Classification Framework for Analysis and Modeling of Physically Induced Reliability Violations

DIMITRIOS RODOPOULOS, MicroLab, ECE, NTUA
GEORGIA PSYCHOU, RWTH Aachen University
MOHAMED M. SABRY, ESL, EPFL
FRANCKY CATTHOOR, imec and KU Leuven
ANTONIS PAPANIKOLAOU and DIMITRIOS SOUDRIS, MicroLab, ECE, NTUA
TOBIAS G. NOLL, RWTH Aachen University
DAVID ATIENZA, ESL, EPFL

Technology downscaling is expected to amplify a variety of reliability concerns in future digital systems. A good understanding of reliability threats is crucial for the creation of efficient mitigation techniques. This survey performs a systematic classification of the state of the art on the analysis and modeling of such threats, which are caused by physical mechanisms to digital systems. The purpose of this article is to provide a classification tool that can aid with the navigation across the entire landscape of reliability analysis and modeling. A classification framework is constructed in a top-down fashion from complementary categories, each one addressing an approach on reliability analysis and modeling. In comparison to other classifications, the proposed methodology approaches the target research domain in a complete way, without suppressing hybrid works that fall under multiple categories. To substantiate the usability of the classification framework, representative works from the state of the art are mapped to each appropriate category and are briefly analyzed. Thus, research trends and opportunities for novel approaches can be identified.

Categories and Subject Descriptors: B.8 [**Hardware**]: Performance and Reliability—*Reliability, testing, and fault-tolerance*; C.4 [**Computer Systems Organization**]: Performance of Systems—*Reliability, availability, and serviceability*

General Terms: Design, Performance, Reliability

Additional Key Words and Phrases: Reliability analysis, error, failure, fault, classification framework

ACM Reference Format:
Dimitrios Rodopoulos, Georgia Psychou, Mohamed M. Sabry, Francky Catthoor, Antonis Papanikolaou, Dimitrios Soudris, Tobias G. Noll, and David Atienza. 2015. Classification framework for analysis and modeling of physically induced reliability violations. ACM Comput. Surv. 47, 3, Article 38 (February 2015), 33 pages.
DOI: http://dx.doi.org/10.1145/2678276

Dimitrios Rodopoulos, Antonis Papanikolaou, Francky Catthoor, and Dimitrios Soudris were partial support by EU-FP7-612069-HARPA; Georgia Psychou and Tobias G. Noll were support by EU ARTEMIS Joint Undertaking under grant agreement n° 621429 (project EMC^2) and by the Dutch National programmes/Funding Authorities; Mohamed M. Sabry and David Atienza were partial support by the ONR-G (grant no. N62909-14-1-N072), and the ObeSense (no. 20NA21_143081) RTD project evaluated by the Swiss NSF and funded by Nano-Tera.ch with Swiss Confederation financing.
Authors' addresses: D. Rodopoulos, A. Papanikolaou, and D. Soudris, ECE-NTUA, 9 H. Polytechneiou Str., 157 80 Athens, Greece; G. Psychou and T. G. Noll, EECS–RWTH Aachen University, Schinkelstr. 2, 52062, Aachen, Germany; M. M. Sabry and D. Atienza, EPFL-STI-IEL-ESL, Station 11, 1015 Lausanne, Switzerland; and F. Catthoor, imec, Kapeldreef 75, 3001 Leuven, Belgium; contact email: drodo@microlab.ntua.gr.

DOI: http://dx.doi.org/10.1145/2678276

1. INTRODUCTION

Reliability threats in the hardware (HW) of digital systems are expected to intensify with the reduction of device dimensions [McPherson 2006]. Aggressive energy reduction threatens the performance [Borkar et al. 2011] and binary correctness [Kumar et al. 2009] of modern digital systems. As a result, low-energy designs should be enhanced with appropriate reliability solutions. In the past, static mitigation techniques have been developed [Sorin 2009]. As the power and area constraints become increasingly strict, exclusive use of these techniques introduces infeasible overheads. It is highly desirable to explore near-optimal mitigation solutions, which alleviate reliability threats while not violating the power and area restrictions of the design [Cho et al. 2012]. The creation of such solutions requires a comprehensive understanding of the physically induced violations that need to be mitigated. To create appropriate mitigation techniques, we need to understand *how* these effects propagate within a system. Furthermore, we need to quantify *what* is the effect of this propagation in relation to the system's operating conditions and workload.

Reliability analysis and modeling are the answers to these two questions. The purpose of this survey is to provide a complete and systematic categorization methodology that addresses all approaches in this domain. This categorization allows identification of regions that have received varying degrees of attention by the research community. The interaction and interdependencies between reliability analysis and reliability modeling is also illustrated. Research trends can be identified and guidance is provided for novel approaches. Judging from the plethora of reliability aspects addressed in the literature, we need to set the content boundaries of the current survey. Hence, the following areas are beyond our scope: (1) reliability issues due to malicious attacks [Avizienis et al. 2004]; (2) software (SW) quality/interaction [Walia et al. 2006; Tyagi et al. 2011]; (3) nonelectronic threats [Koganemaru et al. 2008; Peng et al. 2009]; (4) power systems engineering or power grid issues (e.g., [Yokoyama et al. 2008]); (5) derivation of novel reliability metrics and enumeration of the existing ones; (6) classification based on the duration and repeatability of reliability violations; and (7) the actual mitigation solutions handling reliability threats. An exhaustive listing of possible reliability violations is also out of scope.

We focus on the analysis and modeling of malfunctions that are caused by specific physical mechanisms in the system's HW. Analysis techniques track the propagation of these mechanisms' effects to higher abstraction levels. Modeling formulates correlations between the degree of propagation and combinations of operating conditions and workload. It should be noted that papers presenting mitigation approaches usually come with an evaluation of the latter, in the form of a reliability analysis experiment. Such papers still provide a good source of related analysis techniques.

The contributions of this article are as follows. We build a broad classification framework on reliability analysis and modeling. It is composed of complementary categories, where state-of-the-art approaches are mapped based on their relevance. Works belonging to multiple categories (i.e., hybrids) are inspected, with each of their aspects analyzed in the appropriate category. Thus, we create a complete and consistent classification of approaches toward reliability analysis and modeling. We also provide a consistent set of definitions for terms frequently used across the text.

This survey is organized as follows. Section 2 provides key terminology, related classification work, and an overview of our methodology. Section 3 presents reusable concepts across our classification and an overview of the latter. Reliability analysis is presented in Section 4. Section 5 deals with reliability modeling. The extensibility of our classification is illustrated in Section 6. Conclusions are summarized in Section 7.

2. THEORETICAL BACKGROUND

2.1. Terminology

In this section, we define some terms that will be extensively used to avoid any ambiguity in later parts of the text. We also strive for consistency with the terminology that has been used up to this point in the literature. It is important to note that the definitions presented section are outlined in accordance to the classification framework methodology of Section 2.3—namely, they create complementary pairs. This allows the creation of reusable classification frameworks (Section 3.1), which will alleviate the complexity of our classification.

2.1.1. Digital System Components and Partitioning.

Definition 2.1. *Hardware (HW)* is a set of permanent components of a digital system. To emphasize on the availability of a manufactured sample, we use the term *pure HW*. When we are unable to obtain pure HW, we need to resort to a *HW description*.

The Gajski-Kuhn Y-chart [Gajski et al. 1983] presents the possible representations and abstraction levels where a HW description may reside.

Definition 2.2. *Software (SW)* refers to the set of coded features that are executed by a digital system. It must not be confused with a HW description, even if the latter may be available in the form of an algorithmic description.

The SW of a digital system may be specific to a single service among the ones delivered by this system (e.g., device driver). Generic SW utilities may be available, which are not specific to a single provided service (e.g., scheduler of the operating system).

Definition 2.3. *Operating conditions* represent the interference caused to a digital system by its environment. This interference may originate from neighboring systems (e.g., electromagnetic interference (EMI)) or from natural stimuli (e.g., cosmic particles).

Hence, to completely define a digital system, we need to describe its:

(1) HW, either as pure HW or an equivalent HW description
(2) SW, with parts specific to a single service and general-purpose components
(3) Operating conditions: natural stimuli and interference with neighboring systems.

Definition 2.4. A *black box* is a subset of the digital system's HW, the internals of which cannot be observed. Hence, its component connection scheme is unavailable.

A black box may refer either to pure HW or a HW description. To define a black box, we need to describe its boundaries and the respective abstraction level. For a reliability assessment of a digital system, it is important to identify its black boxes and assess their observability and controllability through a series of input/output (I/O) ports.

2.1.2. Reliability Assessment.

Definition 2.5. *Reliability violation* is an umbrella term describing any deviation of a digital system from its correct and expected operation. The term is used regardless of the deviation's severity, and hence it covers minor through fundamental malfunctions. In the context of this survey, we only assume physically induced reliability violations.

Definition 2.6. The quantity that represents the susceptibility of a digital system to reliability violations is called a *reliability metric*.

Exhaustive enumeration of reliability metrics is beyond the scope of this survey.

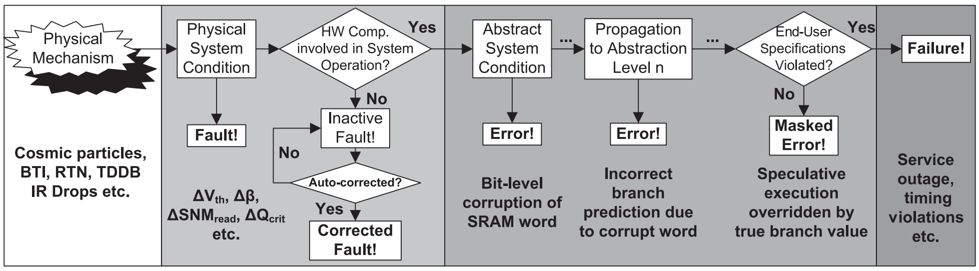

Physical Mechanism
Cosmic particles, BTI, RTN, TDDB IR Drops etc.
Physical System Condition
HW Comp. involved in System Operation?
Yes
No
Fault!
Inactive Fault!
Auto-corrected?
No
Yes
Corrected Fault!
ΔV_{th}, $\Delta\beta$, ΔSNM_{read}, ΔQ_{crit} etc.
Abstract System Condition
...
Propagation to Abstraction Level n
...
End-User Specifications Violated?
Yes
No
Failure!
Error!
Error!
Masked Error!
Bit-level corruption of SRAM word
Incorrect branch prediction due to corrupt word
Speculative execution overridden by true branch value
Service outage, timing violations etc.

Fig. 1. The chain of events from a physical mechanism to a user-perceived failure.

Definition 2.7. The term *reliability concern* will be used as a generic term to refer collectively to both reliability violations and reliability metrics.

If we inspect the reliability violations and metrics that are present in the literature, we can make a distinction based on the nature of these reliability concerns.

Definition 2.8. *Functional* reliability concerns deal with the correctness of the binary digits that are stored, processed, or communicated within and across systems.

A functional reliability violation is the *soft error* [Mukherjee 2008]—that is, the binary corruption of a node (e.g., of the bit in a static random access memory (SRAM) cell). As for reliability metrics, a functional example is the bit error rate (BER), describing the percentage of bits that may be erroneous within a stream of data.

Definition 2.9. A *parametric* reliability concern refers either to operation margins of the digital system or a black box within the former. These may be related to the timing or quality cost (e.g., total energy) of that system or black box.

A good example of parametric reliability violations is the fluctuation of a pMOSFET's threshold voltage (V_{th}) due to negative bias temperature instability (NBTI) [Grasser et al. 2011]. From a reliability metric viewpoint, a good example is the static-noise margin (SNM), which illustrates the voltage margin of an SRAM cell, beyond which the cell becomes unstable [Seevinck et al. 1987] (i.e., prone to "bit flips").

Next, we define three key terms that connect the chain of events from a physical phenomenon to a malfunction that is perceived by the end user. A schematic presentation of these terms is given in Figure 1. The following three terms are reliability violations (see Definition 2.5) and are presented in descending order of severity.

Definition 2.10. A *failure* is an undesired behavior or result delivered by a digital system. It is perceived by the end user and violates the system's specifications.

The end user may be any party that cooperates with the system under test in a sequential arrangement. We will assume that the service provided by the system is specified in a complete way by the constraints of its end user. As a result, failures due to inadequate system specifications are out of scope [Avizienis et al. 2004].

Definition 2.11. An *error* is an abstract condition "that may lead to a failure" [Avizienis et al. 2004]. The failure manifestation is a result of the error propagating across abstraction levels of a system. In case no failure arises, the error is *masked*.

The abstract nature of the system condition contains no information about the underlying physics of the error. Examples of such conditions are corrupt binary values or an unspecified state in the finite state machine (FSM) of the system.

Definition 2.12. Based on Avizienis et al. [2004], a *fault* is defined as the "adjudged or hypothesized" physical condition of a system and is the cause of an error.

Unlike errors and failures, the fault contains information about the underlying physics of a malfunction. It is restricted to a small region of the digital system (e.g., a handful of devices), whereas the error involves a larger subset. Examples of faults are shifts in the threshold voltage of a transistor (ΔV_{th}) or a reduced critical charge (Q_{crit}) [Roche et al. 1999] of an SRAM cell. A fault remains inactive until the respective HW component gets involved in system operation. Once the fault is activated, an error is triggered. A fault may also be naturally corrected before it is activated.

Definition 2.13. *Reliability analysis* of a digital system is the monitoring of the system's performance under the presence of reliability violations.

Reliability violations are *injected* in the system either naturally, through the operating conditions, or artificially by manipulating the ported SW or regions of the HW. The choice between fault or error injection depends on the complexity of the injection procedure and the target abstraction level. Finally, the direct injection of failures in the digital system would be less useful, because the user-specified constraints are violated on injection. Monitoring of the system during reliability analysis may address functional and parametric reliability violations with various degrees of severity. Furthermore, the propagation of a reliability violation to higher abstraction levels is also *detected*. With reliability analysis, we can estimate the portion of the injected faults that become activated (i.e., become errors) within a digital system. More details about the possible injection and detection techniques are presented in Section 4.

Definition 2.14. *Reliability modeling* involves the derivation of a formal expression for a reliability metric of a well-defined digital system (see Definitions 2.1 through 2.3).

During a reliability analysis phase, we monitor the occurrence of errors or failures. These measurements yield correlations between reliability metrics and the usage profile (i.e., models). Models of reliability metrics can be used for analysis at higher abstraction levels, usually with a HW description. We present the interaction between analysis and modeling in Section 3.2. Modeling options are presented Section 5.

2.2. Related Classification Work

By inspecting classification and survey papers related to our target domain, it is evident that reliability analysis and modeling are of major importance to the research community and industry. Two classification trends are observed, which reside in the vicinity of the reliability analysis and modeling domains. Through the evaluation of these works, the optimality and novelty of our classification can be substantiated.

2.2.1. Term Classifications. Papers that are dedicated exclusively to terminology are quite rare. In many cases, we see a bank of definitions that accompanies a novel approach on reliability analysis and modeling. Such classifications are important, as they create a consistent and reusable terminology for the target research domain.

A paper in the area of reliability and dependability, mainly dedicated to terminology, is Avizienis et al. [2004]. It classifies erroneous states and connects them through propagation or causality relations. It maintains a system-level point of view toward dependability and security. This makes it difficult to create connections between the defined terms and physical phenomena that threaten the reliability of digital systems. We must also note the addition of the security aspect—namely, malicious attacks to a digital system. This is a valid concern for the dependability of digital systems but is not part of our classification scope, because it is unrelated to the physical aspects as such.

However, the security aspect of system dependability is complementary to our goals. Overall, Avizienis et al. [2004] is a major source of terminology and illustrates digital systems' dependability with consistent high-level terms. We have stayed as close as possible to those in the projection for our objectives.

A set of definitions related to bit-level corruptions induced by radiation, referred to as soft errors, can be found in the work of Mukherjee et al. [2005]. In addition, a timeline is presented for the possible outcomes of a bit-level corruption in a microprocessor, where detection and mitigation are considered as well. The terms that are introduced throughout this article cover abstraction levels from a single device (e.g., device error rate) and range up to the architecture (e.g., propagation vulnerability factor (PVF)). The scope of the paper includes only reliability threats from radiation sources. Furthermore, formalized expressions are presented for the derivation of reliability metrics like the architecture vulnerability factor (AVF) of a HW structure.

Papers that present a novel approach or technical reports that focus on a specific application are always a valid source of term classifications. Typical examples of that nature are *safety standards*, which contain dedicated chapters where terms related to reliability analysis or modeling are defined. For example, in an introduction to the IEC 61508 standard [Redmill 2005], a quote from the standard reads that "either qualitative or quantitative hazard and risk analysis techniques may be used." In Redmill [2005], we also see definitions used in the standard, describing various degrees of the likelihood or consequence of hazard occurrence. An entire part of the standard is apparently dedicated to related definitions [Bell 2006]. A similar approach toward term classification is reported for the ARP 4761 aviation standard [SAE International 1996] and the ISO 26262 automotive standard [ISO 26262-1 2011]. Finally, we need to note that the definition of many reliability concepts dates back to very old sources, as in the case of the "bathtub curve' dating back to 1693 [Klutke et al. 2003]. Apparently, a wide range of reference sources define and formulate traditional concepts related to reliability [Ebeling 2009; Kapurl et al. 1977].

2.2.2. Review of Classifications of Existing Techniques. Another set of related work categorizes reliability analysis and modeling techniques. The injection of faults for the assessment of digital systems reliability has yielded significant survey work. The concept of fault injection (see Definition 2.13) involves the creation of disturbance in a digital system (prototype or HW description) to assess the latter's reliability.

In Hsueh et al. [1997], we find a presentation of fault injection techniques. The basic distinction is between fault injection performed in the HW (either with or without a physical contact) and in the SW (either at compilation time or at runtime). Each type of fault injection is assessed, among other factors, in terms of cost, repeatability, and controllability. State-of-the-art tools are provided for each fault injection category. However, faults and errors are not defined in a systematic way, thus reducing the generality of the claims made in this work. In addition, no information is provided regarding the postprocessing of the data that a fault injection campaign yields. As a result, poor connection exists between the reliability analysis and modeling phases.

Ziade et al. [2004] additionally introduces the simulation- and emulation-based fault injection techniques in a more systematic way. In Benso et al. [2010], we find a classification of fault injection techniques based on the HW versus SW distinction. The readouts of the injection tools are also presented. In Arlat et al. [2003], we can see four different injection techniques that have been used for the reliability assessment of a specific fault-tolerant architecture [Kopetz et al. 1989].

The creation of formal reliability models has also received significant attention. Geist et al. [1990] evaluate techniques used for the reliability estimation of fault-tolerant systems. This survey initially describes the generations of reliability models from the

N-modular redundancy model up to Markovian methods and detailed modeling of the detection, isolation, and recovery processes. Five state-of-the-art tools for reliability analysis are presented and evaluated based on the utilized fault and recovery models. The required platform support and output metrics of each tool are also discussed. In Tomek et al. [1994], we find a classification of reliability modeling approaches motivated by a representative test case. The aspect of model calibration gives rise to the issue of fault injection, either to a prototype or a HW description.

Finally, Lyu [2007] presents a roadmap of SW reliability. This classification work presents reliability analysis and modeling approaches from a purely SW point of view. Research trends and future challenges are also discussed. There is a correlation between SW and HW reliability, but it is merely conceptual. This roadmap remains detached from the propagation of fault effects from the HW up to the system SW and end-user abstraction levels.

2.2.3. Assessment of Related Classifications. The existence of many related classifications indicates the apparent need for a systematic survey of reliability analysis and modeling. Some terminology categorizations do not classify the underlying physical mechanisms in a systematic way. Others are application specific. As for technique classifications, the injection of faults for verification purposes is a topic that has received extensive attention by the research community. However, these works feature reduced focus on the modeling phase that usually follows an injection campaign. We have also witnessed technique classifications that are restricted to a specific platform. Papers focusing on reliability models present a thorough formulation of the mathematical principles. However, there is no classification of the possible reliability violations (i.e., the physical threats to system reliability). The same observation holds for SW reliability analysis classifications, where the connection to the HW is scarcely existent.

In view of the related classification work, our approach differentiates itself by equally addressing the SW *and* HW domains in the context of physically induced reliability violations. We also perform a breadth-first exploration of the target research domain. This means that we do not aim to exhaustively include all related papers (depth first) as do typical survey papers. On the contrary, we first present all relevant categories of the target domain and only instantiate representative works. Each cited work is briefly analyzed so that its categorization is fully motivated. The different aspects of hybrid works (which fall under multiple categories) are equally addressed. This results to a complete classification framework for the target domain.

2.3. The Classification Framework Methodology

A basic tool for navigation across a target research domain is the classification framework, resembling a binary tree with orthogonal splits [Kritikakou et al. 2013].

Assuming a universe set U of possible approaches on a specific topic (in our case, reliability analysis and modeling), the classification methodology proposes a split of this set into two complementary sets A and B based on a specific criterion p. For every possible approach x of the universe set, $p(x)$ can be either 1 or 0, depending on whether the criterion is satisfied by the possible approach or not. That way, each x is placed under the correct A or B subset. No x can belong in both A and B. The formalization of a single split, as proposed by our methodology is presented in Equation (1). With such consecutive top-down splits, we end up with a classification framework, the branches of which are created by various splitting criteria. In other words, we perform the same splitting procedure in each of the derived sets A, B and carry on in an inductive way.

The criterion chosen for each split results in strictly complementary branches. The lowest-level branches, or *leaves*, of the framework are the derived categories of the initial domain. These categories are not independent. Constraints are propagated between each pair in a split, which determines how to proceed in each of the two branches.

Thus, that also implies an ordering to use them. In that respect, we can see them as "constraint orthogonal" to each other. The levels of splitting define the abstraction levels of the framework. Each state-of-the-art work is mapped to the framework leaf, where it bears the most resemblance in terms of assumptions and/or implementation:

$$\begin{aligned} &U = \{x : x \text{ is a possible approach}\} \\ &A = \{x \in U : p(x) = 1\} \\ &B = \{x \in U : p(x) = 0\} \\ &A \cap B = \emptyset \text{ and } A \cup B = U \end{aligned} \tag{1}$$

The completeness of the classification framework is based on the property $A \cap B = \emptyset$ of Equation (1), which refers to categorization of *possible approaches* on the topic in question. This property propagates to the leaves of the classification framework. It is obvious that real approaches cannot comply entirely with the preceding property. Apparently, it is possible to identify state-of-the-art works that can be placed under more than one leaf of the classification framework. This is acceptable, because research involves assumptions that relax the complementarity constraint to all other possible approaches on the topic in question. When categorizing such a hybrid work, we will analyze the aspect of that work that conforms to the appropriate framework leaf.

A simple illustration of the splits performed for the creation of a classification framework can be seen in Figure 2. In this example, we use criteria p, q, and r to create the framework of Figure 2(a). The partitioning of all the possible approaches into subsets can be seen in Figures 2(b) through 2(d). Once we have reached the leaves of the framework, we populate it with state-of-the-art works. Some of these works may be hybrid, as illustrated in Figure 2(d), and are suitable for more than one leaf of the framework.

3. CLASSIFICATION OVERVIEW

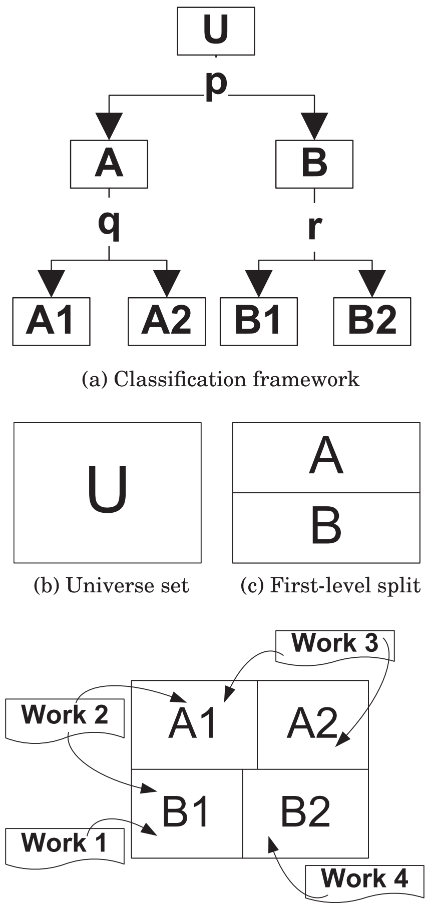
3.1. Primitive Classification Frameworks

Here we provide frameworks that alleviate the complexity of the state-of-the-art categorization, as they are reusable in a hierarchical way. We instantiate them in the final classification framework with a single node wherever necessary. Primitive classification frameworks can also illustrate the definitions of Section 2.1.

3.1.1. Digital System Instantiation. In Section 2.1, we defined the HW, SW, and operating conditions as the three components of a digital system. Here, we provide a reusable classification framework for the definition of the digital system based on its three components. The idea is that we split each component to its two possible instantiations based on the respective definitions of Section 2.1. That way, before delving into the classification of the analysis and modeling techniques, we have a systematic way with which to create a consistent definition of any digital system.

These simple splits are presented in Figure 3(a). As stated earlier, the HW may exist either in a pure HW form or an equivalent HW description. The SW of a digital system can be either service specific or generic. Finally, the operating conditions include the interference either from neighboring systems or other natural stimuli. A valid boundary with which to distinguish systems that are neighboring to the system under test is the *packaging*. As a result, any black boxes that are included in the packaging of the inspected system are assumed to be members of the system under test. All black boxes that reside outside this packaging are assumed to be neighboring systems.

A digital system definition based on these three classification frameworks will be compatible with the splitting criteria that are used for our entire classification. A digital system definition in a state-of-the-art approach may also be hybrid. For example, the SW of a digital system contains both generic and service-specific components.



(a) Classification framework

(b) Universe set (c) First-level split

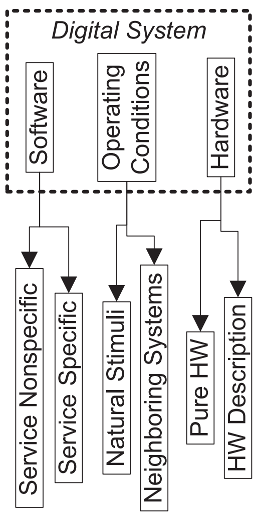
(d) Second-level splits and literature mapping

Fig. 2. Example of our classification framework methodology.

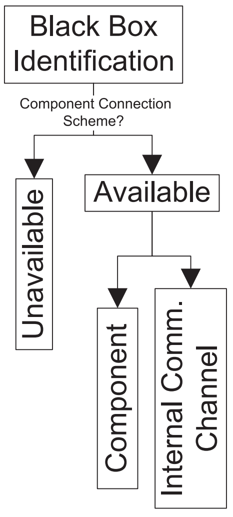
3.1.2. Black Box Identification. In many cases of reliability analysis or modeling, we need to reach a specific abstraction level of the HW of a digital system (in the form of pure HW or a HW description). In other words, we are looking for a black box within this system. Based on Definition 2.4, we can identify any black box's I/O nodes and either observe or control them. The observability is related to the detection of faults or errors, whereas the controllability refers to the injection of faults or errors to the system. To identify black boxes within a digital system, we need to check whether a component connection scheme is available.

If that is the case, we search for black boxes within this digital system (can be either components or intercomponent communication channels). In the opposite case, we can only assume the entire digital system as a black box and perceive it by its I/O. The concept of identifying black boxes is reusable across the field of reliability analysis and modeling and is presented as a reusable classification framework in Figure 3(b).

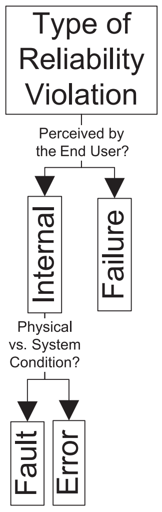
3.1.3. Type of Reliability Violation. Definitions 2.10 through 2.12 deal with the escalation of a reliability violation from a physical mechanism (i.e., fault) up to a user-perceived malfunction (i.e., failure). Figure 3(c) presents the primitive classification framework

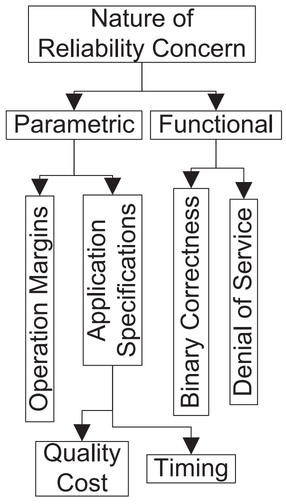


(a) Defining the three aspects of a digital system



(b) Black box identification of a digital system


Type of Reliability Violation
Perceived by the End User?
Internal
Failure
Physical vs. System Condition?
Fault
Error
(c) Type of a reliability violation



(d) Identifying the nature of a reliability concern

Fig. 3. Primitive classification frameworks.

for the type of reliability violations. The first splitting criterion is the perception by the end user, thus distinguishing failures from the errors and faults. The representation of an abstract erroneous system state distinguishes an error from a fault.

3.1.4. Nature of Reliability Concerns. In Definitions 2.8 and 2.9, we have made a distinction between functional and parametric reliability concerns, respectively. The term *concern* may refer to a type of reliability violation (i.e., fault, error, or failure). Alternatively, a reliability concern may refer to the metric that is used to quantify the propagation extent of a reliability violation. In any case, the distinction between functional and parametric reliability concerns is complete. On the one hand, we have parametric reliability concerns, referring to fluctuations of reliable operation margins or service specifications (i.e., quality cost or timing). On the other hand, we have functional reliability concerns, which involve either denial of service or binary corruption (e.g., of a memory cell). This reusable classification framework is depicted in Figure 3(d).

Based on Definition 2.12, a fault contains mostly a parametric information. A functional reliability concern deals with binary correctness, so it is very difficult to characterize a fault as functional. Faults contain no binary or system abstractions. We will mostly find the term *fault* paired with the parametric nature of reliability violations.

3.2. Reliability Analysis Versus Reliability Modeling

Having presented fundamental terminology and reusable classification frameworks, we can start presenting the state-of-the-art categorization. This section introduces the high-level splits of the classification framework. The resulting branches are covered in the next two sections. Here, we will also reflect on the constraint dependencies between these two branches. As indicated by the title of the article, the first split in the assessment of physically induced reliability threats to digital systems yields two major research domains–namely, reliability analysis and reliability modeling. Even though these branches are complementary between them (see Definitions 2.13 and 2.14), it is important to distinguish the interaction between these two domains in real applications. During the reliability analysis phase, we gather data about system

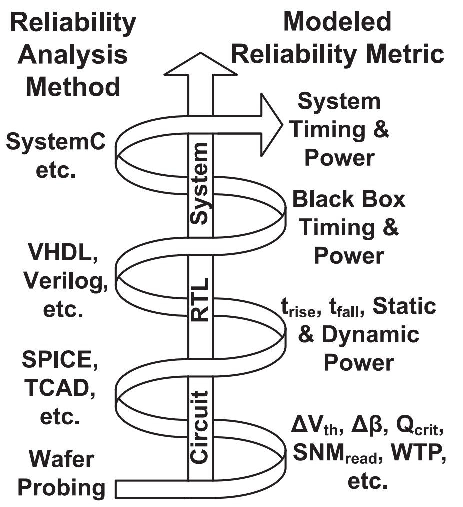

Reliability Analysis Method
Modeled Reliability Metric
SystemC etc.
VHDL, Verilog, etc.
SPICE, TCAD, etc.
Wafer Probing
System
RTL
Circuit
System Timing & Power
Black Box Timing & Power
t_{rise}, t_{fall}, Static & Dynamic Power
ΔV_{th}, $\Delta\beta$, Q_{crit}, SNM_{read}, WTP, etc.

Fig. 4. Horizontal constraint propagation from analysis to modeling; constraints also propagate vertically across abstraction levels.

performance under a variety of usage profiles. During reliability modeling, we will search for and formulate correlations between the system's usage profile (i.e., combination of SW workload and operating conditions) and the quantities measured during the analysis. Here lies an opportunity to use a reliability model—that is, a formulated correlation between a reliability metric and system usage—for the calibration of reliability analysis at a higher abstraction level. In other words, instead of performing the reliability analysis with real disturbance of the operating conditions, we can create fault or error artifacts using a reliability model that has already been created at lower abstraction levels. As a result, we identify two flows of information:

(1) *Major information flow*: Reliability analysis data about system performance under a variety of usage profiles are passed to the modeling phase in search of correlations that will create reliability models.
(2) *Minor information flow*: A reliability model calibrates a reliability analysis campaign at a higher abstraction level. The calibration may involve the aggression of fault or error injection in lieu of actual faults or errors occurring in the system.

In view of this interaction, it is important to understand the *constraint propagation* between analysis and modeling. By nature of these two phases, the major information flow occurs from the reliability analysis to modeling. Hence, constraints are propagated "horizontally" at each abstraction level. The minor information flow happens in a "vertical way" between abstraction levels so that it does not cause any cycle or mutual dependence. In Figure 4, we illustrate this concept for a wide variety of abstraction levels. Following the same logic as the abstraction level ascends, we can perform consecutive modeling and analysis attempts reaching the entire digital system. In each step, analysis data create models used to calibrate reliability analysis at higher abstraction levels. Usually, the latter is performed with HW descriptions, as in technology computer-aided design (TCAD) [Buturla 1991], SPICE [Nagel et al. 1973], VHDL [IEEE 2000], or Verilog [IEEE 2012a] and SystemC [IEEE 2012b] (in ascending order of abstraction according to the Y-chart [Gajski et al. 1983]).

The specific level of abstraction where analysis or modeling takes place is another distinguishing factor between samples of prior art. The aforementioned Y-chart (Figure 5) will be used to identify the specific abstraction level and representation where each

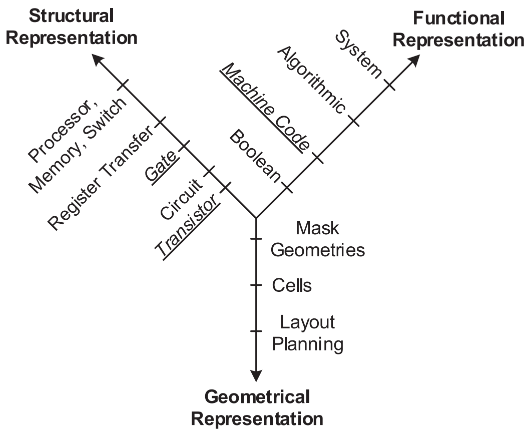


Fig. 5. Representations and abstraction levels in digital system design.

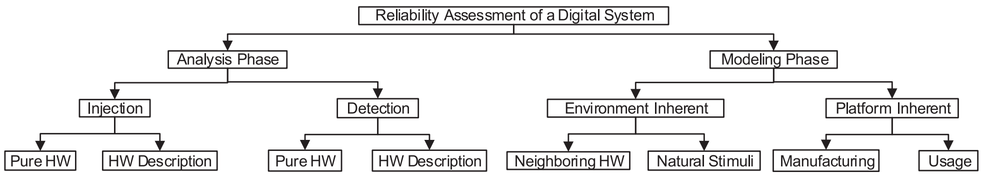


Fig. 6. The three initials levels of the binary classification presented in this article.

cited work resides. To make our classification more complete, we introduce three extra abstraction levels: *gate* and *transistor* level in the structural representation and *machine code* level in the functional representation (underlined in Figure 5).

Having covered the interaction between reliability analysis and reliability modeling across abstraction levels, we turn to each of these two branches and perform further splits in a top-down fashion. In Figure 6, we present the three initial levels of our binary classification. In the analysis domain, we distinguish two complementary steps—namely, the injection and detection of reliability violations. Each one is further split based on the criterion of pure HW availability. In the complementary modeling case, we distinguish between environment- and platform-inherent violations. The former case is further split in terms of the origin of the environmental interference (i.e., neighboring HW or natural stimuli). The latter branch is split between defective manufacturing or a specific usage profile that amplifies certain reliability violations. Due to the exponential growth of the classification framework, we deal with each of the analysis and modeling branches separately in Sections 4 and 5, respectively.

4. RELIABILITY ANALYSIS

At the reliability analysis phase, the digital system activity is monitored under specific SW workload and operating conditions (see Definition 2.13). This section will address the splits of the analysis branch in a top-down fashion, mirroring the classification framework methodology. Reliability violations—namely, faults, errors, and failures (in ascending order of severity according to Definitions 2.10 through 2.12)—are present in the system during the reliability analysis. These violations are either naturally occurring or artificially injected. In any case, a subset of these violations needs to be detected to provide data for the next modeling phase. As a result, the initial split of the

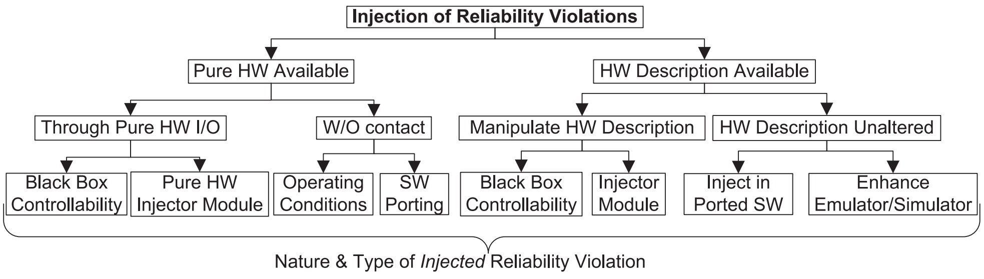

Injection of Reliability Violations
Pure HW Available
HW Description Available
Through Pure HW I/O
W/O contact
Manipulate HW Description
HW Description Unaltered
Black Box Controllability
Pure HW Injector Module
Operating Conditions
SW Porting
Black Box Controllability
Injector Module
Inject in Ported SW
Enhance Emulator/Simulator
Nature & Type of *Injected* Reliability Violation

Fig. 7. Classification of possible techniques for the injection part of reliability analysis.

reliability analysis branch is between injection and detection of reliability violations. Each one will be presented in Sections 4.1 and 4.2, respectively.

4.1. Injection

The first step of reliability analysis requires the creation of reliability violations within the digital system (i.e., faults, errors, and failures). In reality, physical mechanisms are the ones that trigger the escalation up to a user-perceived fundamental malfunction. However, it is also acceptable to create artifacts of reliability violations (of varying severity based on Definitions 2.10 through 2.12) for analysis purposes. This section will cover all possible approaches with respect to the injection of such violations. This part of the classification framework can be seen in Figure 7. The first splitting criterion is based on the availability of the digital system's HW (see Figure 3(a)). This yields two branches, one for *pure HW* and one for the case of a *HW description*.

4.1.1. HW Physically Available. In case the HW is physically available, we can inject reliability violations either by directly contacting the HW under test or without a physical conduction path. This distinction yields a pair of branches at a lower abstraction level.

In case we opt to physically connect to the target system, we can check for black boxes that are *controllable*. The inputs of these black boxes, assuming that they are controllable, can be used for the injection of reliability violations. To identify such black boxes, the generic classification framework of Figure 3(b) can be used.

Example 4.1. A representative example of injection to pure HW through contact is the Advanced Fault Injection Tool (AFIT) [Martínez et al. 1999]. This tool enforces disruptive signals through probes to nodes of a prototype.

The alternative option is to implement *injector modules* in the HW of the digital system under test. These additional pure HW regions are exclusively dedicated to the creation of disturbance that will cause reliability violations in the pure HW. In general, the addition of extra pure HW in a system, even for injection purposes, has to be assessed in terms of the underlying design and manufacturing overhead.

Example 4.2. Pure HW modules for injection purposes are used in Karlsson et al. [1995]. This paper presents the validation of a maintainable real-time system (Mars [Kopetz et al. 1989]). EMI is created in the pins of the prototype using additional antenna wires.

If we choose to inject reliability violations without a contact, the only remaining components that can be used are its operating conditions and ported SW. In the former case, we can manipulate the natural interferences that surround the digital system, thus creating the desired level of reliability violations. In the latter case, alterations can occur to generic or service-specific SW that is executed on the digital system.

Example 4.3. Injection through the manipulation of operating conditions is used in the case of Velazco et al. [1991], where the prototype is irradiated in a vacuum to evaluate its sensitivity to single event upsets (SEUs).

Example 4.4. Manipulation of ported SW for injection purposes on pure HW is seen in Wei et al. [2012]. SW executed on real multicore systems is injected with bit flips to cause artifacts of silent data corruptions (SDCs). The injected SW is used for the evaluation of an invariance-based detection technique.

4.1.2. HW in a Description Form. When pure HW is unavailable, we have to limit ourselves to a HW description. Without it, injection of reliability violations cannot be performed. A HW description is executable and allows *alterations*—in our case, for the injection of reliability violations. In addition, a simulation/emulation tool that executes the HW description can be enhanced to enable injection of reliability violations. In case we choose to make HW description alterations for injection purposes, we can either designate controllable black boxes or create additional injection modules.

Example 4.5. An example of black box controllability assessment is found in Shafik et al. [2008]. This work presents an injection technique based on a SystemC description of a digital system. The main concept behind the technique is the alteration of signal and data types to composite "fault injection enabler" types. That way, the intrusion overhead of the injection process is minimized.

Example 4.6. In Civera et al. [2001], we see the creation of SEU artifacts through the alteration of target system's VHDL representation.

If we choose to leave the HW description unaltered, the only components that can be used for injection of reliability violations are the SW executed on the HW description and the tool that emulates/simulates the HW description. In the former case, we manipulate the binary of the ported application to emulate reliability violations. In the latter case, we enhance the simulator/emulator tool to enable the controllability of the executed HW description. Given that the HW is in a description form, it would be meaningless to manipulate the operating condition for injection purposes, as the HW is not physically present. The cases where the HW description is emulated or simulated under artificial operating conditions correspond to the previous branch—namely, manipulation of the HW description. In such cases, artifacts of special operating conditions are created using existing features of the emulator/simulator (e.g., SPICE simulation for a specific temperature) and not by adding new ones.

Example 4.7. Emulator/simulator enhancement for injection purposes is featured in Ragel et al. [2006]. Bit flips are injected at the instruction level of a processor to verify a proposed monitoring technique. Injection of bit flips occurs by corrupting bits of the memory where the set of instructions is stored, after application porting. To enable this injection, the authors have modified the SimpleScalar tool set [Burger et al. 1997]. Similarly, "one-clock-cycle" delay can be added in to the erroneous black box to cause timing violations [Li et al. 2009].

Example 4.8. Application binary modification can be seen in Mao et al. [2010]. This work presents a secure processing methodology for embedded systems. The target platform is simulated (HW description form). Artifacts of erroneous states are created by creating bit flips in the application binary.

We note that the context of the preceding paper is malicious attacks, which fall beyond our scope (Section 1). However, this is still a representative example, as corruption of single bits is a valid artifact of physically induced violations (e.g., soft error).

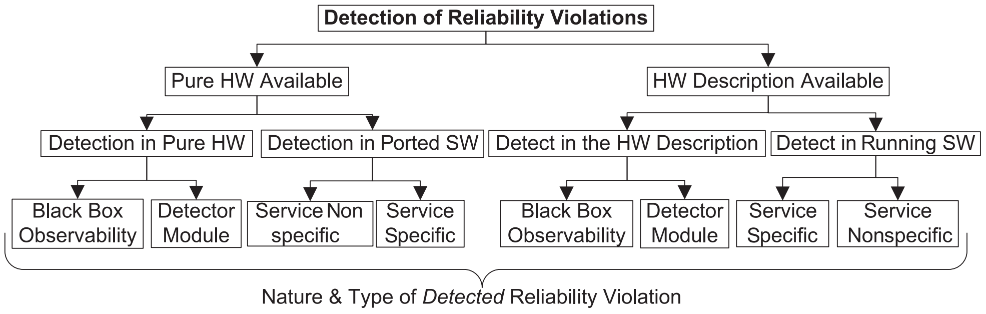


Fig. 8. Classification of possible techniques for the detection of reliability violations.

4.1.3. Type and Nature of Injected Reliability Violation. We have covered the possible approaches toward the injection of reliability violations in a digital system. We have explored all components of the digital system that can be manipulated to create either real or artificial reliability violations. Some of the approaches involve alterations in the HW or SW of the system. Others involve the manipulation of the operating conditions.

In any case, the nature of the injected reliability violation (see Definitions 2.8 and 2.9) also needs to be addressed. Furthermore, the possible injection approaches have to be connected with the type of reliability violation (see Definitions 2.10 through 2.12) that is being injected in each approach. For that purpose, we require the reusable classification frameworks of Figures 3(c) and 3(d). These reusable concepts can be instantiated in each leaf of the injection classification framework. For example, if we are to manipulate the operating conditions for the sake of injection, the type of violation that is injected is a fault, as only the underlying physical mechanism is triggered. Alternatively, if we choose to inject reliability violations by manipulating the HW description of a digital system, we are free to inject either faults or errors based on the features of the available simulator/emulator. Each of the injected violations can be either parametric or functional depending on the target physical mechanism. Table I summarizes the nature and type of injected reliability violations for each cited work. We also reflect on the digital system abstraction and representation in each case.

4.2. Detection

Having injected faults or errors in a digital system, we need to detect their impact across various abstraction levels. This detection provides information about the degree of propagation and the percentage of the injected violations that lead to user-perceived malfunctions (i.e., failures). That way, we can gather measurements and try to correlate them to the SW workload and operating conditions that have been applied to the digital system under test. The latter stage is equivalent to reliability modeling and will be covered in Section 5. The physical availability of the system's HW is chosen again as the initial splitting criterion for the detection branches. The HW of the digital system may be physically present (pure HW) or instantiated as a HW description (Figure 8).

4.2.1. Pure HW. Given that the system is physically implemented, we can turn either to *pure HW* or *SW* utilities to detect reliability violations. In the pure HW case, we can either assess the observability of target black boxes or incorporate detection modules within the system. The former option requires the application of black box exploration in the pure HW as outlined by the classification framework of Figure 3(b). The latter option, as in the case of injection, introduces a manufacturing overhead.

Example 4.9. Velazco et al. [1991] illustrate the observability assessment of pure HW black boxes for detection purposes. A pair of microprocessors are tested for tolerance

Table I. Type and Nature of Injected Reliability Violations: Abstraction Evaluation

Example	Work Cited	Type	Nature	Representation	Abstraction	Details
4.1	[Martínez et al. 1999]	E	F	Structural	Processor	IC pins forced to specific voltage level
4.2	[Karlsson et al. 1995]	F	P	Structural	Processor	EMI bursts (no binary abstraction)
4.3	[Velazco et al. 1991]	F	P	Structural	Processor	Particle interference/IC irradiation
4.4	[Wei et al. 2012]	E	F	Functional	Machine Code	Branch instructions corrupted
4.5	[Shafik et al. 2008]	E	N/A	Structural	Register Transfer	SystemC injection data types
4.6	[Civera et al. 2001]	E	F	Structural	Circuit	VHDL-enabled binary corruption
4.7	[Ragel et al. 2006]	E	F	Functional	Machine Code	Corrupted instruction memory
4.7	[Li et al. 2009]	E	F	Structural	Processor	Effects of permanent violations
4.8	[Mao et al. 2010]	E	F	Functional	Machine Code	Bit flips introduced in the binary

Note: Only works cited in the section are considered. Type of reliability violations: F, fault, E, error. Nature of reliability violations: F, functional; P, parametric; N/A, insufficient information.

against SEUs. Part of the verification process involves the detection of user-perceived reliability violations by comparing the results of the irradiated pure HW with a set of reference values. Black box observability is also employed for the analysis of "voltage noise in production processors" [Reddi et al. 2011].

Example 4.10. Velazco et al. [1991] use additional pure HW modules (on the system and as peripherals) for detection purposes. Built-in detection schemes have been used to classify internal erroneous states by their identification messages. Similar detector modules are observed in the evaluation of Fault-Tolerant Architecture with Stable Storage Technology (FASST) [Martínez et al. 1999].

Velazco et al. [1991] is an example of a hybrid paper, as it is categorized under two neighboring detection leaves. This work features detection of reliability violations on pure HW both by observing black boxes of the HW and by utilizing built-in detector modules. Furthermore, we have seen an instantiation of this paper in Example 4.3, which is strictly related to the injection of reliability violations to pure HW. A similar observation holds for Martínez et al. [1999], which contains aspects both from the injection and detection of reliability violations. In Example 4.1, we identified the injection of reliability violations to pure HW and without a contact for this paper. We also identify the respective detection technique (Example 4.10), which is based on the detection modules that are implemented within the system under test.

When SW is chosen for detection purposes, we can choose between generic SW utilities or utilities that are related to a specific service delivered by the system.

Example 4.11. An example for the generic SW case is Wei et al. [2012]. A monitoring thread tracks branch instructions across threads and assesses the instructions' correctness. Invariant attributes shared between threads are derived at compile time and are used as reference for comparisons at runtime.

Wei et al. [2012] is an example of a hybrid paper. It has also been instantiated in Example 4.4 for the case of injection directly to pure HW by manipulating ported SW.

Example 4.12. The alternative case of detection using service-specific SW can be observed in Kim et al. [2009]. A fault-tolerant streaming engine is presented. The paper refers to virtual application streaming and aims to protect it against network failures. To enable this protection, the connection manager of the client is enhanced to detect network failures and initiate the fault-tolerant scheme, thus avoiding streaming outage on the client's side.

4.2.2. HW Description. If we use the HW description for detection purposes, we can either assess the observability of the design's black boxes or introduce additional detection modules. In the former case, the observability of a design's black boxes depends highly on its complexity as well as the capabilities of the verification tool that is used (e.g., emulator or simulator of the HW description).

Example 4.13. A paper illustrating the observability assessment of a HW description's black boxes is Civera et al. [2001]. The detection of SEU artifacts is achieved by observing the primary output of the system and comparing to a reference. The memory elements are checked at the end of each simulation in case a mismatch is observed with the expected stored values. In the first case, the assumed black box is the entire system, and its output is simply observed. In the second, we narrow down to the memory of the system.

Civera et al. [2001] is another hybrid paper that shares features of the injection and detection domains. It has also been instantiated in Example 4.6 as a representative case for injection of reliability violations using HW-described injector modules.

Example 4.14. The addition of extra modules in a HW description for the detection of reliability violations is illustrated in Fei et al. [2007]. This paper presents a scheme that aims to guarantee code integrity, especially in the presence of code injection attacks or binary corruption from cosmic particles. The idea is to have a table of hash values for code blocks of the running application in advance and compare them to the value derived at runtime. Other approaches employ comparison of the application's course of action with an a priori–derived monitoring graph [Mao et al. 2010; Meixner et al. 2008].

Mao et al. [2010] has also been instantiated in Example 4.8 (hybrid work), where the aspect of functional error injection in a HW description is raised.

In case we choose the SW that is ported on the HW description for detection purposes, we can opt for generic SW (e.g., OS utilities) or SW that is connected to a specific service delivered by the system (e.g., device driver).

Example 4.15. The use of service nonspecific SW for detection is observed in Ragel et al. [2006]. This paper proposes a framework for the mitigation of soft errors or code injection attacks on a specific instruction set architecture (ISA). The ISA has been modified so that the checksums can be decrypted and confirmed at runtime. Additionally, in Fei et al. [2007] processor operations beneath the instruction abstraction level ("micro-operations") are added to enable the comparison of hash values for the detection of corrupt basic blocks of the execution trace.

Fei et al. [2007] is another hybrid paper. In Example 4.14, we instantiated it for extra HW description modules, enabling the detection of corrupt instructions.

Example 4.16. Finally, the detection of reliability violations using service specific SW (ported on a HW description) is observed in Geist et al. [2002]. This paper presents algorithms that have embedded fault-tolerant features. In one class of such algorithms, the units of execution (UEs) share local information to deliver their result. If a faulty UE is identified (one that seizes to send results), it is removed from the respective "neighborhoods."

4.2.3. Nature and Type of Detected Reliability Violations. With the preceding splits, we have covered all possible ways of reliability violation detection in a digital system. We explore the types and natures of detected reliability violations using the classification frameworks of Figures 3(c) and 3(d) (Table II). We also perform an abstraction-level evaluation of each cited work according to Figure 5. It is important to note that it will always be meaningful to detect reliability violations at least one abstraction level higher than the ones injected in the digital system. Detection at the abstraction level of the injection phase will always lead to a tautology, because the reliability violations that are present in the system are always known in advance. A representative example of this concept is shown in [Evans et al. 2012a], where binary corruption is injected in the flip-flops of an ASIC design and detected reliability violations reside in a much higher abstraction level.

5. RELIABILITY MODELING

In the previous section, we presented all possible approaches to the reliability analysis of a digital system. First, we explored the injection of reliability violations in a system. The impact (e.g., propagation) of these violations is detected during system operation. For example, the percentage of violations that cause user-perceived malfunctions can

Table II. Type and Nature of Detected Reliability Violations: Abstraction Evaluation

Example	Work Cited	Type	Nature	Representation	Abstraction	Details
4.9, 4.10	[Velazco et al. 1991]	E	F	Functional	Algorithmic	Program sequencing/addressing/output
4.9	[Reddi et al. 2011]	F	P	Structural	Processor	Hands-on voltage noise measurements
4.10	[Martínez et al. 1999]	E	F	Structural	Register Transfer	Parity/comparison errors of HW modules
4.11	[Wei et al. 2012]	E	F	Functional	System	SDCs detected in the execution output
4.12	[Kim et al. 2009]	E	F	Functional	System	Network outages are detected
4.13	[Civera et al. 2001]	E	F	Structural	Processor	Binary checking of output and memory
4.14, 4.15	[Fei et al. 2007]	E	F	Functional	Machine Code	Binary corrupted instructions detected
4.14	[Mao et al. 2010]	E	F	Functional	System	Monitoring graph discrepancies
4.14	[Meixner et al. 2008]	E	F	Functional	System	Checking von Neuman core tasks
4.15	[Ragel et al. 2006]	E	F	Functional	Machine Code	Checksum comparison for block of code
4.16	[Geist et al. 2002]	E	F	Structural	Register Transfer	Out-of-service units detected

Note: Only works cited in the section are considered. Type of reliability violations: F, fault; E, error. Nature of reliability violations: F, functional; P, parametric; N/A, insufficient information.

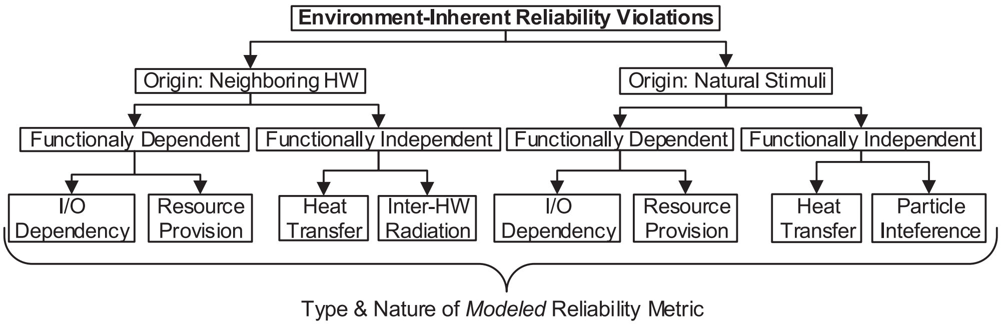


Fig. 9. Possible modeling approaches for environment-inherent reliability threats.

be tracked. Such information needs to be postprocessed to create formal correlations between the usage profile applied to the system and its reliability. This is performed during reliability modeling (see Definition 2.14).

The initial distinction refers to the source of the reliability violations—namely, the physical mechanism that lies under the target reliability violation. A valid distinction of that sort is between environmentally inherent mechanisms and the ones that are rooted within the physical boundaries of the system platform. In Section 3.1.1, we designated the *packaging* of a digital system as a valid physical boundary. As a result, any interference coming beyond the system packaging is considered environmental. All other mechanisms are assumed to occur within the platform. The former branch will be covered in Section 5.1 and the latter in Section 5.2.

5.1. Environment-Inherent Reliability Violations

This part of the modeling classification deals with the reliability models for violations that are rooted in the environment of the system. We can further split between interference from neighboring HW or actually natural stimuli. In the former case, we incorporate all sorts of interaction between systems. In the latter case, we include all mechanisms that happen by nature in the "ecosystem" of the system under test without originating from another system. The classification of modeling techniques for environmentally inherent reliability violations is illustrated in Figure 9.

5.1.1. Originating from Neighboring HW. In the case of interference from a neighboring system (a system beyond the packaging boundaries of the system under test), we can check whether any *functional correlation* is present between the system and its neighbors (i.e., an interdependence may exist for all systems to be able to operate). In case the system under test is functionally dependent to neighboring systems, an *I/O dependency* can be present, as in the case of systems connected in series.

Example 5.1. Srivaree-ratana et al. [1998] performs reliability modeling for a set of systems that are functionally correlated through the exchange of data. This work formulates the "probability that all nodes can communicate with all other nodes." A trained artificial neural network (ANN) illustrates the correlation of network reliability with the reliability of the network's links and its topology.

Alternatively, the two systems (the one under test and its neighbor) may be correlated in terms of *resource provision*. In some cases, this refers to supply from the electrical grid. Such power engineering issues are considered out of the scope of our survey (see Section 1). However, the domain of battery-powered embedded systems presents an excellent source from which to draw relevant papers. In such designs, a battery is a separate package than the processor or any intellectual property (IP) block. This domain is fully compatible to our formulation of the current leaf.

Example 5.2. More specifically, the approach of power deregulation presents interesting reliability considerations for the target design [Kim et al. 2007]. Battery degradation models are used to determine the output battery voltage as a function of time and variable workload. These models are used to impose constraints on the allowed frequency that can be applied to the processor of the embedded system.

When the neighboring system is not functionally coupled to the system under test, the interferences we can inspect may require a conduction path. The case where a conduction path is required for the interference to reach the digital system narrows down to *heat transfer*. The complementary branch deals with *inter-HW radiation*.

Example 5.3. For the case of inter-HW heat transfer, the automotive industry is a valid source of representative papers. In Bharathan et al. [2008], we can see an attempt to find the optimal air cooling for power electronics used in the automotive domain and especially hybrid electric vehicles (HEVs).

Example 5.4. As for inter-HW radiation, a representative modeling paper is Chen et al. [2011]. It targets the interference between wireless communication infrastructures. A “simulation model for adjacent-channel interference in IEEE 802.11b networks” is presented that is compatible with the OMNeT++ framework [Varga et al. 2008]. The reliability metric is the packet delivery ratio (PDR).

5.1.2. Originating from Natural Stimuli. Apart from neighboring systems, nature may also provide a lot of interference that can cause reliability violations to digital systems. Again, we check whether nature is *functionally coupled* to the system under test. If such a functional coupling exists, the next split can be again between *I/O dependency* (namely, the use of natural stimuli as a source of information) and *resource provision*. In the former branch (I/O dependency), the area of sensors contains many examples where natural stimuli function as sources of information *and* interference for a system.

Example 5.5. A representative example of that concept is Andrei et al. [2004]. Different resistance components of the sensor have been evaluated after thermal aging at 150°C. For this paper, the natural stimulus that is source of information for the system is the applied pressure. The ambient temperature is another natural stimulus, which interferes with the system.

In the case of resource provision, renewable energy sources for mass power production are outside the scope of this article, as they refer to power engineering (providing resources to the power grid). However, the emerging domain of energy harvesting for small-scale grids or even embedded systems features significant research that can be classified under the leaf in question. In this domain, natural stimuli (e.g., solar or kinetic energy) are used as a means to power digital systems [Benecke et al. 2012]. A representative application is wireless sensor networks [Raghunathan et al. 2005].

Example 5.6. A very representative sample of prior art deals with the modeling of cooperating nodes of a solar, energy harvesting, wireless sensor network to devise “error control schemes” [Jalali et al. 2012]. In this process, there is modeling of the energy efficiency versus reliability trade-off.

If nature is uncorrelated to the system under test, we can further distinguish based on whether a *conduction path* is required for the natural interference to reach the digital system under test. The branch that requires a conduction path is narrowed down to *heat transfer*. The complementary branch deals with the operation under *particle*

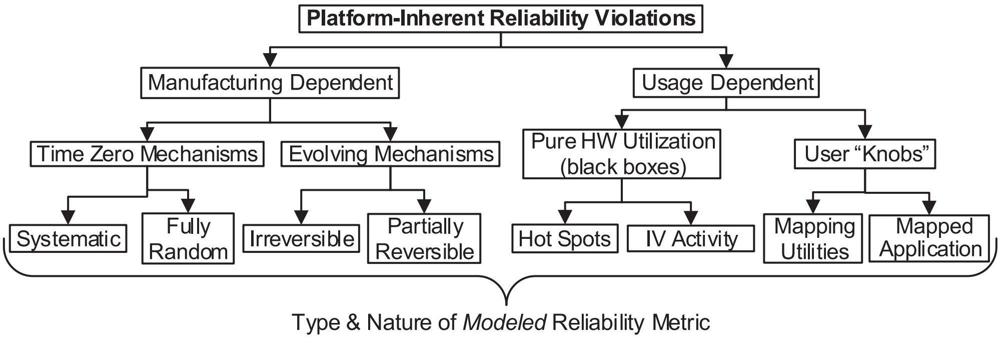


Fig. 10. Possible modeling approaches for platform-inherent reliability threats.

interference. To the best of our knowledge, reliability modeling due to heat transfer is always related to specific applications that involve elevated temperatures. Greenwell, et al. [2011] illustrate the extreme temperatures observed in various aspects of the automotive domain (e.g., HEVs, as discussed earlier for Bharathan et al. [2008]). In the majority of the heat transfer modeling papers, it is neighboring HW that creates the extreme temperatures. We can populate the current leaf with modeling attempts that concentrate on heat transfer from the ambient environment while agnostic of the neighboring HW or the specific "ecosystem" where the digital system is operating.

Example 5.7. Andrei et al. [2004] concentrate on the ambient temperature of the system under test while agnostic on other "background" systems. The response of this system is assessed for various degrees of thermal aging.

Andrei et al. [2004] is another hybrid paper, as it borrows elements from two possible approaches toward reliability modeling. In Example 5.5, we see how this work conforms to the modeling of reliability violations due to natural interference (temperature) for a system that also processes natural signals (pressure sensor).

Example 5.8. Naseer et al. [2007] is a representative reliability modeling work on particle interference. It deals with the critical charge (Q_{crit}) of submicron SRAM cells. BER values are presented for different irradiation scenarios (CREME96 tool [Tylka et al. 1997]). Modeling of particle interference are also presented in [Baumann 2005]. A correlation is made between SRAM soft error rate (SER) and the target technology node. Gordon et al. [2004] present the modeling of particle flux for different particle energies and geographical regions.

5.2. Platform-Inherent Reliability Violations

In this domain of reliability modeling, we are dealing with violations rooted within the digital system's physical boundaries. The impact of the underlying physical mechanism may propagate to the end user. As a result, a valid criterion for the first split in this subdomain is whether the responsible mechanisms are caused by manufacturing defects or are occurring because of specific system utilization (Figure 10).

5.2.1. Violations Depending on Manufacturing. We can further split based on the time instance at which the effects of these mechanisms (i.e., the respective faults) become apparent. This distinction yields the *time-zero* and *evolving mechanisms* branches.

When the platform-inherent physical mechanisms cause reliability violations from time-zero, we can have time-zero device variability, which can be either systematic or fully random. Variability does not make the system completely dysfunctional; however, its impact can be perceived from time-zero.

Example 5.9. For the case of systematic variability, a representative modeling attempt is found in Gupta et al. [2004]. This paper focuses on across chip linewidth variation (ACLV) for the polysilicon level. It presents a tool that models the timing of a circuit under systematic variations. There is also reference to the Bossung plot [Bossung 1977], which is a commonly used graph correlating line width with the defocus condition during processing. The authors report a reduction of timing analysis uncertainty even by 40% compared to traditional methods.

Example 5.10. Ye et al. [2011] deal with the modeling of fully random variability. It models random dopant fluctuation (RDF) and line-edge roughness (LER). The presented methodology uses compact modeling and SPICE simulations instead. An example of the reliability metrics modeled with this tool is the standard deviation of the threshold voltage ($\sigma_{V_{th}}$), which is correlated to the transistor width. A similar sample of prior art is the VariaSim tool [Romanescu et al. 2007].

In the complementary branch lie the mechanisms that are triggered during manufacturing and have a perceivable impact on system operation, only as its lifetime progresses. That impact cannot be perceived at time-zero. Some of these mechanisms are *irreversible*—namely, their impact is strictly increasing with system lifetime. Other mechanisms are evolving during the system lifetime, but they contain a *partially reversible* feature—that is, their impact is partially revoked during the system lifetime. These two types create the branches of the current split.

Example 5.11. An example of accumulating physical mechanisms that may cause reliability violations is time-dependent dielectric breakdown (TDDB). A modeling attempt can be found in Kimura [1999]. A series of experimental measurements yields formal correlations between the cumulative failure of the tested devices and TDDB intensity metrics. Similarly, Srinivasan et al. [2004] illustrate the impact of scaling on TDDB and electromigration (EM).

Example 5.12. A representative modeling attempt of partially reversible physical mechanisms has been made for NBTI in Kaczer et al. [2010]. This paper correlates the activity of interface traps (which capture and emit minority carriers in the device channel) with the fluctuations caused to the threshold voltage (V_{th}) of submicron pMOSFETs. A percolation model is created to mirror the "variations in the local potential" found in the channel of pMOSFET.

5.2.2. Violations Depending on System Utilization. When the utilization of the system is the source of reliability violations, we can check whether the user can have any control over the way that the system is utilized. If that is not the case, we can look into pure HW utilization, as a result of the system's operation. This branch is static in nature in the sense that the user has no control over the way that the HW is utilized. The complementary branch refers to user-controlled HW utilization either by manipulating the ported SW or by configuring the HW utilization options. In other words, there is a set of "knobs" at the user's disposal that allow HW utilization adjustment (e.g., cutting supply to specific HW black boxes or manipulating the application mapping).

We need to note that with the term *utilization*, we refer only to the way in which HW black boxes are utilized. Operating conditions are not included in this node of the classification framework because they have already been covered in Section 5.1, where the various types of environmental interference to the system under test are presented.

In case the user is unable to control the system utilization, we have to focus on the pure HW activity. This requires a black box exploration of the system HW, which can be performed according to Figure 3(b). Then, we can distinguish between the two major issues of pure HW utilization—namely, the existence of hot spots across the pure HW and regions that exhibit a specific current or voltage activity.

Example 5.13. Regarding hot spot modeling, a representative work is Skadron et al. [2004]. Based on the duality between an RC network and the heat transfer problem, the authors propose a compact thermal modeling technique that uses as input the power trace in each physical block of the system's pure HW.

Example 5.14. In Toledano-Luque et al. [2011], experimental data illustrate the impact that a single *interface trap* has on the threshold voltage V_{th} of the device. Correlations are made between the gate voltage (V_{gs}), the occupancy probability of the interface trap, and the final V_{th} value. [Ma et al. 2011] deal with "localized IR drop analysis" by correlating IR drops occurrence with the switching activities of a set of cells that share the same power and ground lines.

When the utilization of the HW is coupled to or configurable by the user, we assess both the mapping utilities of the system and the actually mapped application. The former component is generic, and the latter is coupled to a specific delivered service.

Example 5.15. In Velazco et al. [1991], different susceptibility to heavy ions is observed under the same irradiation conditions but for different executed benchmark. The irradiation impact is reduced when the benchmarks are executed with a deactivated cache (mapping utility knob of the 68882 coprocessor). Similarly, in Biswas et al. [2005], we see different AVF values for the considered benchmark applications. A paper that deals with the impact of ported SW on reliability metrics is Kim et al. [2011]. This work employs a genetic algorithm to maximize the inductance impact (di/dt) on the power delivery network (PDN). Series of instructions cause different voltage droops to the PDN. Similarly, "a pattern of control flow and microarchitectural activity" can be correlated to emergency events related to voltage supply noise [Reddi et al. 2010].

It is interesting to observe how the work of Velazco et al. [1991] is instantiated in many classification framework leaves. This is a valid example of a hybrid paper. It combines aspects of both reliability analysis and modeling. In Section 4.1, we instantiated this paper as a representative example fault injection to pure HW without a physical contact and through operating conditions (Example 4.3). In Section 4.2, this work was cited for error detection in pure HW using both built-in detector modules and primitive output observation (Examples 4.9 and 4.10). In this section, we have completed the presentation of this hybrid by analyzing its modeling aspects.

5.3. Nature of Modeled Reliability Metrics

In the two previous sections, we covered the modeling of a very wide range of reliability violations. We need to explore the nature of the reliability metric that is modeled in each case (i.e., for each leaf of the classification frameworks in Figures 9 and 10). By employing the reusable classification framework of Figures 3(c) and (d), we identify the type and nature of the reliability metric that is modeled in each of the cited papers of this section. Furthermore, we assign the cited works to the abstraction levels and representations of Figure 5. The resulting evaluation is presented in Table III.

6. DISCUSSION

Following the classification of prior art, we should reflect on the impact of the proposed framework. Initially, we observe certain trends in the target research domain. These observations are presented in Section 6.1. Moreover, in Section 6.2, we illustrate how

Table III. Type and Nature of Modeled Reliability Metrics: Abstraction-Level Evaluation

Example	Work Cited	Type	Nature	Representation	Abstraction	Details
5.1	[Srivaree-ratana et al. 1998]	E	P	Functional	System	All-terminal network reliability formulation
5.2	[Kim et al. 2007]	F	P	Functional	System	Battery models specify clock frequency
5.3	[Bharathan et al. 2008]	F	P	Structural	Processor	Laminar flow modeled for different heat fluxes
5.4	[Chen et al. 2011]	E	F	Functional	System	PDR modeled for variable wireless interference
5.5, 5.7	[Andrei et al. 2004]	F	P	Structural	Circuit	Component resistance vs. ambient temperature
5.6	[Jalali et al. 2012]	F	P	Functional	System	Efficiency vs. reliability trade-off
5.8	[Naseer et al. 2007]	E	F	Structural	Processor	BER values vs. different irradiation scenarios
5.8	[Baumann 2005]	E	F	Structural	Gate	Modeling of the SER for SRAM cells
5.8	[Gordon et al. 2004]	F	P	Structural	Transistor	Particle flux for different geographical regions
5.9	[Gupta et al. 2004]	F	P	Geometrical	Cell	Timing analysis for representative cells
5.10	[Ye et al. 2011]	F	P	Structural	Transistor	$\sigma_{V_{th}}$ calculated under RDF and LER
5.10	[Romanescu et al. 2007]	F	P	Structural	Gate	Static statistical timing simulations
5.11	[Kimura 1999]	F	P	Structural	Transistor	Analytical model for t_{BD} based on TDDB impact
5.11	[Srinivasan et al. 2004]	F	P	Structural	Transistor	Impact of scaling on EM and TDDB
5.12	[Kaczer et al. 2010]	F	P	Structural	Transistor	V_{th} fluctuations and V_{th0} values
5.13	[Skadron et al. 2004]	F	P	Geometrical	Layout	Thermal map derived from module power traces
5.14	[Toledano-Luque et al. 2011]	F	P	Structural	Transistor	V_{th} fluctuations correlated to V_{gs} waveform
5.14	[Ma et al. 2011]	F	P	Geometrical	Cells	IR drops vs. activity of neighboring cells
5.15	[Velazco et al. 1991]	F	P	Structural	Processor	LET vs. cross section for different usage profiles
5.15	[Biswas et al. 2005]	E	F	Structural	Processor	Calculation of the AVF reliability metric
5.15	[Kim et al. 2011]	F	P	Structural	Processor	Stressmarks causing maximum di/dt impact
5.15	[Reddi et al. 2010]	F	P	Structural	Processor	Recognition of patterns leading to voltage droops

Note: Only works cited in the section are considered. Type of reliability violations: F, fault; E, error. Nature of reliability violations: F, functional; P, parametric; N/A, insufficient information.

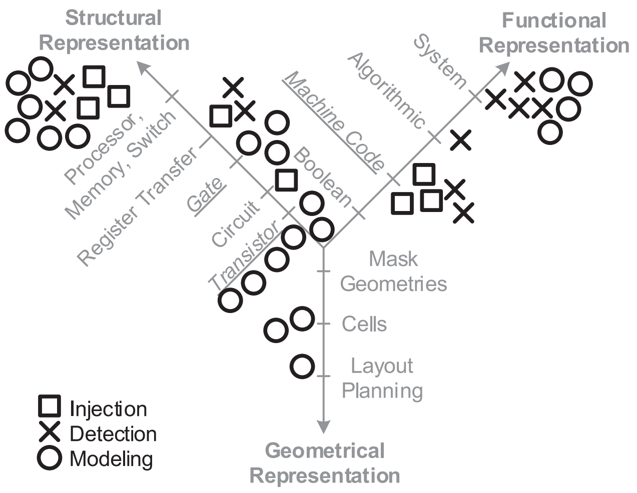


Fig. 11. Visualization of Tables I through III. A mark is placed for each cited work in the corresponding representation and abstraction level.

the presented methodology can be reused and extended by interested readers for the assessment of specific subdomains in reliability analysis and modeling. We use two samples of prior art to substantiate this reusability.

6.1. Trends and Observations in Reliability Analysis and Modeling

As a result of our classification, there is a series of observations that can be made following our inspection of prior art. Initially, we notice that state-of-the-art works are spread across a variety of abstraction levels and system representations. Figure 11 places a mark for each cited work in the appropriate abstraction level and representation according to the enhanced Y-chart of Figure 5. We notice a concentration of analysis techniques around the structural and functional representations. This is to be expected, as geometry-/layout-aware analysis of integrated system reliability comes with increased computational overheads (tools have to account for parasitic elements in a layout-aware analysis). Geometrical phenomena are typically briefly analyzed, and their *models* are used for reliability analysis experiments at higher abstraction levels.

We can also notice a segmentation of analysis and modeling techniques across abstraction levels. Such techniques rarely span across more than a handful of abstraction levels. Even though the concept of cross-layer reliability is already manifested in the literature in the form of diverse mitigation schemes [Carter et al. 2010], it is noteworthy that analysis/modeling techniques have not followed a similar cross-fertilization. We can identify certain prior art that attempts to reconcile reliability/lifetime information across abstraction levels [Evans et al. 2012b]; however, it is difficult to identify a vibrant cross-layer reliability analysis/modeling literature.

A major reason behind this observation lies primarily on the interfaces that are required for reliability information to propagate from the material science and characterization domain to higher levels of digital system design. This effort may succeed, but typically the ascend does not span the entire scale of abstraction levels. The existence of hybrid work in analysis and modeling is an observation that creates hope for a more holistic perception of digital system reliability. Already, prior art typically borrows elements from a variety of reliability analysis and modeling subdomains. The abundance of hybrid papers identified in the current classification explicitly confirms this claim. As a result, achieving a cross-layer reliability analysis/modeling paradigm requires the synthesis of segmented approaches that already exist in the literature. For that to happen, a seamless interaction between researchers working on various abstractions of reliability should take place. From the material characterization domain up to the level of system architecture, dependability, and resiliency, a succession of analysis and modeling phases should enable truly holistic reliability assessment of digital systems.

Finally, it is interesting to look in the type and nature of reliability violations that are injected and detected in analysis campaigns. From Tables I and II, we see that the majority of cited work concentrates on the detection of functional errors. The scope of injection is much wider by comparison. This is not surprising, as the detection of parametric errors/faults is typically more complex. A functional violation is easy to identify because it is abstract: either a functional error occurs or not. However, a parametric violation is represented by a continuous or generally multiple-value random variable. This makes parametric violations more difficult to classify and hence to detect. As a result, it may make more sense to *inject* a parametric violation and then track its propagation by *detecting* the corresponding functional aftereffects.

6.2. Reusability and Extensibility of the Proposed Classification Framework

In Sections 4 and 5, we instantiated representative samples in the appropriate leaves of our classification framework. This binary diagram can be extended and reused when the attention needs to be drawn to subdomains of reliability analysis and modeling. For the sake of brevity of this article, we have restricted to only five levels of splitting. However, readers interested in technical development for reliability analysis or modeling can further elaborate the framework by adding leaves in lower levels. In the current section, we use two test cases and demonstrate the reusability/extensibility of our classification framework. Let us consider the TDDB mechanism. It manifests itself both in the gate stack of devices and across wires of an integrated system. We also assume two modeling papers: one for TDDB manifestation on wires [Bekiaris et al. 2011] and one for devices [Kimura 1999]. In case we would like to systematically classify these two papers, we should further split the "irreversible mechanism" leaf of our classification framework (see Figure 10) into two complementary categories. The "wires versus transistors" split comes about, as illustrated in Figure 12, where the resulting leaves are again complementary.

Thus, the two papers can be seamlessly classified in the specialized leaves. Following the same concept, the classification presented in this article can be extended in specialized areas of reliability analysis and modeling. Interested readers can start from the splits proposed in this article and elaborate the framework in a way that best suits their research area. Samples of prior art can ripple through the existing and newly introduced splits (see Figure 12).

7. CONCLUSIONS

In this survey, we have covered a wide variety of prior art dealing with the analysis and modeling of physically induced threats to digital system reliability. Analysis and modeling are crucial for the development of efficient mitigation solutions that can guarantee the reliability of modern electronic systems. Especially in view of reduced device dimensions following the downscaling trend, a good understanding of reliability violations and their underlying physical mechanisms is required.

Existing papers on reliability analysis and modeling classify either related terms or available techniques, indicating an evident interest in this domain. However, some classifications do not handle reliability violations in a systematic way. In other

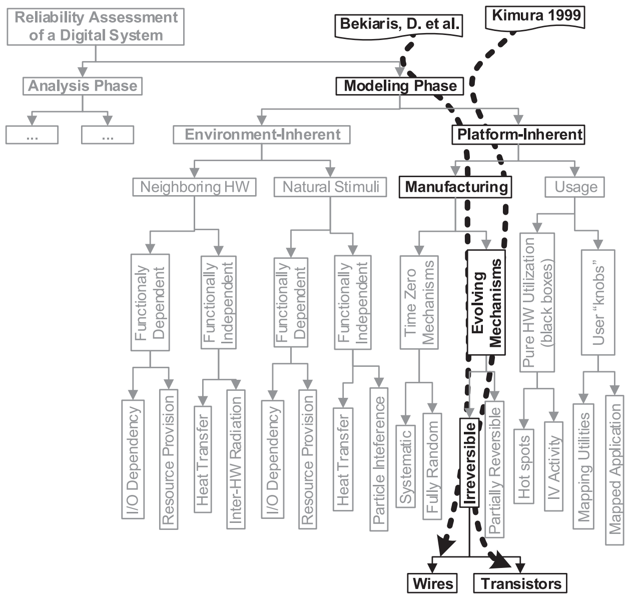


Fig. 12. Extending the proposed framework to lower classification levels.

cases, they appear to be application specific. In view of these features, we propose a novel and systematic classification of prior art on reliability analysis and modeling. We use terminology that is as in-line as possible with terms used already in the literature.

Our classification methodology is guaranteed to provide orthogonal categories because it is based on a binary decision diagram. The set of possible approaches toward the topic in question is partitioned into subcategories in a top-down way. Each category, at any given point, is split into two branches based on a distinguishing criterion. Hence, a binary tree is created, the leaves of which are complementary. The target research domain (i.e., the root of the tree) is thoroughly covered. In addition, the definitions presented herein do follow the complementarity principle of the classification framework. The orthogonal nature of the classification framework allows mapping of an inspected paper to more than one category. This is acceptable, as real research scarcely is strictly complementary to all other alternative approaches. As a result, such hybrid papers are classified easily: each aspect of the paper is analyzed when the appropriate classification framework leaf is inspected. We should note that the ability to properly account for hybrids is not supported in a systematic way by related classifications.

The overall result of our categorization can be seen in Figure 13. All branches of the classification framework have been populated with at least one representative paper. A brief description of each paper supports the classification choice. To verify our classification methodology, we have drawn papers from many domains ranging from the architecture to the technology abstraction levels. That way, the systematic quality and the completeness of our classification framework are substantiated. For each cited paper, we identify the nature and type of target reliability violations. Furthermore, we map each cited work to the corresponding representation and abstraction of digital system design. This triggers interesting observations regarding the potential for a truly cross-layer reliability analysis and modeling paradigm.

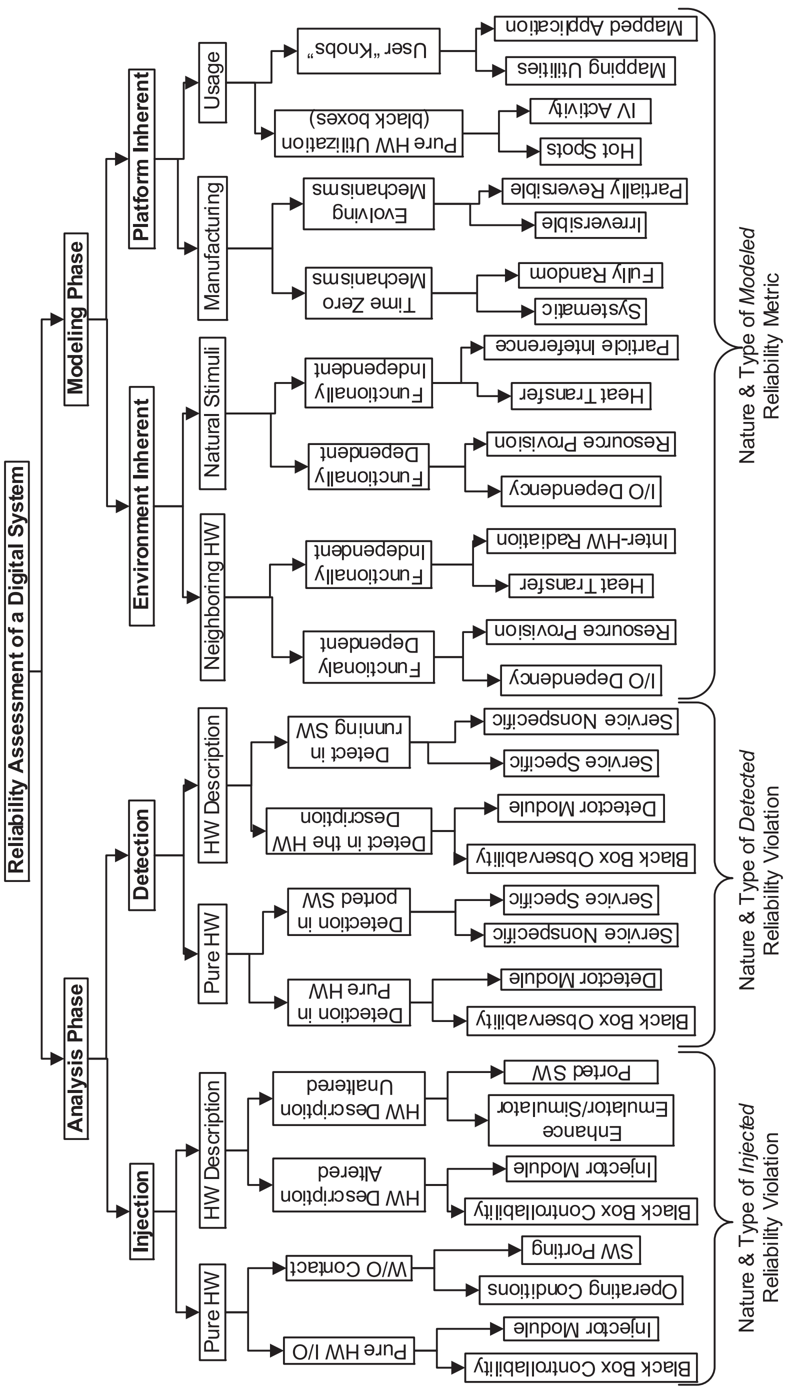


Fig. 13. The entire framework of our classification on the analysis and modeling of digital system reliability. All intermediate classification frameworks of Figures 6 through 10 are included. State-of-the-art works have been classified in the leaves of this graph by verifying the criteria of each split.

REFERENCES

A. Andrei et al. 2004. Reliability study of AlTi/TiW, polysilicon and ohmic contacts for piezoresistive pressure sensors applications. In *Proceedings of IEEE Sensors*. 1125–1128. DOI:http://dx.doi.org/10.1109/ICSENS.2004.1426374

J. Arlat et al. 2003. Comparison of physical and software-implemented fault injection techniques. *IEEE Transactions on Computing* 52, 9, 1115–1133. DOI:http://dx.doi.org/10.1109/TC.2003.1228509

A. Avizienis et al. 2004. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing* 1, 1, 11–33. DOI:http://dx.doi.org/10.1109/TDSC.2004.2

R. C. Baumann. 2005. Radiation-induced soft errors in advanced semiconductor technologies. *IEEE Transactions on Device and Materials Reliability* 5, 3, 305–316. DOI:http://dx.doi.org/10.1109/TDMR.2005.853449

D. Bekiaris, A. Papanikolaou, C. Papameletis, D. Soudris, G. Economakos, and K. Pekmestzi. 2011. A temperature-aware time-dependent dielectric breakdown analysis framework. In *Proceedings of the 20th International Conference on Integrated Circuit and System Design: Power and Timing Modeling, Optimization and Simulation (PATMOS'10)*. Springer-Verlag, Grenoble, France, 73–83. http://dl.acm.org/citation.cfm?id=1950238.1950248.

R. Bell. 2006. Introduction to IEC 61508. In *Proceedings of the 10th Australian Workshop on Safety Critical Systems and Software, Vol. 55 (SCS'05)*. Australian Computer Society, Darlinghurst, Australia, 3–12.

S. Benecke et al. 2012. Energy harvesting on its way to a reliable and green micro energy source. In *Proceedings of Electronics Goes Green 2012+ (EGG), 2012*. 1–8.

A. Benso et al. 2010. *Fault Injection Techniques and Tools for Embedded Systems Reliability Evaluation*. Springer.

D. Bharathan et al. 2008. An assessment of air cooling for use with automotive power electronics. In *Proceedings of the 11th Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems, 2008*. 37–43. DOI:http://dx.doi.org/10.1109/ITHERM.2008.4544251

A. Biswas et al. 2005. Computing architectural vulnerability factors for address-based structures. In *Proceedings of the 32nd International Symposium on Computer Architecture, 2005 (ISCA'05)*. 532–543. DOI:http://dx.doi.org/10.1109/ISCA.2005.18

S. Borkar et al. 2011. The future of microprocessors. *Communications of the ACM* 54, 5, 67–77. DOI:http://dx.doi.org/10.1145/1941487.1941507

J. W. Bossung. 1977. Projection printing characterization. In *Proceedings of SPIE, Vol. 100: Developments in Semiconductor Microlithography II*. 80–84. DOI:http://dx.doi.org/10.1117/12.955357

D. Burger et al. 1997. The SimpleScalar tool set, v. 2.0. *SIGARCH Computer Architecture News* 25, 3, 13–25. DOI:http://dx.doi.org/10.1145/268806.268810

E. Buturla. 1991. The use of TCAD in semiconductor technology development. In *Proceedings of the IEEE 1991 Custom Integrated Circuits Conference*. 23.1/1–23.1/7. DOI:http://dx.doi.org/10.1109/CICC.1991.164016

N. P. Carter et al. 2010. Design techniques for cross-layer resilience. In *Proceedings of the Design, Automation, and Test in Europe Conference and Exhibition (DATE), 2010*. 1023–1028. DOI:http://dx.doi.org/10.1109/DATE.2010.5456960

F. Chen et al. 2011. Realistic simulation and experimental validation of adjacent-channel interference in planning of industrial wireless networks. In *Proceedings of the 8th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*. ACM, New York, NY, 97–104. DOI:http://dx.doi.org/10.1145/2069063.2069080

H. Cho et al. 2012. ERSA: Error resilient system architecture for probabilistic applications. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 31, 4, 546–558. DOI:http://dx.doi.org/10.1109/TCAD.2011.2179038

P. Civera et al. 2001. Exploiting circuit emulation for fast hardness evaluation. *IEEE Transactions on Nuclear Science* 48, 6, 2210–2216. DOI:http://dx.doi.org/10.1109/23.983197

C. E. Ebeling. 2009. *An Introduction to Reliability and Maintainability Engineering*. Waveland Press.

A. Evans et al. 2012a. Case study of SEU effects in a network processor. In *Proceedings of the IEEE Workshop on Silicon Errors in Logic–System Effects (SELSE)*.

A. Evans et al. 2012b. RIIF: Reliability information interchange format. In *Proceedings of the 2012 IEEE 18th International On-Line Testing Symposium (IOLTS)*. 103–108. DOI:http://dx.doi.org/10.1109/IOLTS.2012.6313849

Y. Fei et al. 2007. Microarchitectural support for program code integrity monitoring in application-specific instruction set processors. In *Proceedings of the Conference on Design, Automation, and Test in Europe (DATE'07)*. 815–820. http://dl.acm.org/citation.cfm?id=1266366.1266540

D. D. Gajski et al. 1983. New VLSI tools. *Computer* 16, 12, 11–14. DOI:http://dx.doi.org/10.1109/MC.1983.1654264

A. Geist et al. 2002. *Development of Naturally Fault Tolerant Algorithms for Computing on 100,000 Processors*. Technical Report. Oak Ridge National Laboratory.

R. Geist et al. 1990. Reliability estimation of fault-tolerant systems: Tools and techniques. *Computer* 23, 7, 52–61. DOI:http://dx.doi.org/10.1109/2.56852

M. S. Gordon et al. 2004. Measurement of the flux and energy spectrum of cosmic-ray induced neutrons on the ground. *IEEE Transactions on Nuclear Science* 51, 6, 3427–3434. DOI:http://dx.doi.org/10.1109/TNS.2004.839134

T. Grasser et al. 2011. The paradigm shift in understanding the bias temperature instability: From reaction-diffusion to switching oxide traps. *IEEE Transactions on Electron Devices* 58, 11, 3652–3666. DOI:http://dx.doi.org/10.1109/TED.2011.2164543

R. L. Greenwell et al. 2011. SOI-based integrated circuits for high-temperature power electronics applications. In *Proceedings of the 26th Annual IEEE Applied Power Conference and Exposition*. 836–843. DOI:http://dx.doi.org/10.1109/APEC.2011.5744692

P. Gupta et al. 2004. Toward a systematic-variation aware timing methodology. In *Proceedings of the 41st Design Automation Conference, 2004*. 321–326.

M.-C. Hsueh et al. 1997. Fault injection techniques and tools. *Computer* 30, 4, 75–82. DOI:http://dx.doi.org/10.1109/2.585157

IEEE. 2000. *1076-2000: IEEE Standard VHDL Language Reference Manual*. i–290. DOI:http://dx.doi.org/10.1109/IEEESTD.2000.92297

IEEE. 2012a. *1800-2012: IEEE Standard for SystemVerilog–Unified Hardware Design, Specification, and Verification Language*. 1–1304.

IEEE. 2012b. *1666-2011: IEEE Standard for Standard SystemC Language Reference Manual*. 1–638. DOI:http://dx.doi.org/10.1109/IEEESTD.2012.6134619

ISO 26262-1. 2011. ISO 26262-1:2011: Road Vehicles–Functional Safety–Part 1: Vocabulary. Retrieved December 26, 2014, from http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=43464

F. Jalali et al. 2012. Error control schemes in solar energy harvesting wireless sensor networks. In *Proceedings of the 2012 International Symposium on Communications and Information Technologies (ISCIT)*. 979–984. DOI:http://dx.doi.org/10.1109/ISCIT.2012.6381047

B. Kaczer et al. 2010. Origin of NBTI variability in deeply scaled pFETs. In *Proceedings of the 2010 IEEE International Reliability Physics Symposium (IRPS)*. 26–32. DOI:http://dx.doi.org/10.1109/IRPS.2010.5488856

K. C. Kapurl et al. 1977. *Reliability in Engineering Design*. John Wiley & Sons.

J. Karlsson et al. 1995. Application of three physical fault injection techniques to the experimental assessment of the Mars architecture. In *Proceedings of the 5th IFIP International Working Conference on Dependable Computing for Critical Applications*. IEEE, Los Alamitos, CA, 267–287.

S. Kim et al. 2007. Power deregulation: Eliminating off-chip voltage regulation circuitry from embedded systems. In *Proceedings of the 5th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*. 105–110.

W.-Y. Kim et al. 2009. Evergreen: A fault-tolerant application streaming technique. In *Proceedings of the 11th International Conference on Advanced Communication Technology*. IEEE, Los Alamitos, CA, 2302–2307. http://dl.acm.org/citation.cfm?id=1701655.1701814

Y. Kim et al. 2011. Automated di/dt stressmark generation for microprocessor power delivery networks. In *Proceedings of the 2011 International Symposium on Low Power Electronics and Design (ISLPED)*. 253–258. DOI:http://dx.doi.org/10.1109/ISLPED.2011.5993645

M. Kimura. 1999. Field and temperature acceleration model for time-dependent dielectric breakdown. *IEEE Transactions on Electronic Devices* 46, 1, 220–229. DOI:http://dx.doi.org/10.1109/16.737462

G. A. Klutke et al. 2003. A critical look at the bathtub curve. *IEEE Transactions on Reliability* 52, 1, 125–129. DOI:http://dx.doi.org/10.1109/TR.2002.804492

M. Koganemaru et al. 2008. Evaluation of stress-induced effect on electronic characteristics of nMOSFETs using mechanical stress simulation and drift-diffusion device simulation. In *Proceedings of the 2nd Electronics System-Integration Technology Conference, 2008 (ESTC'08)*. 839–844. DOI:http://dx.doi.org/10.1109/ESTC.2008.4684461

H. Kopetz et al. 1989. Distributed fault-tolerant real-time systems: The Mars approach. *IEEE Micro* 9, 1, 16. DOI:http://dx.doi.org/10.1109/40.16792

A. Kritikakou et al. 2013. A systematic approach to classify design-time global scheduling techniques. *ACM Computing Surveys* 45, 2, Article No. 14. DOI:http://dx.doi.org/10.1145/2431211.2431213

A. Kumar et al. 2009. SRAM supply voltage scaling: A reliability perspective. In *Proceedings of the Conference on Quality of Electronic Design, 2009 (ISQED'09)*. 782–787. DOI:http://dx.doi.org/10.1109/ISQED.2009.4810392

M.-N. Li et al. 2009. Accurate microarchitecture-level fault modeling for studying hardware faults. In *Proceedings of the IEEE 15th International Symposium on High Performance Computer Architecture, 2009 (HPCA'09)*. 105–116. DOI:http://dx.doi.org/10.1109/HPCA.2009.4798242

M. R. Lyu. 2007. Software reliability engineering: A roadmap. In *Proceedings of the 2007 Future of Software Engineering Conference (FOSE'07)*. IEEE, Los Alamitos, CA, 153–170. DOI:http://dx.doi.org/10.1109/FOSE.2007.24

J. Ma et al. 2011. Layout-aware critical path delay test under maximum power supply noise effects. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 30, 12, 1923–1934. DOI:http://dx.doi.org/10.1109/TCAD.2011.2163159

S. Mao et al. 2010. Hardware support for secure processing in embedded systems. *IEEE Transactions on Computers* 59, 6, 847–854. DOI:http://dx.doi.org/10.1109/TC.2010.32

R. J. Martínez et al. 1999. Experimental validation of high-speed fault-tolerant systems using physical fault injection. In *Proceedings of the Conference on Dependable Computing for Critical Applications*. IEEE, Los Alamitos, CA, 249–265.

J. W. McPherson. 2006. Reliability challenges for 45nm and beyond. In *Proceedings of the 43rd Annual Design Automation Conference (DAC'06)*. ACM, New York, NY, 176–181. DOI:http://dx.doi.org/10.1145/1146909.1146959

A. Meixner et al. 2008. Argus: Low-cost, comprehensive error detection in simple cores. *IEEE Micro* 28, 1, 52–59. DOI:http://dx.doi.org/10.1109/MM.2008.3

S. Mukherjee. 2008. *Architecture Design for Soft Errors*. Morgan Kaufmann.

S. Mukherjee et al. 2005. The soft error problem: An architectural perspective. In *Proceedings of the 11th International Symposium on High-Performance Computer Architecture*. IEEE, Los Alamitos, CA, 243–247. DOI:http://dx.doi.org/10.1109/HPCA.2005.37

L. W. Nagel et al. 1973. *SPICE (Simulation Program with Integrated Circuit Emphasis)*. Technical Report UCB/ERL M382. EECS Department, University of California, Berkeley. Retrieved December 26, 2014, from http://www.eecs.berkeley.edu/Pubs/TechRpts/1973/22871.html.

R. Naseer et al. 2007. Critical charge characterization for soft error rate modeling in 90nm SRAM. In *Proceedings of the IEEE International Symposium on Circuits and Systems, 2007 (ISCAS'07)*. 1879–1882. DOI:http://dx.doi.org/10.1109/ISCAS.2007.378282

Z. Peng et al. 2009. Impact of humidity on dielectric charging in RF MEMS capacitive switches. *IEEE Microwave and Wireless Components Letters* 19, 5, 299–301. DOI:http://dx.doi.org/10.1109/LMWC.2009.2017595

R. G. Ragel et al. 2006. IMPRES: Integrated monitoring for processor reliability and security. In *Proceedings of the 2006 43rd ACM/IEEE Design Automation Conference*. 502–505. DOI:http://dx.doi.org/10.1109/DAC.2006.229268

V. Raghunathan et al. 2005. Design considerations for solar energy harvesting wireless embedded systems. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks, 2005 (IPSN'05)*. 457–462. DOI:http://dx.doi.org/10.1109/IPSN.2005.1440973

V. J. Reddi et al. 2010. Predicting voltage droops using recurring program and microarchitectural event activity. *IEEE Micro* 30, 1, 110. DOI:http://dx.doi.org/10.1109/MM.2010.25

V. J. Reddi et al. 2011. Voltage noise in production processors. *IEEE Micro* 31, 1, 20–28. DOI:http://dx.doi.org/10.1109/MM.2010.104

F. Redmill. 2005. *An Introduction to the Safety Standard IEC 61508*. Technical Report. Centre for Software Reliability, Newcastle University, Newcastle upon Tyne, England. Available at http://www.csr.ncl.ac.uk/FELIX_Web/4B.IEC%2061508%20Intro.pdf.

P. Roche et al. 1999. Determination of key parameters for SEU occurrence using 3-D full cell SRAM simulations. *IEEE Transactions on Nuclear Science* 46, 6, 1354–1362. DOI:http://dx.doi.org/10.1109/23.819093

B. F. Romanescu et al. 2007. *VariaSim: Simulating Circuits and Systems in the Presence of Process Variability*. Technical Report 2007-3. Department of Electrical and Computer Engineering, Duke University, Durham, NC.

SAE International. 1996. Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment. Retrieved December 24, 2014, from http://standards.sae.org/arp4761/.

E. Seevinck et al. 1987. Static-noise margin analysis of MOS SRAM cells. *IEEE Journal of Solid-State Circuits* 22, 5, 748–754. DOI:http://dx.doi.org/10.1109/JSSC.1987.1052809

R. A. Shafik et al. 2008. SystemC-based minimum intrusive fault injection technique with improved fault representation. In *Proceedings of the 14th IEEE International On-Line Testing Symposium*. IEEE, Los Alamitos, CA, 99–104. DOI:http://dx.doi.org/10.1109/IOLTS.2008.25

K. Skadron et al. 2004. Temperature-aware microarchitecture: Modeling and implementation. *ACM Transactions on Architecture and Code Optimization* 1, 1, 94–125. DOI:http://dx.doi.org/10.1145/980152.980157

D. J. Sorin. 2009. *Fault Tolerant Computer Architecture*. Morgan & Claypool.

J. Srinivasan et al. 2004. The impact of technology scaling on lifetime reliability. In *Proceedings of the 2004 International Conference on Dependable Systems and Networks*. 177–186. DOI:http://dx.doi.org/10.1109/DSN.2004.1311888

C. Srivaree-ratana et al. 1998. Estimating all-terminal network reliability using a neural network. In *Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 5. 4734–4739. DOI:http://dx.doi.org/10.1109/ICSMC.1998.727600

M. Toledano-Luque et al. 2011. Response of a single trap to AC negative bias temperature stress. In *Proceedings of the 2011 IEEE International Reliability Physics Symposium (IRPS)*. 4A.2.1–4A.2.8. DOI:http://dx.doi.org/10.1109/IRPS.2011.5784501

L. Tomek et al. 1994. Reliability modeling of life-critical, real-time systems. *Proceedings of the IEEE* 82, 1, 108–121. DOI:http://dx.doi.org/10.1109/5.259430

K. Tyagi et al. 2011. Reliability of component based systems: A critical survey. *ACM SIGSOFT Softwware Engineering Notes* 36, 6, 1–6. DOI:http://dx.doi.org/10.1145/2047414.2047434

A. J. Tylka et al. 1997. CREME96: A revision of the cosmic ray effects on micro-electronics code. *IEEE Transactions on Nuclear Science* 44, 6, 2150–2160. DOI:http://dx.doi.org/10.1109/23.659030

A. Varga et al. 2008. An overview of the OMNeT++ simulation environment. In *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks, and Systems and Workshops*. Article No. 60.

R. Velazco et al. 1991. Heavy ion test results for the 68020 microprocessor and the 68882 coprocessor. In *Proceedings of the 1st European Conference on Radiation and Its Effects on Devices and Systems*. 445–449. DOI:http://dx.doi.org/10.1109/RADECS.1991.213557

G. S. Walia et al. 2006. Requirement error abstraction and classification: An empirical study. In *Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engin.* ACM, New York, NY, 336–345. DOI:http://dx.doi.org/10.1145/1159733.1159784

J. Wei et al. 2012. BLOCKWATCH: Leveraging similarity in parallel programs for error detection. In *Proceedings of the 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. 1–12. DOI:http://dx.doi.org/10.1109/DSN.2012.6263959

Y. Ye et al. 2011. Statistical modeling and simulation of threshold variation under random dopant fluctuations and line-edge roughness. *IEEE Transactions on VLSI Systems* 19, 6, 987–996. DOI:http://dx.doi.org/10.1109/TVLSI.2010.2043694

R. Yokoyama et al. 2008. Modeling and evaluation of supply reliability of microgrids including PV and wind power. In *Proceedings of the Power and Energy Society General Meeting*. 1–5. DOI:http://dx.doi.org/10.1109/PES.2008.4596651

H. Ziade et al. 2004. A survey on fault injection techniques. *International Arab Journal of Information Technology* 1, 2, 171–186.

Received November 2012; revised December 2013; accepted October 2014