Intermittent Failures in Hardware and Software

Article *in* Journal of Electronic Packaging, Transactions of the ASME · March 2014

DOI: 10.1115/1.4026639

CITATIONS
27

READS
7,761

3 authors:

Roozbeh Bakhshi
University of Maryland, College Park
17 PUBLICATIONS **116** CITATIONS
SEE PROFILE

Surya Kunche
Nordson Corporation
4 PUBLICATIONS **42** CITATIONS
SEE PROFILE

Michael Pecht
University of Maryland, College Park
1,456 PUBLICATIONS **30,331** CITATIONS
SEE PROFILE

Roozbeh Bakhshi[1]
CALCE Electronic Products and Systems Center,
Building 89, Room 1103,
University of Maryland,
College Park, MD 20742
e-mail: Roozbeh@calce.umd.edu

Surya Kunche
CALCE Electronic Products and Systems Center,
Building 89, Room 1103,
University of Maryland,
College Park, MD 20742
e-mail: ksurya@umd.edu

Michael Pecht
CALCE Electronic Products and Systems Center,
Building 89, Room 1103,
University of Maryland,
College Park, MD 20742
e-mail: pecht@calce.umd.edu

Intermittent Failures in Hardware and Software

Intermittent failures and no fault found (NFF) phenomena are a concern in electronic systems because of their unpredictable nature and irregular occurrence. They can impose significant costs for companies, damage the reputation of a company, or be catastrophic in systems such as nuclear plants or avionics. Intermittent failures in systems can be attributed to hardware failures or software failures. In order to diagnose and mitigate the intermittent failures in systems, the nature and the root cause of these failures have to be understood. In this paper we have reviewed the current literature concerning intermittent failures and have a comprehensive study on how these failures happen, how to detect them and how to mitigate them. [DOI: 10.1115/1.4026639]

1 Introduction

Intermittent failures are sporadic failures that are not being easily repeatable because of their complicated behavioral patterns. These are also sometimes referred to as "soft" failures, since they do not manifest themselves all the time and disappear in an unpredictable manner. In contrast, "hard" failures are permanent failures that occur over a period of time (or are sometimes instantaneous). They have a specific failure site (location of failure), mode (how the failure manifests itself), and mechanism, and there is no unpredictable recovery for the failed system. Since intermittent failures are not easily repeatable, it is more difficult to conduct a failure analysis for them, understand their root causes, or isolate their failure site than it is for permanent failures.

Intermittent failures are reported under the category of NFF, which means that a failure was observed in the system, but when the device was tested for it, a failure mode could not be identified or the failure could not be duplicated. Some terms that are used interchangeably with NFF are trouble not identified, no trouble found, cannot duplicate, and retest ok [1]. These failures are hard to identify or duplicate, although they are recurrent. Many factors could cause intermittent failures, such as process variation, manufacturing residuals, in-progress wear-out, and voltage and temperature fluctuations [1]. NFFs have availability consequences; for example, they can lead to mission aborts, flight and train delays, or cancellations and increase the downtime period of the system so reduce the availability [2].

A no failure found phenomena can have catastrophic consequences in sensitive industries such as avionics or transportation. For example, in a case of NFF, a potentially faulty electronic component returning to field and then again showing signs of failure during operation compromises the safety of the system [3]. This can be a result of identifying a NFF as a "false alarm," where a real failure exists in the system, but since nothing was found in maintenance tests, it is branded as a false alarm.

Intermittent failures can be very expensive for companies. When no failure appears in the diagnosis, the maintenance team has to run extra tests in order to identify the failure. For example, a disruption in a voice command system of an airplane can be initially diagnosed as a component malfunction while after running test on components the intermittent failure happen again and further tests show that the reason is a loose connection. These extra tests impose additional costs, which account for a large portion of the maintenance budget. In the case of NFFs, since the failure cannot be found, the costs are higher than in permanent failures. Maintenance can include hours of time and manpower trying to identify the failure without any success and sometimes with blind replacement of parts that are suspected of having a defect (without finding any specific problem), increasing the cost of inventory. For example, in 2001, F-16 plane customers spent $10 million to replace parts that were tested as NFF at the shop level [4]. In another case, the thick film integrated ignition module in Ford cars in the 1980s models led to a lawsuit and a settlement by Ford Motors Company [1]. A study by WDS in 2005 found that NFFs account for about 63% of the mobile phones that were returned to the manufacturer, costing the industry $4.5 dollars a year [5].

The consequences of intermittent failure and NFF phenomena emerges the need for a comprehensive study on the root causes of these failures. This paper describes hardware and software intermittent failures and includes the causes, diagnosis, and mitigation methodologies for intermittent failures.

2 Hardware Intermittent Failures

Hardware malfunctions can cause intermittent failure in electronic devices. This section discusses hardware components that experience intermittent failures, and their failure. The diagnosis and mitigation of hardware intermittent failures is also discussed.

2.1 Causes. Unlike permanent failures with persistent causes, in intermittent failures, the failure cause vanishes with changes in the working environment. An intermittent failure can lead to permanent failures in later stages of the life cycle. Based on the component, the intermittent failure can have different causes, including coefficient of thermal expansion (CTE) mismatch, corrosion, and electromigration. In this section, the different failure causes for various components are investigated.

2.1.1 Connectors and Wire Bond Failures. Intermittent failure in connectors and wire bonds can be caused by vibration, CTE mismatch, stress relaxation, and movement of the wiring harness

[1]Corresponding author.

Contributed by the Electronic and Photonic Packaging Division of ASME for publication in the JOURNAL OF ELECTRONIC PACKAGING. Manuscript received September 23, 2013; final manuscript received January 18, 2014; published online February 18, 2014. Assoc. Editor: Yi-Shao Lai.

because of the magnetic field [6], which happens in working cycles, increases the contact resistance to a point of failure, and then disappears in the next cycle [6]. For example, in new tin-plated contacts, no intermittent behavior is observed, but in later stages of the life cycle, there are intermittent failures which occur and then disappear in the next cycle [6].

Wire bond intermittent failures caused by CTE mismatch can dislodge poorly bonded wire bonds when the temperature changes. The wire bonds restore to their normal state once the stress caused by CTE mismatch is removed. Most of the time, the failure mode is an open circuit. In other cases, failure mode is a short circuit because of loose conducting material in the package [7].

Intermittent failures can also be caused by loose conducting material. Loose materials can be detected using screening on electronic products, including X-ray, vibration, and acoustic tests. These screening methods focus on the driving forces that cause loose short circuits and the effects of the short circuit on the component performance [8].

The molding process can introduce stresses which damage wire bonds in a way which is not visible and result in intermittent behavior [9]. This behavior is attributed to the gold ball bond weakening and lifting during the molding process, on the side of the package opposite to where the injection occurred during the molding process [9].

In addition to CTE mismatch and loose connections, corrosion is also responsible for intermittent failure of the electronics by degrading the electrical contact and can occur in the early stages of the life cycle. Corrosion on electronic parts can result in short circuits, an increase in the electrical resistance of the components, and the occurrence of intermittent or permanent failures. For example, electrochemical migration, which occurs between anodes and cathodes (and can be a reason behind NFF reports), is a corrosion related failure mechanism that forms dendrites between opposite biases and eventually results in short circuits. This process of dendrite formation is because of moisture condensation and depends on parameters such as the potential bias, cleanliness of the surfaces (lack of environmental contamination), and metals that are used (Sn, Pb, Cu, Ag, and Au are susceptible). These dendrites can result to intermittent behavior since the contact through them can be lost due to evaporation of moisture because of the local high temperatures created by the passing current [10,11]. Another example of the effects of corrosion can be seen in the corrosion of copper connectors that have layers of nickel and gold to protect against wear out, where intermittent failure behaviors occur under harsh environments (high relative humidity and H_2S presence) because of the formation of the corrosive component Cu_2S [12]. With vibration and temperature fluctuations, the conductive path between the power supply board and power convertor breaks and causes current and voltage spikes to the rest of the system as a result intermittent failures in contacts occur because of frequent connection and disconnections.

2.1.2 Digital Integrated Circuit (IC) Failures. With the evolution of integrated chip technology, devices are being scaled down rapidly over the time. This reduction in size has been making digital integrated circuits more susceptible to intermittent behavior. Intermittent failure modes in digital ICs have been categorized as stuck-at-zero or stuck-at-one, intermittent short or open, or timing violation [13].

Constantinescu [14] studied the causes of intermittent behavior in ICs, identifying various failure modes and causes. The study attributed voltage fluctuations across ICs to oxide layer breakdown. As ICs become smaller and smaller, the thickness of the oxide layers decreases. This leads to an increased risk of breakdown in oxide layer thickness. When this oxide layer breaks down, it creates a conducting path, thereby increasing the leakage current. The introduction of high k dielectrics reduces the rate of oxide breakdown, enabling thinner dielectrics. This, in turn, leads to timing violations. Timing violations occur when there is an excessive delay during signal propagation. Timing violation [13], are also caused by an increase in the resistance of interconnects due to thermal or mechanical loads, electromigration, and material diffusion. This increases the time for signal propagation, leading to timing delay. These failures manifest because of thermal and electrical loads and signal frequency variations. This also leads to intermittent shorting or opening caused by signal traces coming in contact with each other or losing contact.

Intermittent stuck-at-zero or stuck-at-one failures occur in the storage elements. Digital circuits have two states, zero or one, and a fault occurs when a particular signal is tied to either one or zero. This produces a logical error. Pan et al. [13] developed a metric for stuck-at-zero/stuck-at-one to characterize the vulnerability of a microprocessor to intermittent failures based on its structure. Intermittent failures have an active time and an inactive time. The active time is the time during which the failure is active and causes unexpected behavior, while the inactive time is the time when the failure does not affect performance. The length of this active time determines how the failure affects the various processes running on a microcontroller. One of the major causes of intermittent short or open failure mode is electromigration. Electromigration is caused by movement of metal atoms when electrons flow through them. This movement of atoms leads to short circuits. As IC chip technology becomes smaller, there is a reduction in wire widths. When currents passing through these wires are not proportionally scaled down, there is an increase in current densities, which make the ICs more vulnerable to electromigration than ICs with lower current densities [15].

The processes occurring in a processor are categorized as the instruction fetch and decode stage, the register fetch stage, the execute stage, the memory stage, and the write back stage. Kothawade et al. [16] found that time delay in processors can be attributed to factors such as temperature, process variations, negative bias temperature instability, temperature fluctuations, hot carrier injection, and voltage fluctuations. The many simultaneous factors causing faults present a challenge for processor designers, as they have to design built-in fault tolerance that is capable of mitigating the effect of time these faults.

2.2 Diagnosis. For detection of intermittent failures in hardware, the (failure mode, mechanism, and effect analysis (FMMEA) methodology [17], which is used to address permanent failures, can be used. The FMMEA steps are illustrated in Fig. 1.

The first two steps "define systems and identify elements and functions to be analyzed" and "identify potential failure modes," are more challenging for intermittent failures than for permanent failures. To define a system with a failure (step one), in a complex system consisting of several sub systems working together, a failure in one of the sub systems could affect another sub system and result in its failure. Finding the primary sub system with the initial failure is not an easy task, since intermittent failures are not detected when the system is tested for faults. However, the second

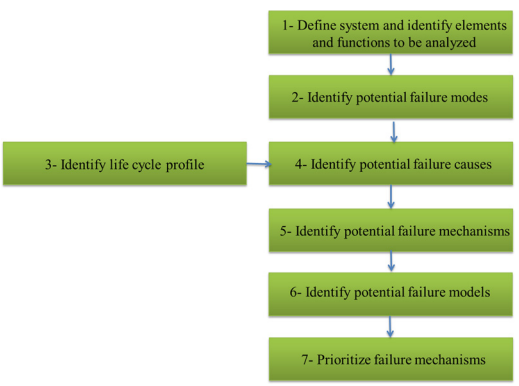

Fig. 1 FMMEA methodology [17]

step (identifying potential failure modes) is necessary because intermittent failures are categorized as NFFs due to the inability to define the correct failure mode. It is almost impossible to specify a certain list of methods that detect intermittent failures. Kirkland [18] suggests a variety of methods to detect intermittent failures in electronic devices, including signal looping, pattern looping, signal stepping, frequency deviation, pattern adjustment in critical areas, signal strength variation, current path duplication, measuring capacitance variations, Vcc adjustments, resistive or impedance rebounce, temperature change application, and noise dissimilarity testing. The details of these methods can be found in Ref. [18]. Using these methods can help identify failure modes, such as increased gate delays, degraded signals, increased leakage, and failing at high frequencies. A minimum set of conditions (like voltage drop threshold) for these possible failure modes needs to be set up to make the failure mode observable within a reasonable amount of time to be able to take preventive actions.

Another systematic method for analyzing intermittent failures is using a cause and effect diagram (fishbone diagram). An example of this diagram is presented in Fig. 2. This method defines the major effects, investigates the possible causes of each of the effects and, if possible, the causes of these causes, and then gives a perspective of all the possible factors which affect the system. It is a good method to analyze complex systems. For example, Qi et al. [19] investigated intermittent failures in plastic ball grid array packages using this method and narrowed down the possible causes of failure and identifying solder joints failure as the main cause of intermittent failure.

Another approach to investigate intermittent failures is introduced by Steadman et al. [20], which is primarily developed for the avionic systems in the cases of aging aircrafts. This method subjected the avionics system to thermal and vibrational loads, while simultaneously monitor the system for the faulty components. While this methodology could potentially help in reducing the re-occurrence of intermittent faults and the cost incurred due to these faults, the authors have not suggested what test conditions must be used and how these test conditions can replicate the actual usage conditions of a system.

2.3 Mitigation. ICs avoid failure by having failure tolerance built into them. Failure tolerance masks the occurrence of failures from the end user (i.e., it helps prevent end users from experiencing performance drops). For example, most processors are underclocked from their maximum capable clocking speed to tolerate the innate faults in circuits. ICs also have chip-level failure tolerance, such as error correcting codes, self-checking circuits, and hardware implemented check pointing and retries [14]. Three main methodologies to mitigate the intermittent behavior in ICs are: variable-latency, core frequency scaling, and thread migration.

During the normal operation of a processor, if the processor incurs more than the expected time to execute a process, it leads to time delay and timing violation. This is overcome by using dynamic instruction delaying, which is referred to as variable latency. Another approach to mitigate delay is core frequency scaling. As previously mentioned, most processors are underclocked to reduce the occurrence of delays. In core frequency scaling, the processor frequency is scaled down from its operating frequency to reduce the occurrence of timing violations. Frequency scaling directly affects the processor's speed; therefore, this approach has potential performance trade-off issues if employed frequently. In a processor, multiple threads are at work. If a specific thread encounters intermittent failures, all the relevant data of the thread saved in the faulty core are restored in the idle core through cache. Once it is move to an idle core, the thread is restarted. This process is referred to as thread migration.

The intermittent failures in avionic systems have been primarily attributed to solder joints and multilayer ribbon cables [28]. These failures are caused by loading conditions which activated the faults during operation of a system, but may disappear due to remelting, crack closing, or void filling during the operation of the system. While work has been done on mitigating intermittent behavior at the chip level [14], there is no method to mitigate solder joint and multilayer ribbon cable failures. Hence, research on solder joint or wire bond self-healing and material compliance need to be conducted.

3 Software Intermittent Failures

Software intermittent failures occur only when a certain set of conditions are met. For example, if the available memory and CPU processing power are both below a certain threshold due to other applications running on a computer, a desired program can exhibit failures due to insufficient resources. Software that is not robust because it has bugs can also fail whenever a user

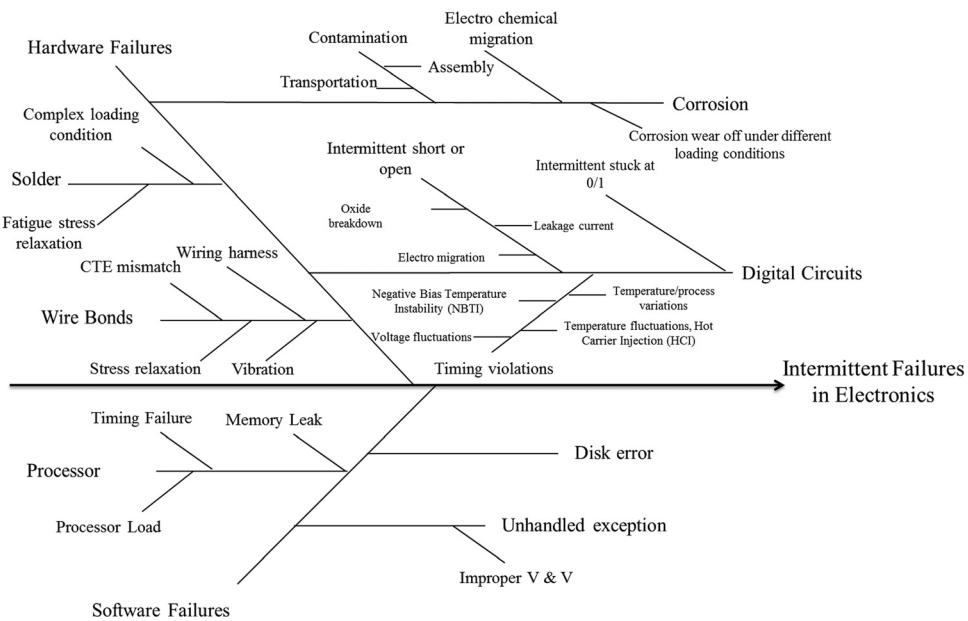

Fig. 2 Fishbone diagram for intermittent failures in hardware and software

encounters buggy parts of the program. In Secs. 3.1, 3.2 and 3.3, the causes of software intermittent behavior are investigated, and then the methods for identification and mitigation of these failures are described.

3.1 Causes. Even though software intermittent failures occur in most software-based systems, the end user may not experience performance drops. This is known as the observability of faults. The observability of software intermittent failures is affected by three main factors: processor speed, memory capacity, and processor load. A low processor speed increases the occurrence of intermittent failures, whereas with high processor speed, intermittent failures may not be observed. A high memory capacity reduces the observability of software intermittent failures. A change in the processor load affects the occurrence of intermittent failures. For example, a high load could cause a system to crash and restart.

The observability of an intermittent failure is different from the fault causes and factors affecting the observability of a fault can be used to mitigate the intermittent behavior by suppressing the causes. Gracia et al. [21] classified the causes of intermittent failures as error in memory, error in disk, and timing failures. Timing failures occur when process executions are delayed during processing or when the sequence of their execution is disturbed. For example, because process executions are time sensitive, the timing of various processes running simultaneously is critical during software executions and could lead to timing failures if not appropriately timed.

Memory leaks and memory errors occur because of improper memory allocation or de-allocation. This can happen when the memory footprint of a program increases (especially with prolonged usage), resulting in intermittent freezes and crashes. Software failure because of unhandled exceptions happen when an unexpected error occurs during execution and this error is not handled by the software. For example, when the software tries to divide a zero by one, an error is generated. If this error is not handled, it could lead to an intermittent failure. Disk error failures represent the anomalous behavior of software resulting from errors in physical disk drives. Concurrency-related failures occur when concurrent tasks are being executed, leading to heavy usage of the system.

3.2 Diagnosis. In most software, there are many different configurations possible. It is difficult, if not impossible, to test a product under all these configurations, and intermittent failures can occur on configurations which are not tested.

When testing for intermittent software failures, five techniques are currently used. First, deterministic replay debugging is where the engineers record all instructions up to the point where the system crashes and then replay that recording to determine the roots of the failure. Second, fuzzy testing uses random, invalid, or unexpected data and observes how the system reacts. Fuzzy-testing is most useful in detecting failures related to data corruption, memory leaks, assertions, and crashes. Third, high volume test automation (HVTA) is where there is an automated execution of a large number of tests cases. HVTA techniques have been shown to be particularly useful in finding failures such as buffer overruns, stack overflows, resource exhaustion, and timing-related errors. Fourth, load testing includes stress tests (testing at the operating condition limits until the system breaks) and volume tests (large tasks). Finally, disturbance tests are where the engineers disrupt the normal operation of the system by introducing physical failures, such as by unplugging the power cord.

While testing for intermittent behavior, there are various other factors that need to be considered. For example hardware configurations play a major role in causing intermittent software failure. Syed et al. [22] observed that the software testing on particular software in different hardware configurations provided significantly different numbers of intermittent failures based on the hardware configuration. For example, parameters such as processor speed, memory, hard drive capacity, and processor load led to variation in the number of intermittent failures observed. Wei et al. [23] developed a test methodology to inject faults at the hardware architecture level to understand the effect of hardware intermittent failures on software failures. The authors identified the signal anomalies affecting the software execution in the architecture of a processor and studied the effect of those anomalies on software execution. This work correlates software failures with their hardware causes, and hence, helps in diagnosis.

3.3 Mitigation. The mitigation of software faults is referred to as fault tolerance. The aim of fault mitigation is to prevent unexpected outputs and control errors. Fault tolerance is vital in systems where observable faults could be disastrous. Anderson et al. [24] discussed the main phases that constitute fault mitigation. The phases of this process are error detection, damage assessment, and error recovery. Error detection is used to identify the source of intermittent faults, while damage assessment assesses the extent of damage to the normal operation of the system. Once the nature of the fault is clearly identified, the next phase, error recovery, involves the mitigation of these faults. This stage focuses on avoiding the trickle down of the fault to the end user. There are three widely used techniques for error recovery: recovery block, n version software, and self-checking software [25].

Recovery blocks were originally proposed by Randell [26]. They prevent faults in software components from affecting functionality at the system level. In this approach, results from various sequences in a software component are verified by adjudicator software. Each of the outputs of the software component needs to pass an acceptance test by the adjudicator.

N version software emulates the concept of redundancy along with voting [27]. Here, for each task that software needs to perform, n components of the software modules perform the same task. The results from each of these n software components are polled to check for the most frequently occurring result in the n components. This methodology eliminates the chances of common mode failures.

Self-checking software [28] detects the occurrence of software errors, locates and identifies the causes, and stops the propagation of errors. To implement a self-checking software, a system needs to monitor functional aspects of the process and the data. Functional monitoring checks for infinite loops and incorrect loop terminations in a software program, while data monitoring checks the integrity of defined data structures in software.

4 Conclusion

Intermittent failures are difficult to diagnose because when they are investigated, no faults can be identified. In this paper, we addressed this problem for both hardware and software failures, and investigated causes, methods of diagnosis, and mitigation solution. For hardware failures, a fish bone diagram is a useful way to investigate the possible causes behind the intermittent behavior, but for investigating the intermittent behavior at system level, the FMMEA methodology helps to detect and analyze these behaviors. Because of the nature of these failures, their occurrence is not apparent during testing; hence, health monitoring can help reveal the cause of these failures during operation. Following FMMEA and using a fishbone diagram addresses the issue of no failure found. Further research in exploring the possibility of using the healing properties of solder material utilizing their viscoelastic properties could help mitigate these failures over the lifetime of a system. Software intermittent behavior is affected by the hardware components they are dependent on, which makes it difficult to replicate failures in laboratory conditions or during testing. Therefore, fault tolerance and mitigation is necessary to build reliable software in safety critical systems. Self-checking software has been widely adopted in safety critical systems such as airplanes. But, to even build fault tolerance for software, the

fault causes must be understood. At the system level, where there is a combination of hardware and software intermittent behavior, a considerable amount of work needs to be done to mitigate these failures.

References

[1] Thomas, D. A., Ayers, K., and Pecht, M., 2002, "The 'Trouble Not Identified' Phenomenon in Automotive Electronics," Microelectron. Reliab., **42**(4–5), pp. 641–651.

[2] James, I., Lumbard, D., Willis, I., and Goble, J., 2003, "Investigating No Fault Found in the Aerospace Industry," Annual Reliability and Maintainability Symposium, Tampa, FL, January 27–30, pp. 441–446.

[3] Söderholm, P., 2007, "A System View of the No Fault Found (NFF) Phenomenon," Reliab. Eng. Syst. Saf., **92**(1), pp. 1–14.

[4] Steadman, B., Pombo, T., Madison, I., Shively, J., and Kirkland, L., 2002, "Reducing No Fault Found Using Statistical Processing and an Expert System," IEEE AUTOTESTCON 2002, Huntsville, AL, October 15–17, pp. 872–878.

[5] WDS White Paper, 2006, "No Fault Found Returns Cost the Mobile Industry $4.5 Billion per Year," http://www.wds.co/news/whitepapers/20060717/MediaBulletinNFF.pdf

[6] Maul, C., McBride, J. W., and Swingler, J., 2001, "Intermittency Phenomena in Electrical Connectors," IEEE Trans. Compon. Packag. Technol., **24**(3), pp. 370–377.

[7] Schafft, H. A., 1973, "Failure Analysis of Wire Bonds," 11th Annual Reliability Physics Symposium, Las Vegas, NV, April 3–5, pp. 98–104.

[8] McCullough, R. E., 1972, "Screening Techniques for Intermittent Shorts," 10th Annual Reliability Physics Symposium, Las Vegas, NV, April 5–7, pp. 19–22.

[9] Koch, T., Richliug, W., Whitlock, J., and Hall, D., 1986, "A Bond Failure Mechanism," 24th Annual Reliability Physics Symposium, Anaheim, CA, April 1–3, pp. 55–60.

[10] Minzari, D., Jellesen, M. S., Møller, P., and Ambat, R., 2011, "On the Electrochemical Migration Mechanism of Tin in Electronics," Corros. Sci., **53**(10), pp. 3366–3379.

[11] Minzari, D., Grumsen, F. B., Jellesen, M. S., Møller, P., and Ambat, R., 2011, "Electrochemical Migration of Tin in Electronics and Microstructure of the Dendrites," Corros. Sci., **53**(5), pp. 1659–1669.

[12] Reid, M., Punch, J., Grace, G., Garfias, L. F., and Belochapkine, S., 2006, "Corrosion Resistance of Copper-Coated Contacts," J. Electrochem. Soc., **153**(12), pp. B513–B517.

[13] Pan, S., Hu, Y., and Li, X., 2010, "IVF: Characterizing the Vulnerability of Microprocessor Structures to Intermittent Faults," Design, Automation & Test in Europe Conference Exhibition (DATE), Dresden, Germany, March 8–12, pp. 238–243.

[14] Constantinescu, C., 2008, "Intermittent Faults and Effects on Reliability of Integrated Circuits," Annual Reliability and Maintainability Symposium (RAMS 2008), Las Vegas, NV, January 28–31, pp. 370–374.

[15] Blaauw, D. T., Oh, C., Zolotov, V., and Dasgupta, A., 2003, "Static Electromigration Analysis for On-Chip Signal Interconnects," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., **22**(1), pp. 39–48.

[16] Kothawade, S., Chakraborty, K., Roy, S., and Han, Y., 2012, "Analysis of Intermittent Timing Fault Vulnerability," Microelectron. Reliab., **52**(7), pp. 1515–1522.

[17] Mathew, S., Das, D., Rossenberger, R., and Pecht, M., 2008, "Failure Mechanisms Based Prognostics," International Conference on Prognostics and Health Management (PHM 2008), Denver, CO, October 6–9, pp. 1–6.

[18] Kirkland, L. V., 2011, "When Should Intermittent Failure Detection Routines be Part of the Legacy Re-Host TPS?," IEEE AUTOTESTCON 2011, Baltimore, MD, September 12–15, pp. 54–59.

[19] Qi, H., Ganesan, S., and Pecht, M., 2008, "No-Fault-Found and Intermittent Failures in Electronic Products," Microelectron. Reliab., **48**(5), pp. 663–674.

[20] Steadman, B., Berghout, F., Olsen, N., and Sorensen, B., 2008, "Intermittent Fault Detection and Isolation System," IEEE AUTOTESTCON 2008, Salt Lake City, UT, September 8–11, pp. 37–40.

[21] Gracia, J., Saiz, L., Baraza, J. C., Gil, D., and Gil, P., 2008, "Analysis of the Influence of Intermittent Faults in a Microcontroller," 11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS 2008), Bratislava, Slovakia, April 16–18, pp. 80–85.

[22] Syed, R. A., Robinson, B., and Williams, L., 2010, "Does Hardware Configuration and Processor Load Impact Software Fault Observability?," Third International Conference on Software Testing, Verification and Validation (ICST), Paris, April 6–10, pp. 285–294.

[23] Wei, J., Rashid, L., Pattabiraman, K., and Gopalakrishnan, S., 2011, "Comparing the Effects of Intermittent and Transient Hardware Faults on Programs," 2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W), Hong Kong, June 27–30, pp. 53–58.

[24] Anderson, T., and Knight, J. C., 1983, "A Framework for Software Fault Tolerance in Real-Time Systems," IEEE Trans. Software Eng., **SE-9**(3), pp. 355–364.

[25] Lyu, M. R., 1995, *Software Fault Tolerance*, John Wiley & Sons, Inc., New York.

[26] Randell, B., 1975, "System Structure for Software Fault Tolerance," International Conference on Reliable Software, Los Angeles, CA, April 21–23, pp. 437–449.

[27] Avizienis, A., 1985, "The N-Version Approach to Fault-Tolerant Software," IEEE Trans. Software Eng., **SE-11**(12), pp. 1491–1501.

[28] Blum, M., Luby, M., and Rubinfeld, R., 1993, "Self-Testing/Correcting With Applications to Numerical Problems," J. Comput. Syst. Sci., **47**(3), pp. 549–595.