The Challenge of Detection and Diagnosis of Fugacious Hardware Faults in VLSI Designs

Jaime Espinosa, David de Andrés, Juan-Carlos Ruiz, and Pedro Gil

Fault-Tolerant Systems Group (GSTF), Instituto de Aplicaciones de las TIC Avanzadas (ITACA)
Universitat Politècnica de València (UPV), Campus de Vera s/n, 46022, Valencia, Spain
Phone: +34 96 3877007 Ext {75774, 75752, 79707}, Fax: +34 96 3877579
{jaiesgar,ddandres,jcruizg,pgil}@disca.upv.es
http://www.stf.webs.upv.es

Abstract. Current integration scales are increasing the number and types of faults that embedded systems must face. Traditional approaches focus on dealing with those transient and permanent faults that impact the state or output of systems, whereas little research has targeted those faults being logically, electrically or temporally masked -which we have named fugacious. A fast detection and precise diagnosis of faults occurrence, even if the provided service is unaffected, could be of invaluable help to determine, for instance, that systems are currently under the influence of environmental disturbances like radiation, suffering from wear-out, or being affected by an intermittent fault. Upon detection, systems may react to adapt the deployed fault tolerance mechanisms to the diagnosed problem. This paper explores these ideas evaluating challenges and requirements involved, and provides an outline of potential techniques to be applied.

Keywords: Fault detection; transient faults; intermittent faults; permanent faults; fault diagnosis; VLSI design workflow

1 Introduction

Current embedded VLSI systems are widespread and operate in multitude of applications in different markets, ranging from life support, industrial control, or airborne electronics to consumer goods. It is unquestionable that the former require different degrees of fault tolerance, given the human lives or great investments at stake, but it is not so obvious to admit that unexpected failures in consumer products can undermine their success in the marketplace [1]. Hence, there is great interest in protecting equipment from eventual faults, which in turn involves providing a certain degree of service reliability over the whole lifetime. Specifically this relies on controlled operation of both software and hardware. While it is clear that potential programming bugs will affect the software behaviour, recent studies on complete systems highlight the disastrous impact which even transient faults happening in the hardware may have in the code execution of critical applications [2]. Therefore in order to achieve dependable devices it is no longer possible to preclude hardware implications from software design.

In the design stage of a product it is foreseeable an evolution in its operational state, from an ideal scenario to another posing several dependability threats. Since a set of specifications has to be met, a conservative approach is taken and security margins are applied to compensate for expected negative effects which may hinder correct service delivery. But that evolution can be no longer predicted accurately enough [3], leaving the only alternative to adapt the system to unexpected changes during its service lifetime. Hence, it is a growingly important requisite to create an information flow from environment to hardware and finally software. A major source of such sudden changes in a system is the occurrence of faults.

To explain why faults appear, there are a number of reasons to be mentioned. For instance, manufacturing capabilities have been evolving at a fast pace, bringing a new breadth of improvements to embedded systems in terms of logic density, processing speed and power consumption. However, those benefits become threats to the dependability of systems, causing higher temperatures, shorter timing budgets and lower noise margins which increase fault proneness. In addition, deep-submicron technologies have both decreased the probability of manufacturing defect-free devices, and increased the likelihood of problematic events originated by wear-out. Moreover, the susceptibility of extremely integrated electronics to α-particles and neutrons, arriving from outer space or radioactive materials grows steadily, yielding a non-negligible degree of so called *soft errors* [4], which affect temporarily the correctness of processing.

The mentioned faults can be classified in permanent or transient categories. Research in the field has focused mainly in tackling permanent faults, disregarding transient faults when their effect is not visible as errors in the captured data. For instance, transient faults with short activation times (percentage of time in which it is affecting the system relative to clock period), which have been shown difficult to detect by conventional means [5], may not produce incorrect outputs at once, but are a good indication of a problematic environment. We have named them *fugacious*. According to [6], out-of-range supply voltages, abnormal noise, temperature, etc. are triggers for such transient faults, which if repeated are called *intermittent* faults. Whether the final nature of the fault is transient or intermittent will depend on the wear-out conditions. For that reason we must make an effort to be able to detect and diagnose such types of faults, because these will provide valuable information when taking decisions for the evolution of the system. An example would be to change the data codification in a bus to a more robust scheme in the hardware, or to enable additional processing iterations or variable checks in the software, for redundancy purposes. Studies devoted to detection and diagnosis of fugacious faults are scarce or non-existent. However, certain known detection techniques could be applied to fugacious faults with limited success [24], since only a reduced period of time is monitored.

The contribution of this paper is based in 2 major points: (i) to identify and ponder the challenges of detection and diagnosis of fugacious faults in VLSI systems and (ii) to provide insight on methods and technologies to cover such challenges.

The rest of the paper is structured as follows. Section 2 justifies the importance of on-line detection, and underlines the difficulties of detecting transient and intermittent faults with short activation times. Furthermore it presents the different fault models while providing an overview on diagnosis of such faults. A set of methods and technologies is presented in Section 3 to cope with the task. Finally, Section 4 indicates the following actions to be taken and related issues.

2 The problem of Fast Fault Detection and Diagnosis

According to Avizienis [7] the basic criterion to catalogue faults in permanent or transient type is the persistence. This can however be an incomplete information to comprise the whole picture and thus, *activation reproducibility* is the concept introduced to better describe the observed situations. For permanent faults, different activation patterns lead to solid, hard faults when these are systematically reproducible or to elusive, soft faults when they are not. Depending on circumstances those soft faults can be intermittent in time. For transient faults, elusive activation is the most common but certain circumstances can likewise make them manifest intermittently.

Such differentiated activation patterns require tailored fault tolerant techniques of detection and diagnosis for dependability threats caused by faults and errors. In several situations including high availability or high performance systems, a concurrent detection (on-line) becomes critical. Next the existing scenario related to such concepts is explained.

2.1 On-line detection of faults and errors

In order to test proper development of the systems several methods have been described. From post-manufacture checking by means of test vectors or burn-in testing used to discard flawed units, to assigning slots of regular service time for test, for instance, many off-line techniques are currently employed. But the advantage of on-line detection is clear. A loose detection or notification latency, can have disastrous consequences in certain situations [8]. Besides, the longer a fault is present in the system without detection the higher the probability of facing a multiple fault situation. Provided that the latter is a problem of increased complexity we find justified interest in early detection.

There is long tradition in the dependability community to develop on-line error detectors. Typically, they are based in the use of special data codification or in the replication and comparison of outputs or state variables. But the relationship between a fault occurred at the processing network and an error manifested in the outputs or state variables is a limiting factor known as *observability* [9]. When the observability in an output is null for a given fault, no matter which input data combination was applied the fault will not show at the output. Likewise if more than one output is observable for that fault, multiple alterations would be detectable at those outputs. This can be specially important when using encoded circuits, where several properties describe different types of such circuits according to the consequences of a set of faults [10].

– *Fault-Secure.* Circuits where in presence of a fault either the outputs are correct or they are not a valid code word.
– *Self-Checking.* If for every fault of the fault models and for every input either the output is correct or it is detected as error.
– *Self-Testing.* Circuits in which for every fault of the fault set there is at least one possible input which causes the error to be detected.
– *Totally Self-Checking.* Circuits that are simultaneously Fault-Secure and Self-Testing.
– *Code-disjoint.* Circuits where a non-code word input generates a non-code output. This allows detection of erroneous inputs or cascading of blocks.

Therefore when employing codification it will be desirable to maximise the observability in order to achieve a good fault coverage.

There are 3 possible causes for fault filtering: electrical, logical or temporal. When dealing with permanent faults, temporal causes are discarded and due to the nature of hard faults, logical filtering can only be short in time. For those reasons any checking methodology will obtain positive results with hardly any misses meanwhile the observability is good enough.

Nevertheless, detection of transient or elusive and intermittent faults is not so straightforward. On the one hand, and according to field data from digital systems [11], transient faults have been shown to account for up to 80% of failures. These can be caused by several reasons as it is known. Among those reasons, the arrival of α-particles, protons or neutrons from radiation is one of the most studied and popular. If we pay attention to the evolution of transient fault duration produced by one of these particles impacting a CMOS node, the result is directly proportional to the feature size of the electronics [12]. However, the operational frequency of devices has not been following the historic monotonic growing trend, due to well known power dissipation issues. Consequently transients produced by radiated particles and charge build-ups are narrower and narrower compared to the clock periods. This paves the way to believe that although the number of faults affecting a system may be high, chances are these would not be easily captured by clock edges at the storage elements (heavy temporal filtering, see Figure 1) . The derivatives of this are that the moment a fault is detected many more could have already happened and the available time for reaction could be too short. Therefore for self-awareness purposes it is desirable to detect them.

On the other hand, intermittent errors caused as studied by Nightingale [13] a total of 39% of all hardware errors which, according to reports by Microsoft from 950.000 computers, induced a crash in the operating system. This gives a hint on the number of intermittent faults that can be happening in the system if we consider that not all of them will end in an operating system crash. Other

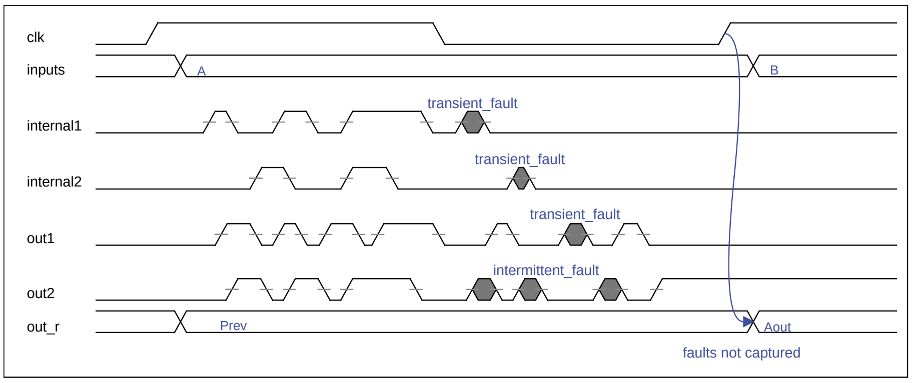

Fig. 1. Temporal filtering

examples of can be found in certain cruise control modules for vehicles [14] which hit a return rate of 96% due to undetected intermittent failures. These figures can be uneven depending on the context of operation since, as distinguished by Savir [15], random originated intermittent faults appear and disappear in an unpredictable fashion whether systematic intermittent faults evolution can be numerically characterized [16]. This enables a proper decision on the best moment to apply recovery actions to maximise availability. Such systematic intermittent faults start by small fluctuations which grow in time and intensity until their effect is severe [17]. In order to set focus on the considered problem, a description of fault models is required.

2.2 Considered fault models

According to the presented concept of activation time, and the activation reproducibility described earlier, we have established the name of *fugacious* faults to refer to a set of 3 different types of faults. The fugacious transient faults are defined as those which remain active less than a clock cycle of the system. Likewise, fugacious intermittent faults are those transient faults which activate at least twice in a clock period. Finally, permanent faults are active the whole time span of the clock period, that means for us a fault lasting more than one clock period will be considered permanently active.

2.3 Fault diagnosis

Multiple efforts have been conducted towards an effective diagnosis of different types of faults based on their activation reproducibility. As demonstrated in [18] there are important benefits for the Mission Time Degradation and Mean Time To Failure (MTTF) Degradation associated to correctly discriminating transient from permanent faults. It is clear that no equal treatment has to be given to both of them. For instance, transient faults will require no corrective action at all when hardware redundancy provides a voted fault tolerance. Disregarding the affected element for a certain period of time will negatively affect the dependability of the system. Furthermore, given the nature of intermittent faults and their proneness to become permanent, a proper distinction provides insight on the convenience to isolate or recover the functional unit. Intermittent faults diagnosis is a hot-topic in the field. An analytical model for a fault controller was presented in [19], using a thresholds-based α count methodology to discriminate transient from intermittent faults. Its Stochastic Activity Networks (SAN) analysis is specifically based on the time step, where transient faults last for less than one step and intermittent faults repeat their appearance in subsequent steps. Its drawbacks are it requires a long latency to discriminate, and infrastructure to detect and accumulate the respective faults. Other recent studies which also employ SAN with thresholds [20] applied to real systems only consider intermittent errors captured in state variables, which last more than one clock cycle.

In the case of fugacious faults, we take into account events of a quickly 'evanescent' nature where the capture and diagnosis procedure must have intrinsically low latency. It must be able to process two or more faults per cycle in order to discriminate an intermittent activation from a transient activation, avoiding a new constraint in the frequency requirements.

3 Solutions for detection and diagnosis

Our effort has been focused in two directions: determining an appropriate structure to detect and diagnose the set of faults we have previously presented and defining a procedure to apply such structure to the standard design flow.

3.1 Architecture of a faults detection and discrimination system

In every VLSI circuit we can find combinational stages separated by registers. Since the pursued goal is to have accurate and flawless computations, we will require 2 conditions:

- *To produce correct results.*
- *To sense any deviations in the datapath which may be out of reach by just checking registered values.*

The steps to take in order to reach these goals start by considering hardware replication and comparison. The large number of commercial systems utilizing such technique tells about the effectiveness of the approach at the expense of important amounts of hardware. The foremost advantage is quick on-line mitigation (when a voter is included), and usually there is no need to include voters in every stage but just in critical ones. Nevertheless, for detection and diagnosis a lighter, cheaper technique would enable the possibility to deploy detection to a larger number of partitions spread around the system. The use of codification may well fill the gap and combine with replication in a wise manner.

An interesting feature of codification is that systematic codes do not require to alter the original bits, thus alleviating the decoding of outputs in the datapath. In order to minimise speed penalty applied to outputs, this makes a great advantage [21]. Berger codes and parity groups are the most popular systematic codes and have been profoundly studied. In [22] conditions for fault secureness in parity predictors are derived. Furthermore [23] presents a generic optimisation technique for parity prediction functions, to achieve quick and small circuits.

The envisioned topology using codification would follow that in Figure 2. In it, a *Detection Block* would receive inputs directly from partition input registers, and also from outputs prior to registering. A properly selected codification could reduce block area and optimise the speed. This block would include thus a set of Commercial Off-The-Shelf (COTS) encoder and decoder which can be a single bit parity prediction/decoding pair in its simplest form.

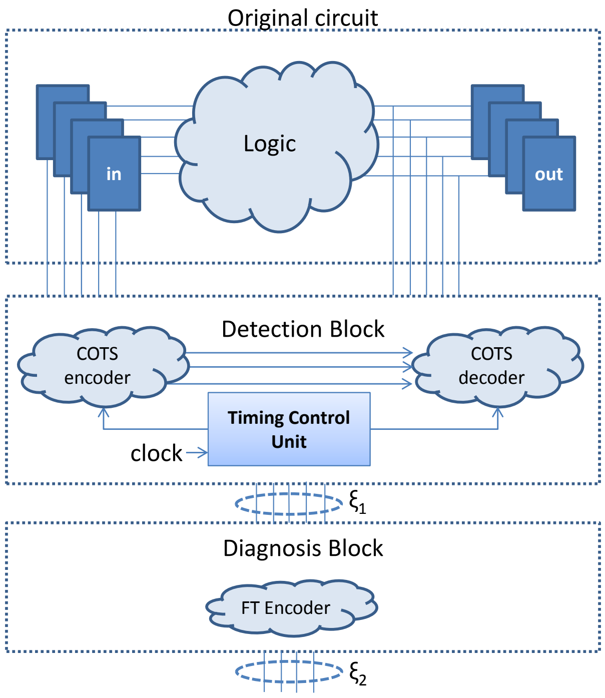

Fig. 2. Global scheme of the faults detection and diagnosis infrastructure. Timing Control Unit handles temporisation of Detection decoder

Is that enough to handle every type of fault? The answer is 'no'. Coding functions are effective techniques to detect permanent errors, or transient errors which are not time filtered. For effective detection of transient faults of a limited activation time (smaller than a clock period in general), additional elements are required. An example using triplication was presented in [24], where intermittent faults were not considered at all, and the sensing time was rather reduced. Online detection of intermittent faults has been previously devised in different ways. One proposed idea was to inject a carrier signal in the line under study and monitor the correct behaviour of it [25]. Again, the cost is rather high: an injector and receiver for the lines under analysis, plus extra wear-out due to increased switching of the lines. A cheaper detection can be achieved by monitoring those coded lines devoted to detection.

To avoid those shortcomings, an additional element included is the *Timing Control Unit* (TCU). Its function is to adjust the timings of detection elements with one goal in mind, i.e. to increase the *observation window.* The term refers to the percentage of the clock period when the lines under study are monitored for any potential faults. If we reduce the switching interval as opposed to the stability interval of the signals, we will have increased the observation window and thus the effectiveness of the detection (see Figure 3). The reason for preferring this method to the observation of a reduced period of time assuming equiprobable distribution of faults is clear, i. e. to gain in speed of detection.

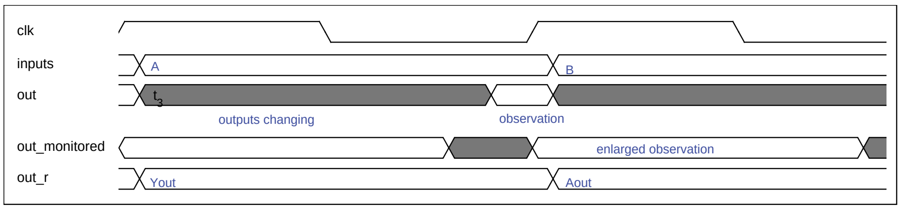

Fig. 3. Observation window enlarged by means of reducing period of signal switching

Finally once gathered, the detection information could be codified against faults using a code (ξ_1) and passed to a *Diagnosis Block*, where the same or a different code (ξ_2) can be used to notify the diagnosed output to a fault controller.

Inside the *Diagnosis Block*, inputs must be analysed and discriminated to offer 5 different output possibilities:

– *Transient fault.*
– *Permanent fault.*
– *Intermittent fault.*
– *No fault.*
– *Error in diagnosis infrastructure.*

To achieve the goal, the Diagnosis Block will be built using a fault-tolerant (FT) encoder designed to minimise resources taken. By providing all these different outputs and doing so in a fault tolerant codification, the most adequate decision will be enabled to be taken at the fault controller. Hence, smart reactions can be applied well in advance to an eventual collapse of fault tolerance infrastructure.

3.2 Workflow to apply in the proposed technique

In order to automate the process of deploying a detection and diagnosis infrastructure to a generic design block, a suggested procedure is shown in Figure 4. What is depicted is a typical semi-custom design flow for VLSI products, where the standard steps are on the left hand side. *Technology files* can represent a silicon foundry design kit or an FPGA manufacturer primitives library. Likewise, *Physical* element can be a layout file or a programming bitstream for an FPGA. On the right we find detail of 2 interventions in the flow.

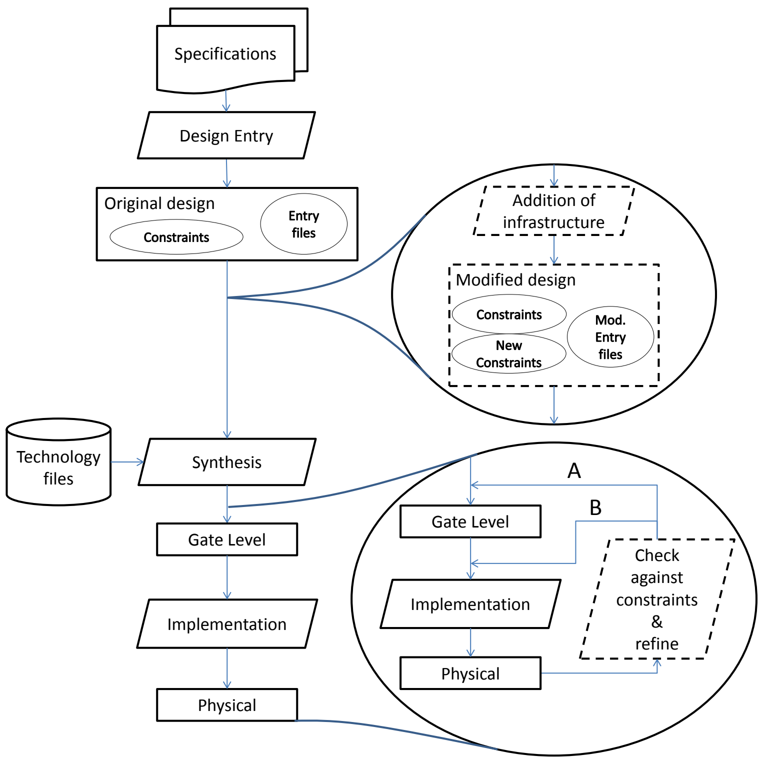

Fig. 4. Tools interaction

A first intervention comes before the *Synthesis* and after *Design Entry*. This step comprises an addition of required infrastructure in the Detection Block, i.e. the COTS components and Timing Control Unit. Entry files are modified as required and new timing constraints are generated for the TCU, to drive the remainder of the design flow.

A second intervention happens in a loop between Gate-level and Physical stages of the design. The purpose is to check timings against new constraints, mainly affecting the TCU, and refine the implementation in a loop by tweaking in one of the 2 re-entry points A or B. If path B is selected, a faster process will be obtained as a result, but deep knowledge of the underlying technology will be required and we will find a side effect of loss of portability. With path A, a more general solution will be obtained at the cost of speed of implementation.

The challenge in the integration of processes is derived from the difficulty to achieve the optimum observation window for the whole range of process variability. Other difficulties can come from the capabilities and restricted information offered by technology suppliers.

4 Ongoing Work

An initial implementation is currently under development, where an FPGA-based design flow has been chosen to support initial testing. Following the presented ideas, we have been able to develop first modification point working models. To reach optimal performance, we need first is to maximise the detection capabilities of the structure, both in area and time. This means achieving a high degree of observability at the check lines.

As for the second modification our efforts are devoted to achieve low performance penalty results and at the same time maximising the period in which lines are under surveillance. We need the least possibly intrusive system in order not to give in too much in exchange for detection. This is vital when applied to extreme performance demanding systems.

Last but not least, keeping the additional area small can be complex in certain circuits, if a powerful logic optimisation is not wisely applied. The upper limit will be that imposed by pure replication but this should be perfectly reducible without loosing much of the observability. An associated parameter to area increase is the power drain due to new infrastructure. As usual in engineering, specifications and market constraints drive the balance between detection and diagnosis capability and power/area/performance penalty.

Acknowledgements. This work has been funded by Spanish Ministry of Economy ARENES project (TIN2012-38308-C02-01).

References

1. Narayanan, V., Xie, Y.: Reliability concerns in embedded systems design. IEEE Computer **1**(39) (2006) 118–120
2. Hannius, O., Karlsson, J.: Impact of soft errors in a jet engine controller. In Ortmeier, F., Daniel, P., eds.: Computer Safety, Reliability, and Security. Volume 7612 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2012) 223–234
3. Borkar, S.: Designing reliable systems from unreliable components: the challenges of transistor variability and degradation. Micro, IEEE **25**(6) (nov.-dec. 2005) 10 – 16

4. JEDEC: Measurement and reporting of alpha particle and terrestrial cosmic ray-induced soft errors in semiconductor devices. In: JEDEC Standard JESD89A. JEDEC (2006)
5. Gracia-Moran, J., Gil-Tomas, D., Saiz-Adalid, L.J., Baraza, J.C., Gil-Vicente, P.J.: Experimental validation of a fault tolerant microcomputer system against intermittent faults. In: DSN. (2010) 413–418
6. Constantinescu, C.: Intermittent faults and effects on reliability of integrated circuits. In: Proceedings of the 2008 Annual Reliability and Maintainability Symposium, Washington, DC, USA, IEEE Computer Society (2008) 370–374
7. Avizienis, A., Laprie, J.C., Randell, B., Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. Dependable Secur. Comput. **1** (January 2004) 11–33
8. Johnson, C., Holloway, C.: The dangers of failure masking in fault-tolerant software: Aspects of a recent in-flight upset event. In: System Safety, 2007 2nd Institution of Engineering and Technology International Conference on. (oct. 2007) 60 –65
9. Bolchini, C., Salice, F., Sciuto, D.: Fault analysis for networks with concurrent error detection. IEEE Des. Test **15**(4) (October 1998) 66–74
10. Goessel, M., Ocheretny, V., Sogomonyan, E., Marienfeld, D.: New Methods of Concurrent Checking (Frontiers in Electronic Testing). 1 edn. Springer Publishing Company, Incorporated (2008)
11. Iyer, R.K., Rossetti, D.J.: A statistical load dependency model for cpu errors at slac. In: Fault-Tolerant Computing, 1995, ' Highlights from Twenty-Five Years'., Twenty-Fifth International Symposium on. (jun 1995) 373
12. Dodd, P.E., Shaneyfelt, M.R., Felix, J.A., Schwank, J.R.: Production and propagation of single-event transients in high-speed digital logic ics. IEEE Transactions on Nuclear Science **51** (December 2004) 3278–3284
13. Nightingale, E.B., Douceur, J.R., Orgovan, V.: Cycles, cells and platters: an empirical analysisof hardware failures on a million consumer pcs. In: Proceedings of the sixth conference on Computer systems. EuroSys '11, New York, NY, USA, ACM (2011) 343–356
14. Kimseng, K., Hoit, M., Tiwari, N., Pecht, M.: Physics-of-failure assessment of a cruise control module. Microelectronics Reliability **39**(10) (1999) 1423 – 1444
15. Savir, J.: Detection of single intermittent faults in sequential circuits. IEEE Trans. Comput. **29**(7) (jul 1980) 673–678
16. Correcher, A., Garcia, E., Morant, F., Quiles, E., Rodriguez, L.: Intermittent failure dynamics characterization. Reliability, IEEE Transactions on **61**(3) (sept. 2012) 649 –658
17. Sorensen, B., Kelly, G., Sajecki, A., Sorensen, P.: An analyzer for detecting intermittent faults in electronic devices. In: AUTOTESTCON '94. IEEE Systems Readiness Technology Conference. 'Cost Effective Support Into the Next Century', Conference Proceedings. (sep 1994) 417 –421
18. Sosnowski, J.: Transient fault tolerance in digital systems. IEEE Micro **14**(1) (feb 1994) 24–35
19. Bondavalli, A., Chiaradonna, S., Di Giandomenico, F., Grandoni, F.: Threshold-based mechanisms to discriminate transient from intermittent faults. IEEE Trans. Comput. **49**(3) (March 2000) 230–245
20. Rashid, L., Pattabiraman, K., Gopalakrishnan, S.: Intermittent hardware errors and recovery: modelling and evaluation. In: International Conference on Quantitative Evaluation of Systems (QEST). (2012)

21. Touba, N.A., McCluskey, E.J.: Logic synthesis of multilevel circuits with concurrent error detection. IEEE TRANS. CAD **16**(7) (1997) 783–789
22. Nicolaidis, M., Manich, S., Figueras, J.: Achieving fault secureness in parity prediction arithmetic operators: General conditions and implementations. In: Proceedings of the 1996 European conference on Design and Test. EDTC '96, Washington, DC, USA, IEEE Computer Society (1996) 186–
23. Ko, S.B., Lo, J.C.: Efficient realization of parity prediction functions in fpgas. J. Electron. Test. **20**(5) (oct 2004) 489–499
24. D'Angelo, S., Sechi, G.R., Metra, C.: Transient and permanent fault diagnosis for fpga-based tmr systems. In: Proceedings of the 14th International Symposium on Defect and Fault-Tolerance in VLSI Systems. DFT '99, Washington, DC, USA, IEEE Computer Society (1999) 330–338
25. Kim, C.: Detection and location of intermittent faults by monitoring carrier signal channel behavior of electrical interconnection system. In: Electric Ship Technologies Symposium, 2009. ESTS 2009. IEEE. (april 2009) 449 –455