performance and quality cost. Given the digital nature of these systems, marginal design is typically chosen in order to mask the fluctuation of operation parameters. When that fluctuation surpasses the implemented margins, a functional violation is materialized. In the current report, we explore the research domain of performance variability, addressing equally parametric and functional reliability concerns.

Initially, we introduce basic concepts related to the field. We elaborate on the term of “performance variability” and show its connotations for both the Embedded Computing (EC) and High Performance Computing (HPC) alike. Then, we reflect on the interplay between functional (related to binary correctness) and parametric (related to parametric performance targets) reliability violations. Through this discussion, we demonstrate how both types of violations are correlated and how each type may propagate to the other under certain performance variability conditions. Furthermore, we perform a brief presentation of state-of-the-art performance dependability metrics, while citing representative prior art samples. In any case, the intention of this project is to remain aligned with industry standard reliability metrics, thus enabling direct comparison of the technical contributions with prior art.

After the theoretical introduction, we turn to the classification of state-of-the-art samples. We employ a systematic classification methodology. This addresses the target research domain in a top-down fashion by building complementary categories based on binary splits. Prior art samples are seamlessly mapped to the leaves of the resulting binary tree, thus substantiating the completeness of our classification. Hybrid and cross-layer approaches are fully captured, given the orthogonality of the classification leaves. The classification framework that we present in this report is very broad covering all possible aspects of performance variability mitigation. However, for the sake of brevity and readability, the instantiation of prior art is performed with a moderate number of representative samples.

Within our prior art classification, we identify sub-domains that are of interest to the various Work Packages of the HARPA project. In such cases, we populate the respective prior art classes with more literature samples and/or comment on the current trends identified in these areas.

From the prior art analysis, we draw certain conclusions, relevant to the HARPA project: (1) The wealth of information available at the technology level, regarding performance variability, needs to be propagated to higher abstraction levels, to enable tackling of reliability concerns in the deca-nanometer era. (2) As a result, reactive techniques have to be complemented with proactive countermeasures that can estimate or even predict variability trends of a digital system. (3) Many-core and HPC computing relies mostly on software for the bookkeeping and maintenance of the respective systems. Careful placement of hardware knobs and monitors across different system packages may improve the responsiveness of performance variability mitigation in this context.

Table of Contents

Table of Figures

1 Introduction

Reliability has always been a major concern for the design of digital systems. Modern computing has to meet reliability specifications, while the dimensions of semiconductor devices reach tens of nanometers. This downscaling trend brings about a variety of mechanisms that cause fluctuations of operating parameters of digital systems. The increasing complexity of single- and multi-chip computing platforms also contributes to the variability of system performance, responsiveness and quality cost. As a result, Embedded Computing (EC) and High Performance Computing (HPC) alike, have to respect performance dependability constraints throughout the system's lifetime.

The current report aims to classify mitigation techniques against performance variability in digital systems. Firstly, it is important to distinguish the parametric vs. functional element of target reliability violations. On the one hand, some reliability violations can be perceived as *functional*, namely related to binary corruption or complete denial of service. On the other hand, *parametric* reliability violations refer to the fluctuation of operation parameters of a digital system, like delay, power consumption etc. Subsection 2.2 elaborates on the differences between these classes of violations. Given the variety of reliability violations found in modern digital systems, it is necessary to restrict the scope of the current state-of-the-art review with the following boundaries:

1. We refrain from reliability analysis/modeling techniques [1], since this domain is complementary to performance variability mitigation.
2. Parametric reliability violations that are not related to electronic operation as such are assumed out of scope (e.g. mechanical stress, relative humidity, corrosion etc).
3. Violations due to ill-specified systems or due to “intentional malicious faults” [2] are out of scope.

In the current report, we employ a systematic classification methodology in order to present the target research domain in a complete way. We break down the target domain using binary splits. This process produces a binary tree, the leaves of which constitute orthogonal classes, where state-of-the-art works can be classified. In the current report, we document the splitting process and instantiate representative papers for each classification leaf. That way, our survey is unique among related classifications, given that it approaches the performance variability mitigation in a *top-down* fashion. The complementarity of the resulting categories allows proper presentation of hybrid or cross-layer approaches, which borrow elements from many categories.

The breadth-first classification implemented in this report is a very useful tool for the exploration of established performance variability mitigation techniques and the research towards new ones. Without restricting to a specific subset of performance dependability, this survey provides guidance across the entire field. Approaches that are saturated with prior art can be identified and regions that have received reduced attention can be isolated. That way, our categorization can provide a complete view of the performance variability mitigation research domain. It also gives hints towards sub-domains that invite meaningful contribution. From the prior art analysis, it will become apparent that the contributions of the HARPA project are timely positioned among such sub-domains that have been scarcely addressed by state-of-the-art approaches.

The prior art review of this report covers all aspects of performance variability mitigation. It provides a solid background of terminology related to performance variability, covering all aspects of mitigation. Thus, process technology and circuit/architecture solutions are briefly addressed. However, we place increased emphasis on run time mitigation of performance variability. As a result, the respective prior subcategory is populated with more samples of prior art and is, in general, more elaborate.

The current report is organized in the following way: Section 2 presents some basic concepts, necessary for the current literature review. It covers the parametric vs. functional nature of dependability violations, the systematic methodology that we employ in this survey as well as an overview of our classification. The next two Sections cover the actual prior art review, which is the goal of the current deliverable. In Section 3 we present techniques that mitigate performance variability within a single silicon die. Techniques that guarantee performance dependability across a set of silicon dies are addressed in Section 4. Finally, a discussion of the literature review findings, along with a set of conclusions is drawn in Section 5.

2 Basic Concepts

In this Section, we present some fundamental ideas that will aid the prior art presentation. In Subsection 2.2, we reflect on the distinction between functional and parametric reliability violations and show how both can be culprits of performance variability. In Subsection 2.3, we present the principles of the state-of-the-art categorization that is implemented in this report. Subsection 2.4 focuses on the metrics that are typically used to assess the performance dependability of a digital system. Finally Subsection 2.5 presents an overview of the state-of-the-art classification performed in the current report.

2.1 What is Performance Variability?

Performance variability is the observable variation in quality cost (e.g. energy budget) and/or delay (e.g. latency) observed in a digital system, under specific workload and operating conditions. The observable quality results to a very important distinction among digital systems, based on synchronization. A digital system is called synchronous as long as its I/O is expected to arrive at strictly pre-defined intervals (equal to a “global clock” period). In the opposite case, the system is called asynchronous. It is important not to associate the term asynchronous only with systems that don't have a clock [3]. It may very well be that a system contains clocked components, which communicate among them in an asynchronous way (e.g. message passing or common memory hierarchies). We carry on with the following observations:

- Variability in the quality cost is observable in both synchronous and asynchronous contexts, both at design time (DT) and at run time (RT).
- In the asynchronous context, we can detect variability in delay, since the latency of each task is measurable and not masked by a “global clock”.
- In the synchronous domain, the latency of each executed “task” is observable *only at design time*. At runtime, we can only observe whether the deadline is met or not.

These observations have direct implications for the considered ES and HPC domains: HPC systems are composed of clocked systems (e.g. nodes of a massively parallel system), which communicate in an *asynchronous* fashion to produce their output (e.g. message passing). Variable latency at the output of the HPC system degrades the offered *quality of service* but does not destroy the system's functionality. However, this very latency can be observed, since an HPC system does not deliver output at a predefined rate.

On the contrary, embedded systems have *hard deadline* constraints that have to be met at all times during the system's operation [4] [5]. The system's state is observable *only* at these instances at runtime. As a result, we can only monitor task latency at design time. At runtime, we can only check whether the deadline is met or not. A deadline miss at runtime may lead to functional errors (e.g. corrupt data) or functional failures (e.g. denial of service). Based on this discussion, we can perceive performance variability in different ways, according to the context (synchronous vs. asynchronous) and abstraction level. Some examples are illustrated in Table 1 (appropriate system monitors are assumed for each case). The observations made in this Subsection are summarized in Figure 1.

Table 1: Perceiving performance dependability according to level of system synchronization

Digital System Case	**Context**	**Deadline**	**Observability of …**		
			Functionality	**Delay**	**Energy**
Circuit w/o Clock	Asynchronous	-	DT & RT		
Circuit w/ Clock	Synchronous	Clock Period	DT & RT	-	DT & RT
Embedded Video Decoder	Synchronous	FPS Rate	DT & RT	-	DT & RT
Distributed MapReduce	Asynchronous	-	DT & RT		

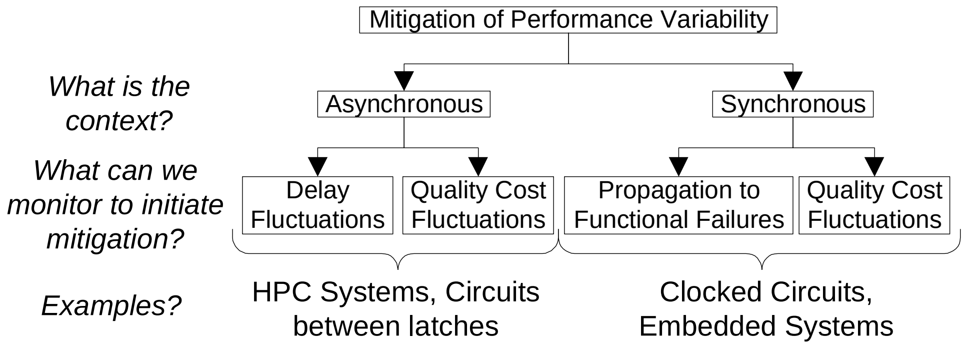

Mitigation of Performance Variability
What is the context?
Asynchronous
Synchronous
What can we monitor to initiate mitigation?
Delay Fluctuations
Quality Cost Fluctuations
Propagation to Functional Failures
Quality Cost Fluctuations
Examples?
HPC Systems, Circuits between latches
Clocked Circuits, Embedded Systems

Figure 1: Mitigation of performance variability is a highly context-specific task

Finally, it is very important to reflect on interplay between the causes and challenges of performance variability. As already indicated in Figure 1, this concept is highly context dependent. As a result, we will present in Table 2 some representative examples of the causes and effects of performance variability, while properly accounting for the appropriate EC or HPC context.

Table 2: Representative examples of the causes and effects of performance variability

Performance Variability	**Context**		**Cause**	**Effect**
	EC	**HPC**		
ΔV_{th}	X		Bias Temperature Instability [6]	*Delay* fluctuations of the combinational circuit – clock violations – inability to scale the clock
ΔR_{wire}	X		Electro-Migration [7]	
$\Delta I_{leakage}$	X	X	Non-uniform temperature spread [8]	Increasing static energy consumption
URL Fetch		X	Workload variability, time-sharing [9]	Poor quality of service
I/O Variability		X	Time-sharing of resources [10]	Overall system performance variability

2.2 Functional vs. Parametric Reliability Violations

One of the most crucial concepts of the current state-of-the-art review is the isolation of parametric reliability violations as one of the problems that prior art attempts to solve. Its distinction from the functional counterpart is equally important and will be addressed in the current Subsection, based on the following two Definitions:

A *parametric reliability violation* is the fluctuation of a system's operation parameters, which is quantified using a *fidelity metric* [11]. Examples include delay, power consumption, signal-to-noise ratio (SNR) [12], etc. It is possible to observe parametric reliability violations at various abstraction levels [13]. In the artist's impression of Figure 2, we see such a violation for a hypothetical $V_{out}(t)$ signal.

A *functional reliability violation* is the deviation from expected system behavior in terms of service delivery or binary correctness. Examples include silent data corruptions [14], or in the worst case, a total denial of service. We can illustrate this concept with Figure 3. A deadline imposed on the hypothetical $V_{out}(t)$ signal (e.g. assuming the case of a clocked system) translates a subset of parametric violations to functional ones.

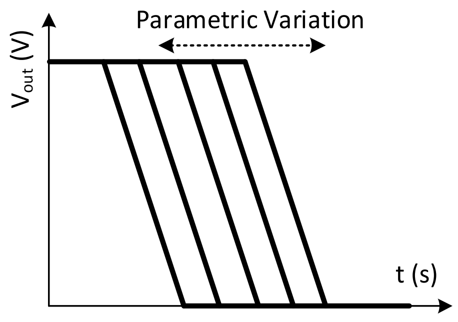


Figure 2: Variable signal delay originating from various sources is perceived as a parametric reliability violation.

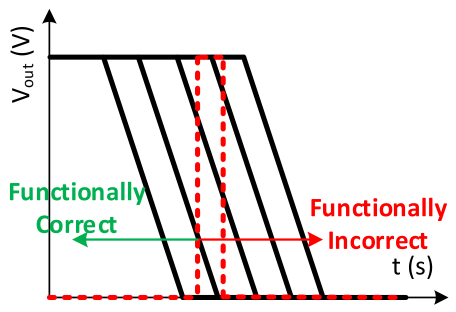


Figure 3: In a clocked system, some parametric fluctuations materialize as functional reliability violations.

Even though the above Definitions are strictly complementary, it is important to realize that citing a reliability violation as parametric or functional is highly *context dependent*. Let us assume the example of Figure 4: A system that implements a video decoder is suffering from bit flips of the Silent Data Corruption (SDC) class [14], which are functional violations according to the Definitions above. However, given that the decoding operation of the system is not interrupted (SDC class), many of these binary corruption may cause output distortion. At the application level, this distortion is perceived as a degradation of the SNR fidelity metric, assuming that no error correction codes are in place in the system. That way, at the application level, a seemingly functional violation appears as parametric.

As a result, it is very important to define the context before assigning the functional or parametric attribute to a reliability violation. In the previous example, we can observe a functional violation at the architecture or circuit level of the decoder and a parametric violation at the level of the application output. In the general context of performance variability, it is very important to realize the interplay between functional and parametric reliability violations. Thus, saturation of functional violations may lead to performance degradation. Conversely, performance variability may lead to functional violations and most severely a complete denial of service.

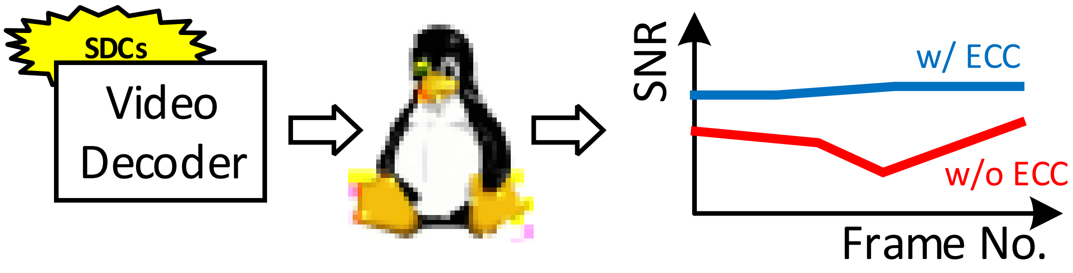


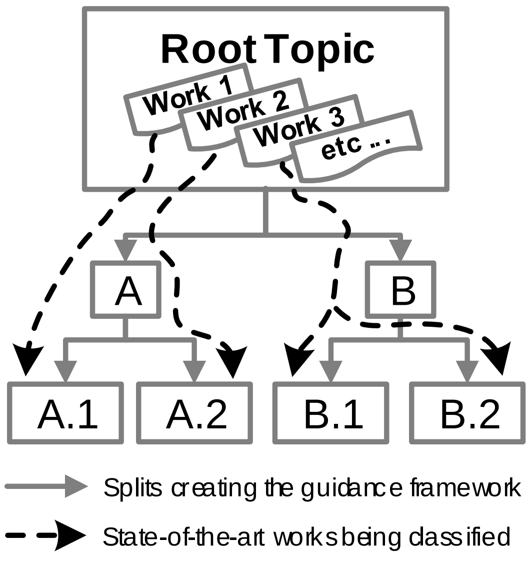
Figure 4: Binary corruptions classified as functional violations cause a parametric violation at the application level, as perceived by the SNR fidelity metric.

2.3 A Systematic Classification Methodology

The guidance framework methodology is a useful technique for the organization and evaluation of prior art in a target research domain [1] [15]. In Figure 5 we can see an illustration of this methodology, which is composed of two steps:

1. The target research topic is treated as the root of a tree. Binary splits are decomposing the initial topic to subcategories. The split at each node is initiated by a criterion that yields two *strictly complementary* children. The splitting procedure is continued until a sufficient depth of the guidance framework is reached.
2. After the guidance framework is created, state-of-the-art works are mapped to its leaves. Starting from the root of the tree, each work is evaluated against the splitting criteria and finally classified to the most suitable leaf.

Typical surveys and state-of-the-art reviews tend to exhaustively list a set of papers, which are related to the target domain. In most cases, cited work is evaluated against some criteria and trends are evaluated. On complete contrast to typical surveys, a guidance framework provides a useful tool to map existing work on orthogonal categories in a systematic way. Most importantly, hybrid and cross layer works can be thoroughly studied: Each aspect of such prior art is separately accounted for in the appropriate leaf of the guidance framework. Thus, no aspect of a hybrid work is suppressed and the classification is complete.



Root Topic
Work 1
Work 2
Work 3
etc ...
A
B
A.1
A.2
B.1
B.2
Splits creating the guidance framework
State-of-the-art works being classified

Figure 5: Illustration of the binary splits of the guidance framework and the mapping of prior art on the resulting leaves

2.4 Dependability Metrics

As we classify existing work to each of the resulting leaves, it is important to identify the figure of merit used to assess each performance variability mitigation technique. A brief guidance framework that shows the available options is presented in Figure 6. Again, we utilize the classification methodology of Subsection 2.3, in order to cover all relevant aspects of performance dependability quantification. In Table 3, we present examples of such metrics, accompanied by literature samples that utilize them in the evaluation of performance variability mitigation.

In the context of the HARPA project, it is very important to remain as in-line as possible with such industry standard metrics. That way, the technical outcome of this project can be directly comparable to existing techniques found in the literature.

We mainly split performance dependability metrics into two classes, the ones related to *functionality concerns* and the ones dealing with *parametric performance targets*. This split is based on the distinction covered earlier, in Subsection 2.2. In the former class, we further split between the issue of *metastability* and that binary *correctness*.

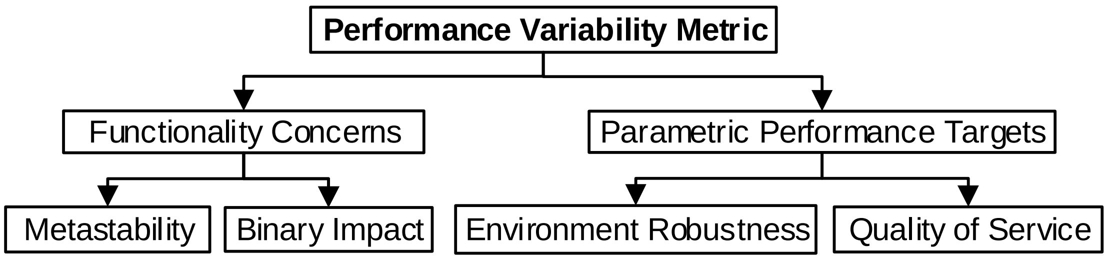


Figure 6: Classification of relevant dependability metrics

Table 3: Examples of metrics related to performance dependability

Type of Performance Dependability Metric	**Representative Example**	**Representative Prior Art Sample**
Metastability	*Static Noise Margin* (SNM): Minimum voltage swing that causes an SRAM cell to flip its content	[16]
Binary Impact	Failures in Time (FIT): The number of functional failures that arise in one billion device hours	[14]
Environment Robustness	Critical Charge (Q_{crit}): The minimum charge that can flip the content of an SRAM cell	[17]
Quality of Service	Defects per Million (DPM)	[18]
Quality of Service	Latency	[19]
Quality of Service	Signal to Noise Ratio (SNR)	[12]

2.5 Classification Overview

The prior art classification of this survey presents the research landscape of performance variability mitigation. Based on binary splits, the resulting guidance framework succeeds in covering all relevant aspects of the target topic. A complete view of our guidance framework can be seen in Figure 7. In the domain of performance variability mitigation, it is important to initially isolate the subset of the target system, where mitigation will be applied. We use the *die item* as the first splitting criterion. Parametric mitigation approaches that are restricted within a single die are distinguished from inter-die techniques. Sections 3 and 4 discuss these two branches respectively.

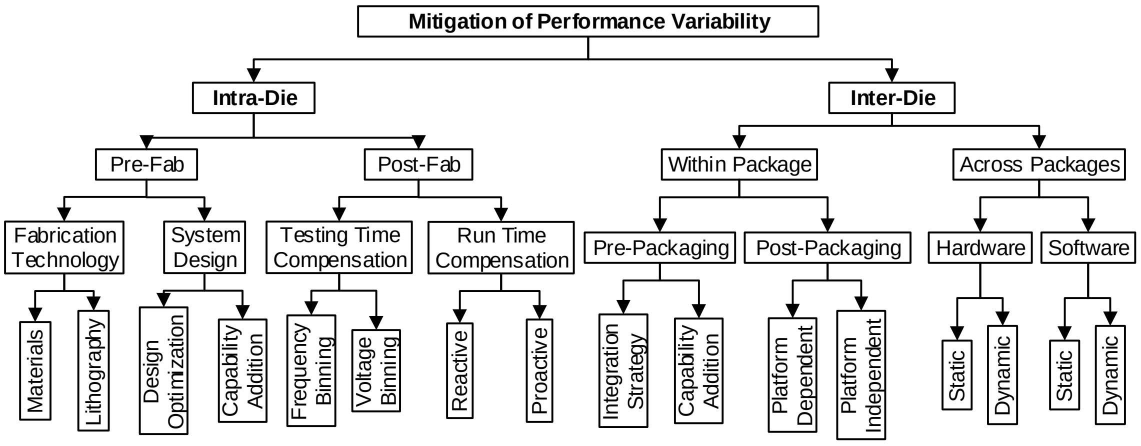


Figure 7: Complete view of the guidance framework on the mitigation of performance variability

Given that we wanted to be complete in the coverage of prior art, our classification includes sub-domains that are rather distant to the HARPA scope. For instance, material selection or advanced lithography techniques fall beyond the scope of the project. Thus they are briefly covered with a simple prior art instantiation.

3 Performance Variability Mitigation within the Die

In this branch of our classification, we explore performance variability mitigation techniques that focus on the internals of a silicon die. We further split this domain between *pre-fabrication* and *post-fabrication* of the system. Hence, the focus is mainly on process technology/design-time techniques.

3.1 Pre-Fabrication

Before the die is fabricated and packaged, we can identify performance dependability enhancements in the *fabrication technology* domain or the actual *design of the system*.

3.1.1 Fabrication Technology

In the domain of fabrication technology, we include all mitigation techniques that are employed in the fabrication area (clean room etc). These may refer to the choice of *materials* or the *lithography* techniques that are employed during process steps.

There is significant effort in choosing *materials* that optimize the system in terms of dependable performance. An important step to determine the optimal selection of materials is that of *material characterization*.

An example of this category is the set of conclusions drawn after measurements on SiGe pFETs [20]. In this work, different process parameters are explored (e.g. Ge concentration) and the consequent reliability improvement of the pFET devices is reported (e.g. acceptable operating overdrive over 10 years).

The work of Islam et al. [21] belongs to the same category: based on a set of measurements, the authors propose the co-optimization of gate leakage and Negative Bias Temperature Instability (NBTI) for various values of nitrogen concentration and effective oxide thickness.

A complementary domain refers to the *lithography* techniques that are employed. The use of alternative wavelengths and multiple patterning techniques are two of the major lithographic concerns in the deca-nanometer era.

In view of the significance of line edge roughness (LER) for correct NAND flash memory operation, Poliakov et al. explore the extent at which LER manifests itself in the case of Self-Defined Extreme Ultra-Violate (EUV) lithography [22].

The work of Joshi et al. [23] deals with the robustness of Static Random Access Memory (SRAM) cells in view of the double patterning lithography (DPL). Apart from the functional concerns, a dynamic energy reduction is achieved, based on the DPL-aware implementation.

3.1.2 System Design

In parallel with fabrication issues, significant attention is paid to the design of the actual system. Performance variability mitigation can be applied either by *optimizing the existing design* (without adding new features) or by *introducing additional capabilities* that enhance reliability.

When a design is created, it usually undergoes a series of optimizations in order to ensure its parametric and functional reliability. This happens without adding extra functionality to the system and involves either *sizing* exploration or optimization of the *placing and routing* (P&R) of the design components.

The work of Yu et al. [24] is proposing the optimization of through silicon via (TSV) placement. This optimization is deemed necessary due to the impact that TSVs have on the electrical behavior of neighboring regions [25].

In the work of Abella et al. [26] the functionality of an SRAM cell is retained, however the implementation of the cell is NAND-based rather than inverter-based. This design is reported to achieve a 50% reduction of degradation due to NBTI.

An alternative option is to add extra features to the design that will contribute to enhanced reliability. A traditional practice is to hide parametric/functional reliability violations under *redundant components* [27] [28] [29]. That way, a component that is unreliable from a parametric point of view can be masked by an identical component that operates according to parametric specifications. Another option is to enable *adaptivity* of the system, by including appropriate functional blocks. In many cases these blocks enable chip observability and controllability, i.e. they serve as *monitors* and *knobs* respectively. Here exactly, lies the relevance of the current classification sub-domain to the knobs and monitors WP of the HARPA project.

An example of this approach is the addition of monitors that capture degradation due to Negative Bias Temperature Instability [30], thus enabling Dynamic Reliability Management (DRM). This approach is further utilized to estimate the degradation of the routers found in a Network on Chip (NoC) and adjust their traffic to minimize NBTI impact [31].

3.2 Post-Fabrication

After the die has been fabricated, we can employ a series of techniques that contribute to the system's dependable performance. One option is to compensate during the *testing* of the fabricated dies. Alternatively, the parametric/functional mitigation techniques can be implemented at *run time*, namely during the entire lifetime of the silicon die.

3.2.1 Testing Time Compensation

A variety of countermeasures against reliability violations can take place during the testing of the fabricated dies. A very frequent practice, also referred to as *binning*, involves the categorization of fabricated systems, according to their parametric performance. The purpose is to identify the correct value of the die's voltage supply and/or clock frequency, within which functional specifications are met. The systems are tested under different conditions and are classified according to their performance. That way, we create an ordered set of fabricated systems and each subset can be priced/marketed accordingly. We can either bin in terms of *voltage* or *frequency*. It is very important to note that typically, the voltage is the deciding factor, given that it dictates the maximum frequency at which the die can operate [32].

In the work of Datta et al. [33] the concept of frequency binning is extensively addressed. An algorithm is proposed, which identifies the boundaries of the frequency bins, in order to maximize the yield of fabricated dies.

In the work of Zolotov et al. [34], voltage binning is correlated with leakage and switching power. As a result, apart from parametric yield and performance the results of power analysis are considered to derive optimal binning schemes.

Compensation against parametric reliability violations is partly performed at testing time in the work of Sanz et al. [35]. A testing phase is assumed, during which the speed of the available memory modules is determined, thus accounting for time-zero variability across the die.

3.2.2 Run Time Compensation

An option complementary to compensation during testing is to apply parametric countermeasures at run time. These can be either *reactive* or *proactive*. In any case we identify knobs and monitors as a major prerequisite for the application of such techniques. This observation hints towards an abundance of hybrid works that belong both in the current sub-domain and that of design time capability addition (i.e. second half of Subsubsection 3.1.2). Along with the system per se, knobs and monitors operate in a closed loop, as illustrated in Figure 8 through Figure 10. Reactive techniques involve the configuration of system knobs after a specific situation has been identified by the installed monitors. In the simple case, the mitigation utility reacts to an event and incrementally tunes the platform knobs upon detection (Figure 8). However, it is possible to equip the mitigation utility with information about platform degradation/variability gathered at design time (as illustrated in Figure 9). However, in both cases, there is no way to estimate future events or take proactive decisions. An alternative to these two solutions is to build the variability profile of the platform at run time (Figure 10). This technique is suitable for highly complex systems, where the possible variability trends are too complicated to microcode at design time. Also, this technique enables significantly credible prediction of future manifestations of performance variability. There is a variety of mathematical tools that can predict future system behavior. For example, chaos theory has been used to identify patterns in wireless workloads [36].

A representative example of the purely reactive case is the work of Zoni et al. [31]. This "sensor-wise" methodology uses monitors of NBTI degradation. According to the monitored V_{th}, virtual channel buffers are utilized differently in order to reduce NBTI impact. The case of Zoni et al. is a perfect example of a *hybrid work*: It borrows elements from both design time (see second half of Subsubsection 3.1.2 and run time parametric mitigation. More specifically, capabilities are added to the system at design time. These additional blocks operate at run time to reduce the impact of NBTI.

The complementary case refers to run time mitigation with some knowledge about performance variability trends. This information may be known either at design time (Figure 9) or built and refined at runtime (Figure 10).

A prior art sample based on design time knowledge of variability trends is featured in the work of Sanz et al. [35], where the concept of system scenarios is utilized to compensate for time zero variability of memory systems.

A runtime SRAM cache configuration system has been proposed to mitigate both time zero and time dependent device variability [37], citing the proposed technique as proactive. This work proposes the monitoring of SRAM columns based on the maximum V_{th} shift observed. A reconfiguration scheme is applied that places degraded columns into a recovery phase following a round robin schedule. However, this work does not feature sufficient elements of run time system training about platform variability trends.

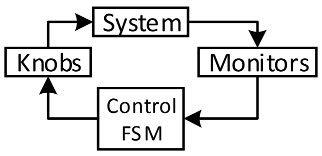


Figure 8: Typical reactive mitigation scheme

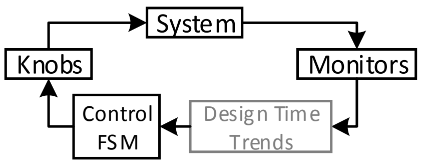


Figure 9: Reactive mitigation of performance variability, based on design-time system characterization

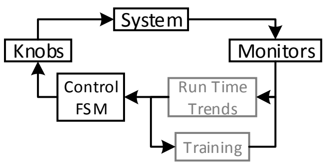


Figure 10: Proactive mitigation of performance variability, by building system operation trends at run time

Run time training of a platform, about system performance trends is featured in the work of Munaga et al. [38], where a confident estimation of platform workload is derived, in order to select the optimal operation mode, thus achieved significant energy savings. However, this prior art element is in no way associated with performance variability due to reliability violations intrinsic to the system (e.g. related to Si phenomena). However, it presents a very interesting approach that can be reused if appropriate variability trend prediction can be achieved.

4 Performance Variability Mitigation across Dies

In this Section, we present parametric mitigation techniques that are applied on a set of silicon dies. We can split between two complementary domains: In the first case, dies are combined into a *single package*. In the second case, the system is composed of discrete packaged components[1].

4.1 Within Package

In this case, we assume that a set of dies is to be packaged into a single discrete component. As a result, we can split the integration procedure between *pre-packaging* and *post-packaging* techniques. That way, the greater area of multi-die integration [39] is fully covered in these two sub-domains.

4.1.1 Pre-Packaging

Before a set of dies is combined into a single package, parametric mitigation can be implemented either through a specific *integration strategy* or by *adding extra capabilities* to the System-in-Package (SiP) [40].

In the work of Garg et al. [41] we see a specific frequency binning strategy that optimizes the yield of three dimensional (3D) SiPs. “Due to process variations, each die actually will have a different maximum operating frequency”. Based on this observation, the authors propose an optimization of the frequency binning process in order to optimize the 3D SiP yield.

The addition of extra capabilities (apart from the functionality of each integrated die) is illustrated in the work of Sabry et al. [42]. The two main additional features added in the 3D SiP involve liquid cooling channels and a fuzzy controller that manages the liquid flow rate and employs Dynamic Voltage and Frequency Scaling (DVFS) in order to improve the thermal profile of the SiP.

4.1.2 Post-Packaging

After the dies have been packaged into a single system, parametric reliability violations can be mitigated using either *platform dependent* or *platform independent* techniques.

The work of Coskun et al. [43] is a very good example of platform dependent techniques. The methodology presented in this work makes use of the built-in thermal sensors of the SiP and performs prediction of future temperature conditions using autoregressive moving average modeling.

A good example of platform independent mitigation of performance variability across packaged silicon dies is that of Chipkill, as developed by IBM [44]. Without intervening with the DRAM memory chips,

[1] We assume at least one packaged component. Each package may contain at least one die, hence die-to-die arrangements are also included.

this technique reroutes the bits of each ECC word, so that “no single ECC word will experience more than one bit of bad data”. However, this mitigation technique is rather static in nature.

4.2 Across Packages

This domain is mostly correlated to HPC, since such systems are typically composed of many processing elements, each being a discrete packaged component [45] [46]. At this point, it is very important to highlight the context-specific meaning of the *performance variability* term, especially in the case of the HPC domain. This term has been used for more than 30 years [47] to illustrate the variability in the responsiveness of a computing system, as a result of varying resource availability and user demands [9]. This term is very suitable to the domain of parametric reliability across system packages, since this is typically the arrangement of High Performance Computing (HPC) infrastructure. In view of performance variability, mitigation techniques exist either at the *hardware* or *software* levels.

4.2.1 Hardware Mitigation

When parametric violations need to be mitigated in hardware across a variety of packaged systems, *static* or *dynamic* countermeasures are proposed.

The work of Krammer et al. [48] presents a very complete picture of performance variability in highly parallel systems (composed of many processing elements, each one instantiated as a single package). According to this paper “making [processing] nodes strictly homogeneous in hardware is key”. However, no proposal of additional static hardware modules is proposed. Similarly, the application of Chipkill [44] also falls under this category, assuming that each DRAM chip is uniquely packaged.

We can identify a scarcity of dynamic hardware mitigation techniques in the mitigation of performance variability across system packages. This observation invites meaningful contribution, which is very suitable to the scope of the knobs and monitors WP of the HARPA project (WP3).

4.2.2 Software Mitigation

Given the scale of an HPC system, it is very attractive to mitigate performance variability fluctuations of such systems at the software level. Again, we distinguish between *static* and *dynamic* techniques.

The work of Kramer et al. [48] also contributes with a static software mitigation technique against performance variability. In this paper, it is recognized that the performance of processing nodes can be affected by “system tasks that run periodically”. As a result, it is proposed to enforce a system-wide “heartbeat” that dictates when “housekeeping tasks on all nodes” will be performed.

This is another example of a *hybrid paper*, since it is also instantiated in the hardware mitigation leaf of our classification (for across-packaging mitigation case). It also suggests the existence of software monitors that will sample the state of the HPC system in a periodic way.

Static software mitigation of performance variability is also illustrated in the work of Petrini et al. [49]. In this presentation “sources of noise” are identified, which cause performance fluctuations. Such sources include daemons that run on some of the processing nodes. Some of them are removed and others are moved to other processing nodes, in order to mitigate the performance variability of the HPC system under examination.

A dynamic software approach is proposed by Lofstead et al. [10]. The authors propose a software supported adaptive input/output (I/O) approach in order to mitigate the performance variability of petascale storage systems. The proposed approach is implemented with the ADIOS middleware [50].

Immediately, we can distinguish that symmetrical software and hardware installations are typically preferred in the context of the above systems. However, the degree of hardware distribution for monitoring/controlling of the system is somewhat reduced.

5 Discussion & Conclusions

In the current report, we have performance a top-down exploration of literature on performance dependability. For that purpose, we initially presented fundamental concepts related to the target research field. Then, we proceeded with the prior art classification. For that purpose, we utilized a guidance framework based on binary splits, to organize the target research domain in an orthogonal fashion. The resulting framework is a binary tree, the leaves of which represent different approaches towards the mitigation of performance variability.

In this report, we have populated this framework with prior art samples, giving emphasis to the sub-domains that are closer to Work Packages of the HARPA project. From this literature review, we can conclude to the following observations:

1. When dealing with performance variability, it is important to consider function and parametric reliability violations alike. In Subsection 2.2, we have illustrated the interplay between these two types of dependability violations and how *both* may contribute to performance variability.

2. Starting from a breadth-first of prior art on performance variability mitigation, we can conclude that the Work Packages of the HARPA project are *evenly distributed* across relevant areas. Excluding the domains of process technology and circuit design, the development of the knobs and monitors, of the HARPA Run Time Engine and the respective Operating System, do cover a diverse set of possible approaches towards performance dependability.

3. State-of-the-art works feature reduced emphasis on proactive mitigation (see Subsubsection 3.2.2). The prior art samples that we managed to isolate tend to place reduced emphasis on training the target platform based on variability/degradation trends. Such dynamic variability control has been implemented in the context of wireless applications, however it did not account for performance variability due to physically induced mechanisms. This observation invites the development of proactive mitigation schemes that should aim towards dependable performance. Milestones of the HARPA project are already aligned towards this direction.

4. The many-core/HPC typically focuses on resource symmetry, in order to guarantee predictable performance. Assuming the diversity of variability sources in this context (processors, storage, utilization, etc.), it is important to strategically distribute novel knobs and monitors that will enable proper mitigation of performance variability. The responsiveness of such utilities should be faster than traditional bookkeeping/maintenance software, hence novel hardware knobs and monitors and clearly motivated.

6 Works Cited

[1] D. Rodopoulos, et al., "Classification Framework for Analysis & Modeling of Physically Induced Reliability Violations," *ACM Computing Surveys (under Major Revision)*, 2012.

[2] A. Avizienis, J. .-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *Dependable and Secure Computing, IEEE Transactions on*, vol. 1, no. 1, pp. 11-33, Jan. 2004.

[3] A. V. Yakovlev, A. M. Koelmans, and L. Lavagno, "High-level modeling and design of asynchronous interface logic," *Design Test of Computers, IEEE*, vol. 12, no. 1, pp. 32-40, 1995.

[4] "3GPP TSG-RAN, Physical Channels and Modulation, v8.1.0 (2007-11)".

[5] "IEEE 802.11n-2009 Standard Document, Amendment 5, 29 Oct 2009".

[6] M. Toledano-Luque, et al., "Response of a single trap to AC negative Bias Temperature stress," in *Reliability Physics Symposium (IRPS), 2011 IEEE International*, Monterey, CA, 2011, pp. 4A21-4A28.

[7] J. V. Mancaa, et al., "Localized monitoring of electromigration with early resistance change measurements," *Microelectronics Reliability*, vol. 38, no. 4, pp. 641-650, Apr. 1998.

[8] M. Zapater, et al., "Leakage and temperature aware server control for improving energy efficiency in data centers," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013* , Grenoble, 2013, pp. 266-269.

[9] A. Iosup, N. Yigitbasi, and D. Epema, "On the Performance Variability of Production Cloud Services," in *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, 2011, pp. 104-113.

[10] J. Lofstead, et al., "Managing Variability in the IO Performance of Petascale Storage Systems," in *High Performance Computing, Networking, Storage and Analysis (SC), 2010 International Conference for* , New Orleans, LA, 2010, pp. 1-12.

[11] X. Li and D. Yeung, "Application-Level Correctness and its Impact on Fault Tolerance," in *High Performance Computer Architecture, 2007. HPCA 2007. IEEE 13th International Symposium on* , Scottsdale, AZ, 2007.

[12] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Prentice Hall, 2007.

[13] D. D. Gajski and R. H. Kuhn, "New VLSI Tools," *Computer*, vol. 16, no. 12, 1983.

[14] S. Mukherjee, *Architecture Design for Soft Errors*. Morgan Kaufmann, 2008.

[15] A. Kritikakou, F. Catthoor, V. Kelefouras, and C. Goutis, "A Systematic Approach to Classify Design-Time Global Scheduling Techniques," *ACM Computing Surveys*, vol. 45, no. 2, 2013.

[16] P. Upadhyay, S. K. Chhotray, R. Kar, D. Mandal, and S. P. Ghoshal, "Analysis of static noise margin and power dissipation of a proposed low voltage swing 8T SRAM cell," in *Information & Communication Technologies (ICT), 2013 IEEE Conference on* , JeJu Island, 2013, pp. 316-320.

[17] R. Naseer, Y. Boulghassoul, J. Draper, S. DasGupta, and A. Witulski, "Critical Charge Characterization for Soft Error Rate Modeling in 90nm SRAM," in *Circuits and Systems, 2007.*

ISCAS 2007. IEEE International Symposium on , New Orleans, LA, 2007, pp. 1879-1882.

[18] S. Mondal, X. Yin, J. Muppala, J. Alonso, and K. Trivedi, "Defects Per Million (DPM) Computation in Service-Oriented Environments ," *Services Computing, IEEE Transactions on*, Nov. 2013.

[19] Z. Wan, "Sub-millisecond level latency sensitive Cloud Computing infrastructure," in *Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2010 International Congress on* , Moscow, 2010, pp. 1194-1197.

[20] J. Franco, et al., "Improvements of NBTI reliability in SiGe p-FETs," in *Reliability Physics Symposium (IRPS), 2010 IEEE International*, Anaheim, CA, 2010.

[21] A. E. Islam, G. Gupta, K. Z. Ahmed, S. Mahapatra, and M. A. Alam, "Optimization of Gate Leakage and NBTI for Plasma-Nitrided Gate Oxides by Numerical and Analytical Models," *Electron Devices, IEEE Transactions on* , vol. 55, no. 5, pp. 1143-1152, Apr. 2008.

[22] P. Poliakov, et al., "Spacer-Defined EUV Lithography Reducing Variability of 12nm NAND Flash Memories," in *Memory Workshop (IMW), 2012 4th IEEE International* , Millan, 2012, pp. 1-4.

[23] V. Joshi, K. Agarwal, D. Blaauw, and D. Sylvester, "Analysis and optimization of SRAM robustness for double patterning lithography," in *Computer-Aided Design (ICCAD), 2010 IEEE/ACM International Conference on* , San Jose, CA, 2010, pp. 25-31.

[24] L. Yu, et al., "Methodology for analysis of TSV stress induced transistor variation and circuit performance," in *Quality Electronic Design (ISQED), 2012 13th International Symposium on* , Santa Clara, CA , 2012, pp. 216-222.

[25] A. Mercha, et al., "Impact of thinning and through silicon via proximity on High-k / Metal Gate first CMOS performance," in *VLSI Technology (VLSIT), 2010 Symposium on* , Honolulu, 2010, pp. 109-110.

[26] J. Abella, X. Vera, O. Unsal, and A. Gonzalez, "Nbti-resilient memory cells with nand gates," U.S. Application US20080084732 A1, Apr. 10, 2008.

[27] Hewlett-Packard Development Company L.P., "Avoiding Server Downtime from Hardware Errors in Sytem Memory with HP Memory Quarantine," Technology Brief TC1112809, 2012.

[28] Hewlett-Packard Development Company L.P., "How Memory RAS Technologies can Enhance the Uptime of HP ProLiant Servers," White Paper 4AA4-3490ENW, 2013.

[29] F. A. Bower, P. G. Shealy, S. Ozev, and D. J. Sorin, "Tolerating Hard Faults in Microprocessor Array Structures," in *International Conference on Dependable Systems and Networks (DSN)*, Florence, 2004.

[30] P. Singh, E. Karl, D. Sylvester, and D. Blaauw, "Dynamic NBTI management using a 45nm multi-degradation sensor," in *Custom Integrated Circuits Conference (CICC), 2010 IEEE* , San Jose, CA, 2010, pp. 1-4.

[31] D. Zoni and W. Fonraciari, "Sensor-wise methodology to face NBTI stress of NoC buffers," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013* , Grenoble, France , 2013, pp. 1038-1043.

[32] Intel Corporation, "Using the RCCE Power Management Calls," revision 1.1, 2011.

[33] A. Datta, S. Bhunia, J. H. Choi, S. Mukhopadhyay, and K. Roy, "Speed binning aware design methodology to improve profit under parameter variations," in *Design Automation, 2006. Asia and*

South Pacific Conference on, Yokohama, 2006.

[34] V. Zolotov, C. Visweswariah, and J. Xiong, "Voltage binning under process variation," in *ICCAD '09 Proceedings of the 2009 International Conference on Computer-Aided Design* , 2009, pp. 425-432.

[35] C. Sanz, et al., "Combining system scenarios and configurable memories to tolerate unpredictability," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 13, no. 3, Jul. 2008.

[36] N. Zompakis, V. Tsoutsouras, A. Bartzas, D. Soudris, and G. Pavlos, "Dynamic Frequency Scaling for MPSoCs based on Chaotic Workload Analysis," in *Architecture of Computing Systems (ARCS), 2010 23rd International Conference on*, Hannover, Germany, 2010, pp. 1-8.

[37] P. Pouyan, E. Amat, F. Moll, and A. Rubio, "Design and implementation of an adaptive proactive reconfiguration technique for SRAM caches," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013*, Grenoble, France, 2013, pp. 1303-1306.

[38] S. Munaga, "METHOD AND SYSTEM FOR OPERATING IN HARD REAL TIME," Belgian Patent US 2010/0191349 A1, Jan. 22, 2010.

[39] V. F. Pavlidis and E. G. Friedman, *Three-dimensional Integrated Circuit Design (Systems on Silicon)*, 1st ed. Morgan Kaufman, 2008.

[40] I. O'Connor, B. Courtois, K. Chakrabarty, and M. Hampton, "Heterogeneous Systems on Chip and Systems in Package," in *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE '07*, Nice, France, 2007, pp. 1-6.

[41] S. Garg and D. Marculescu, "System-level process variability analysis and mitigation for 3D MPSoCs," in *Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE '09*, Nice, France, 2009, pp. 604-609.

[42] M. M. Sabry, A. K. Coskun, D. Atienza, T. S. Rosing, and T. Brunschwiler, "Energy-Efficient Multiobjective Thermal Control for Liquid-Cooled 3-D Stacked Architectures," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 30, no. 12, pp. 1883-1896, Dec. 2011.

[43] A. K. Coskun, T. S. Rosing, and K. C. Gross, "Proactive temperature balancing for low cost thermal management in MPSoCs," in *Computer-Aided Design, 2008. ICCAD 2008. IEEE/ACM International Conference on*, San Jose, CA, 2008, pp. 250-257.

[44] IBM, "IBM Chipkill Memory," International Business Machines Corporation, 1999.

[45] S. H. Fuller and L. I. Millett, "Computing Performance: Game Over or Next Level?," *Computer*, vol. 44, no. 1, pp. 31-38, Jan. 2011.

[46] (2014, Apr.) TOP500 Supercomputer Sites. [Online]. http://www.top500.org/lists/2013/06/

[47] T. E. Bell, "Computer performance variability," in *AFIPS National Computer Conference and Exposition*, 1974.

[48] W. T. C. Kramer and C. Ryan, "Performance variability of highly parallel architectures," in *International conference on Computational science: PartIII*, 2003, pp. 560-569.

[49] F. Petrini, D. J. Kerbyson, and S. Pakin, "The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q," in *SC '03 Proceedings of*

the 2003 ACM/IEEE conference on Supercomputing, 2003.
[50] Oak Ridge National Laboratory, "ADIOS 1.5.0 User's Manual," ORNL/TM-2009/10, 2013.