Self-Test and Diagnosis for Self-Aware Systems
Survey for Design & Test Special Issue on "Self-Awareness in SoCs"

Kochte, Michael A.; Wunderlich, Hans-Joachim

IEEE Design & Test 13 October 2017

doi: https://doi.org/10.1109/MDAT.2017.2762903

Abstract: Self-awareness allows autonomous systems the dynamic adaptation to changing states of the hardware platform and the optimal usage of available computing resources. This demands concurrent, periodical, or on-demand monitoring and testing of the hardware structures to detect and classify deviations from the nominal behavior and appropriate reactions. This survey discusses suitable self-test, self-checking, and selfdiagnosis methods for the realization of self-awareness and presents two case studies in which such methods are applied at different levels.

Preprint

General Copyright Notice

This article may be used for research, teaching and private study purposes. Any substantial or systematic reproduction, re-distribution, re-selling, loan or sub-licensing, systematic supply or distribution in any form to anyone is expressly forbidden.

This is the author's "personal copy" of the final, accepted version of the paper published by IEEE.[1]

[1] **IEEE COPYRIGHT NOTICE**
©2017 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Self-Test and Diagnosis for Self-Aware Systems

Survey for Design & Test Special Issue on "Self-Awareness in SoCs"

Michael A. Kochte, Hans-Joachim Wunderlich
ITI, University of Stuttgart, Germany
kochte@iti.uni-stuttgart.de, wu@informatik.uni-stuttgart.de

***Abstract*—Self-awareness allows autonomous systems the dynamic adaptation to changing states of the hardware platform and the optimal usage of available computing resources. This demands concurrent, periodical, or on-demand monitoring and testing of the hardware structures to detect and classify deviations from the nominal behavior and appropriate reactions. This survey discusses suitable self-test, self-checking, and self-diagnosis methods for the realization of self-awareness and presents two case studies in which such methods are applied at different levels.**

Keywords—Self-test, diagnosis, health monitoring, fault management, on-chip infrastructure

I. Introduction

In addition to monitoring their environment, self-aware systems also need to monitor their own state and the capabilities of the underlying hardware platform [DJS15]. This comprises for instance timing or voltage margins, error rates and locations, or known faulty components. The knowledge of this *hardware fitness* is a requirement for autonomous and qualified adaptation to changing system states and graceful degradation in the observe-decide-act loop in self-aware systems [DJS15]. Examples for such adaptation are voltage or frequency calibration, use of available redundancies, or isolation of known faulty modules. This has become a necessity in autonomous safety-critical systems, found in automotive, industrial, or medical application domains.

To capture the state of the hardware platform, a self-aware system integrates instrumentation and sensors to monitor and to examine the state of its hardware structures. The employed instrumentation spans from design-for-test and diagnosis features, fault-tolerant design, and different types of monitors to the required control and access mechanisms that interconnect them.

This article describes the connection between self-awareness and self-test and self-diagnosis methods. The following section provides a brief definition of the used terminology and a classification of self-test methods. Section III details the underlying principles, followed by methods for autonomous diagnosis and self-test for health and aging assessment. Section VI discusses the effectiveness and costs of such methods. Section VII shows how the test and diagnostic information can be used for fault management. Finally, two case studies of self-aware systems are discussed in Section VIII.

II. Terminology and Classification

A. Terminology

A *defect* is an unintended deviation of the shape or structure of the material of the circuit, i.e. additional, missing, or wrong material at a location. This also comprises material impurities, transistor oxide damages, or wrong dotation. Defects result from manufacturing, imperfect materials, or stress and aging during operation.

Defects and their effects on circuit behavior have a continuous physical nature. A *fault* abstracts defective behavior in a structural model of the circuit to allow algorithmic analysis and processing. Examples for faults include stuck-at faults, bridging faults, or delay faults. A *fault model* is a set of faulty behaviors in a structural model. Faults can be modeled in structural models at different abstraction levels, from electrical, switch-level, gate or RT-level netlists up to the system structure.

A fault can be activated depending on the system state and the system environment, for example the temperature. The resulting effect is wrong information that propagates through the circuit, called an *error.* Errors such as single event upsets do not have a defect as root cause, but a transient event. If the error is not masked internally and causes a violation of the specification, a *failure* (malfunction) occurs. A failure of a module or component can be considered a temporary or permanent fault in the system comprising the component [ALRL04].

Test is an electrical experiment to check for the existence of defects, consisting of the application of input stimuli and the observation and comparison of responses of the circuit under test. Test generation can be based on a structural fault model or on exercising the functional behavior. In a structural fault model, the number of faults is typically linear to the number of components in the structural model and it is easy to quantify the fault coverage or test quality. In contrast, defining a meaningful coverage metric for functional testing is more difficult. *Self-test* refers to the autonomous test execution for fault detection by the system. *Self-checking*, in contrast, is a system's capability to detect errors during operation.

Diagnosis is the localization and classification of the root cause of failures by analyzing the test responses or performing additional diagnostic tests. The desired diagnostic resolution depends on the application and level of abstraction. High resolution increases the diagnosis cost but may allow to pin-point to the root cause at a lower level, for example a defective transistor or via even within a standard cell, compared to resolving only to the defective cell or interconnect wire. Lower

levels of resolution are acceptable if it is sufficient to identify the failing module.

A simple pass/fail test for a module can thus already provide sufficient diagnostic information for a system-level adaptation decision. As an example, we consider an electronic control unit (ECU) in the automotive context, which requires high dependability. In the field, an ECU may autonomously perform a power-on self-test and periodic tests of its components during the operation. Additionally, for critical parts a self-checking design can be implemented, for instance by error detecting codes in busses or memory arrays. Upon detection of an error in the field, tests are performed to diagnose the fault location at module level and decide on fault handling. Relevant failure data is stored in a log. In the workshop, an engineer performs a thorough test to determine whether the unit needs to be replaced. Failed ECUs are returned to the original equipment manufacturer (OEM) that tries to reproduce and diagnose the failure from chip to core and netlist level and further down to the location of the root cause in the chip material.

B. *Classification of Self-Test Methods*

A wide variety of different methods for self-test and self-checking exists, each with unique characteristics such as cost in hardware, energy, or performance, or also coverage of faults and fault detection latency, i.e. the time between the emergence of a fault and its detection. In contrast to offline testing, which typically uses external support and equipment, online self-test methods are applied in the field and the system is not shut-down.

The methods can be classified according to when a test or check takes place in the system as shown in Figure 1. Tests for self-awareness must be conducted online in the field, either in a non-concurrent or concurrent fashion. The non-concurrent activation is intrusive and typically alters the state of the module or component under test, which requires a suspension of the computation or service for the duration of the test. In multi-core systems, this disruption can be hidden by migrating running tasks to other available resources before test execution.

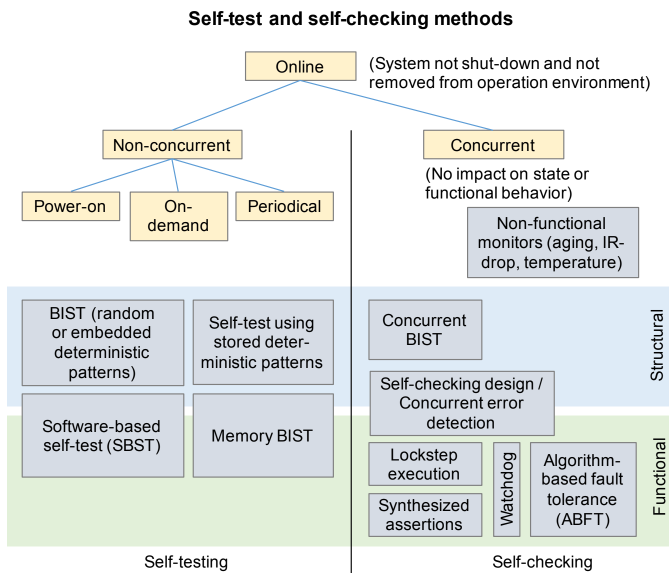

Figure 1. Classification of self-test and self-checking methods

Examples of non-concurrent self-tests are shown in the lower left part of the figure. These methods can be further partitioned into structure-oriented tests and functional tests.

The second class of methods are non-intrusive and active concurrent to the system operation. Such concurrent approaches monitor the system behavior for emergence of faults and errors, the divergence of functional behavior from the specification, or for non-functional observables such as temperature, radiation, or aging effects.

Self-aware systems require a combination of these methods to ensure that unexpected behavior is detected during operation and subsequently classified by test and diagnosis as illustrated in Figure 2.

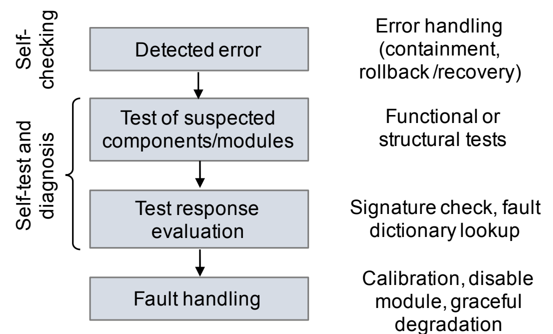

Figure 2. Error detection followed by test and diagnosis

III. Self-Test and Self-Checking Strategies

Non-concurrent self-test interrupts the operation of the component or part under test for the duration of the test. The quality of the test, i.e. its fault coverage, depends on the applied test stimuli. System constraints such as the access to design-for-test infrastructure or the acceptable test time and hardware overhead, restrict the way, type and quantity of applied stimuli.

In conventional built-in self-test (BIST) pseudo-random test stimuli are generated on chip by a linear feedback shift register or linear automaton and applied to the logic using scan chains [BardMS87]. Test responses are highly compacted over space and time into a small response signature. Depending on the design, it may be necessary to increase fault coverage by targeting random pattern resistant faults using weighted random patterns, the insertion of test points, or mixed-mode BIST using different LFSR polynomials or with embedded deterministic test patterns. While such approaches increase the hardware cost, they improve the test quality and can also reduce the required test time.

If the system possesses non-volatile memory, high-quality deterministic test stimuli can be stored in the system and applied via the scan chains on-demand or periodically [LCH05], reusing existing design-for-test infrastructure. This provides very high fault coverage also for delay faults and incurs the lowest test duration since only relevant patterns are applied. This idea has been adopted in [LMM08] for multi-core systems, also discussing the test setup including core isolation, test execution and state restoration after test completion. If non-volatile memory is not available, deterministic patterns can also be

recorded in scan chains with only marginal hardware cost [LTK16].

Self-tests are applied online and may not violate the functional power budget. Since conventional structural tests create a very high switching activity both during scanning of test data and test application, dedicated low-power BIST approaches have been developed. These comprise for instance test scheduling of modules, scan chain segmentation and scan clock control, or masking of switching activity [GNW10].

If the access to scan chains of a component is not possible, it can be exercised by applying functional test stimuli. If the stimuli are applied by a processor or programmable core in the system, the test is called software-based self-test (SBST) [PGSS10]. Apart from testing the processor, SBST is also applied to test caches, memory arrays, and IP cores attached to the processor or the system bus [KLCD02]. While the fault coverage achieved by SBST is typically lower than in a scan-based test, the application is less intrusive. The state of the tested component can be saved before and restored after the test via the functional access. Functional and high-quality structural tests are combined in [CMAB07] using a software-assisted access to the scan chain infrastructure by dedicated test instructions. This achieves very high fault coverage even for complex processor components with hidden state. SBST programs can be generated targeting different objectives such as low test time [GRP15], low power dissipation [ZhWu06], or detection of delay faults [RCS+14].

Memory arrays constitute a significant part of the hardware area of systems-on-chip and suffer proportionally to their area from defects in the read/write logic, the address decoder, and the actual memory cells. Apart from stuck-at faults, dedicated memory fault models comprise coupling faults, pattern-sensitive faults, and faults in the address decoder and read/write logic [GoVe90]. Memory test is performed by functional read and write accesses to the cells in a particular sequence, either by a processor or by memory BIST logic, which can also be micro-coded. Cyclic test sequences for memory restore the start state after test completion and enable transparent memory test [Nico92].

In safety-critical systems, emerging faults and resulting errors must be detected quickly to avoid hazardous system behavior. This requires a high frequency of periodic self-tests to reduce the fault detection latency or self-checking design to concurrently check the computational results for errors. The principal operation of self-checking design is illustrated in Figure 3. The behavior of the mission logic or parts thereof is predicted and checked by additional hardware. The type of the implemented check determines both the hardware overhead and coverage of errors.

The checking function can duplicate the computation and compare all outputs or predict a parity-based or arithmetic check-sum of the computation. Duplication using a diversified design offers higher error coverage since it avoids common mode errors. The area overhead is about 120% of the mission logic and only marginally higher compared to parity or other code based approaches [MiMc00].

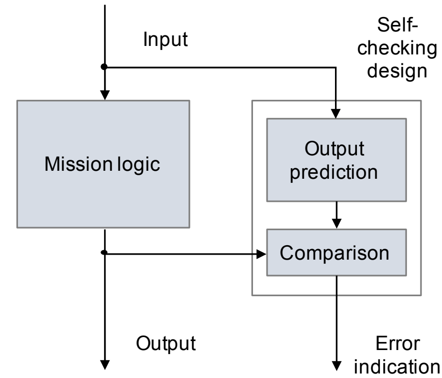

Figure 3. Self-checking design

This overhead can be reduced if the checking is reduced to a set of likely or critical faults. In so called concurrent BIST approaches, the checking logic is derived from a set of test patterns for such faults [KZW09, VoEf14].

The erroneous behavior for which a check is performed can also be derived from the design specification. Watchdog timers, control flow checking, or synthesized assertions are examples of concurrent functional error checks, that are implemented either in hardware or in software [MaMc88, OSM02, BCZ07]. The area and performance overhead depends on the number and types of critical behaviors to be checked during operation.

Self-checking the mission logic in hardware minimizes the error detection latency. If a higher latency is acceptable, the checking can also be implemented at a higher system level in software. General approaches exploit time or structural redundancy by duplicating and comparing the execution of threads. Examples are lockstep execution or redundant multi-threading with loose lockstepping. An overview of such software-implemented checking approaches is given in [GPA+11]. The checking can also be tailored to specific algorithms or operations employing algorithm-based fault tolerance (ABFT, HuAb84). In ABFT, the operands of an operation are encoded before computing, often by extension with a check-sum. The encoded results allow for detection of errors during the computation. The applicability of ABFT for self-awareness and the related performance overhead depends on the application, but can be, for instance, as low as 0.04% for preconditioned conjugate gradient solvers [SBK+15].

For memory elements, error detection relies on encoding and information redundancy. For arrays, the linear codes used to detect and correct a low number of errors incur only a small memory overhead. For distributed flip-flops or registers, error checking for timing and transient faults can be implemented by use of shadow latches for flip-flops with delayed clock and comparing the signal values [EKD+03, ZMM+06, Nico07]. In case of mismatches, correction is implemented by replaying the values from these shadow flip-flops. In [ImWu14], the high overhead of shadow latches is avoided by computing an error detecting and correcting address characteristic over the distributed registers. The resulting fault tolerant structure can also be reused for manufacturing test.

Self-checking needs to be complemented with periodic self-testing to prevent fault accumulation both in the mission logic and in the logic for self-checking and fault tolerance, since fault accumulation threatens the effectiveness of fault tolerance measures. Once an error or fault is detected, a self-aware system must locate its cause by diagnosis to determine an appropriate reaction.

IV. From Self-test to Self-diagnosis

Diagnosis locates and classifies the root cause of a failure. The information provided by diagnosis is required in self-aware systems to determine the appropriate system reaction, as for instance, adjusting the workload of a module to reduce the temperature, reducing the frequency of a module to tolerate timing marginalities, or disabling a module in case of an intolerable permanent fault. While volume diagnosis can rely on external test equipment, adaptive refinement of diagnostic resolution, and offline analysis of failure data, *self*-diagnosis has to be performed autonomously with the available test infrastructure on chip and the limited computational capabilities of the system for data analysis.

Diagnosis can be performed at different levels of granularity with different resolution. At module level, the passing or failing of a module test is a first diagnostic result that may be sufficient for a system level decision for online adaptation. In that case, a compacted test response signature is simply compared with the expected value. For higher resolution, the analysis of responses is more elaborate and typically performed offline.

Diagnosis methods can be classified as cause-effect or effect-cause approaches. In the first case, different causes, i.e. fault or defect candidates, are analyzed to determine their effects for comparison with the responses observed in the actual design under diagnosis, often employing a fault dictionary mapping effects to fault candidates. In the second case, the observed effects are analyzed to infer possible compatible candidates, often by back-tracing in the netlist.

The diagnostic quality increases with the quality of the test stimuli and the available test responses. Diagnosis with high resolution, e.g. at gate level, requires dedicated test stimuli in scan-based test [GMK91] or SBST [BSS+06]. Performing a high-resolution diagnosis online is not practical because of increased test time and memory requirements, large fault dictionaries, or high computational requirements of the diagnosis algorithms. On the other hand, failures in the field may depend on environmental conditions and system states for fault activation that are not known or cannot be reproduced in a failure analysis lab, causing a classification as “no-trouble-found”.

A compromise in self-aware systems is to support in-field diagnosis at moderate resolution and to make detailed diagnostic data available for later analysis at higher system level or offline analysis. This is achieved by design-for-diagnosis and tailored diagnosis algorithms. In [BeBl12], for instance, the size of the fault dictionary is minimized to allow online localization of a fault to the level required by the actual fault handling procedure. Built-in self-diagnosis architectures such as [ElWu10] support the storage of passing patterns and relevant failure data during a test session on chip, employing extreme compaction in space and time. The self-checking architecture in [KDB+15] stores a small amount of functional inputs and error signatures observed during operation to facilitate diagnosis and distinguish transient and intermittent failure causes.

V. BIST and Monitoring for Health Prediction

Testing the hardware for faults or checking it for errors allows the detection of faulty behavior that already violates the nominal specification. Health assessment and prediction, on the other hand, aims at detecting a deviation of the hardware behavior before it impacts the nominal function. This can be achieved by measuring the functional and non-functional factors that cause the deviation or by measuring the actual deviations with sufficient accuracy.

Non-functional observables, such as for example temperature, power-noise, or radiation, impact dependable system operation and must be monitored during operation. Environmental indicators, such as radiation levels, can be derived from the observed memory error rate. Together with timing uncertainties and margins, these observables are indicators of the hardware fitness [FRJ+07] and can serve as inputs to system-wide calibration and adaptation.

Among these observables, temperature and specific stress patterns imposed by the workload are factors that cause or accelerate circuit aging. Workload can be monitored at different resolutions in the system, for instance by tracking the duration of usage of modules. The gate level workload monitor of [BCI+13] allows fine-grained observation of stress patterns in critical circuit structures.

The actual deviation of behavior can be measured by dedicated sensors, such as aging monitors or stability checkers [APZM07], or by conducting in-field self-tests for delay faults. Such tests must have higher sensitivity so that small deviations become detectable. Often, built-in test structures are reused to facilitate the test with low hardware overhead. In [SGH+07], processor cores are tested with functional patterns under increasingly stressful conditions until the timing guardband is exceeded. A signature register captures the responses and internal state at speed for comparison with the expected value. To increase the stress, the voltage is reduced and the frequency is increased.

The authors of [BaMi09] propose to select a small number of paths that turn critical after aging. The corresponding path delay faults are tested in-field. For memory arrays, a BIST-based measurement of aging degradation has been demonstrated in [AM10]. The Dependable Architecture with Reliability Testing (DART) [SKY+12] efficiently integrates logic and memory self-tests for health prediction and their control into the existing DFT and BIST infrastructure of a system. A DART controller orchestrates the self-tests in different test sessions in the system, setups the test conditions, such as increased frequency, and gathers test results for evaluation. Since the controller tracks the delay changes measured during the tests over the system lifetime, threatening deviations can be detected and counter-measures applied before an actual failure occurs. Monitors for non-functional observables, like temperature and voltage, are used to increase the accuracy of the tests or may serve as

standalone indicators of the stress, as in the health monitoring architecture of [ZhKe14].

If self-tests are executed faster-than-at-speed in order to uncover small delay faults, a flexible handling of the high and frequency-dependent ratio of unknown values in the test responses is required [SHQ10, HIK+14]. Alternatively, aging monitors can be reused during such a delay test if already present in the circuit [LKW17].

The information obtained from self-tests and health monitoring is input to an online dependability manager, which reasons about emerging failures and means of appropriate mitigation, for instance based on static policies [TKD+07] or dynamic ones [SCT+16].

VI. Tradeoffs in Self-Testing and Self-Checking

The methods classified in Figure 1 and described in Section III exhibit different characteristics in terms of effectiveness (fault coverage, fault detection latency, diagnostic resolution) and incurred cost (test time, performance impact, area / memory / power overhead). A qualitative comparison is given in Table 1.

The achievable fault and error coverage of the approaches can be traded off with area and energy overhead as well as test time or performance overhead. For pseudo-random BIST, for instance, additional test points can be inserted in the module under test, the number of applied patterns can be increased, or different seeds or polynomials can be used for on-chip pattern generation. The fault detection latency of self-tests depends on the frequency of the tests, which is also influenced by the test time and resulting performance impact. If the test application time is too long, the test can be split up into multiple shorter test sessions to avoid a long suspension of the module operation. Test in general and BIST in particular dissipate a lot of power, which can be controlled by additional design effort or higher test time. For deterministic patterns, the test power can be adjusted at a fine-grained level.

Error coverage and detection latency of self-checking design and software- and algorithm-based fault tolerance can be improved by more fine-grained checking, e.g., instrumenting each operation of an algorithm with checks up to duplicate execution. This also impacts the required area for self-checking and performance for algorithm-based fault tolerance and in general increases energy dissipation.

VII. Fault and Failure Management

In self-aware systems, the data sampled in the on-chip instrumentation must be aggregated and evaluated on chip. This requires both access mechanisms to the instruments and a central or distributed resource management. Resource management can be implemented on a processor by firmware or as a hardware unit [TBH+10].

Serial scan-based access, based on the popular JTAG boundary scan standard, has been widely adopted to connect instruments on chip. However, the increasing number and diversity of instruments lead to the development of more flexible and scalable access mechanisms based on reconfigurable scan networks (RSNs) and their recent standardization in IEEE Std 1149.1-2013 and IEEE Std 1687-2014. In RSNs, the path through which data is shifted can be reconfigured, for instance for minimum access latency to a set of targeted instruments.

These scan-based access mechanisms have been used to construct comprehensive access architectures for system and reliability management [JDS13, HeTe14, ZNL16]. Such architectures support self-aware systems by well-defined (instead of ad-hoc) interfaces, access procedures, and shared resources for instrument management such as for calibration, start and control of measurements, local response storage, or event-based signaling. Figure 4 shows the architecture proposed in [JDS13] to gather data from instruments spread over the resources in the system. The instrument manager decouples the details of the RSN-based communication to instruments in the hardware resources from the fault manager, which maintains the state of the resources as part of the operating system. The obtained data can serve as input to predictive health models. The flexible, self-reconfiguring scan network described in [ZNL16] provides low-latency error signaling for concurrent checkers and monitors and also an efficient error localization.

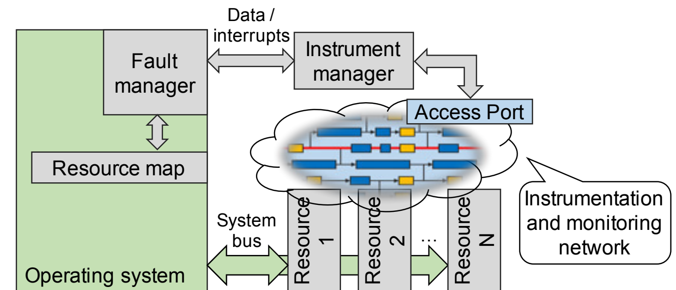

Figure 4. Fault management architecture according to [JDS13]

With the increasing importance of the on-chip infrastructure in self-aware systems, secure infrastructure access must be provided to prevent leakage or manipulation of sensitive instrument data or side-channel based attacks. This level of security is especially important in safety-critical systems, where an attack may cause unsafe system behavior. This requires a design methodology beyond ad-hoc solutions that incorporates access privileges and protection and secure data transmission at infrastructure level [KSG+17]. Secondly, flexible, scalable, and secure infrastructure access architectures must be designed and verified, both for JTAG-based as well as emerging reconfigurable scan-based access [RoKa10, BKW15].

VIII. Case Studies

The following two case studies, an adaptive network-on-chip and a runtime-reconfigurable architecture, illustrate the implementation of self-awareness by self-test and self-diagnosis for dependable and adaptive operation.

A. Networks-on-Chip

A network-on-chip (NOC) is a scalable communication architecture for many-core systems. It employs interconnected routers to route data packets from their source to the destination node. The structure of a router is shown in Figure 5. To ensure high reliability and availability of the communication, self-aware systems monitor the state of the NOC, comprising self-checking for errors, thorough self-test, and in-field diagnosis for adaptation to impaired NOC routers or links.

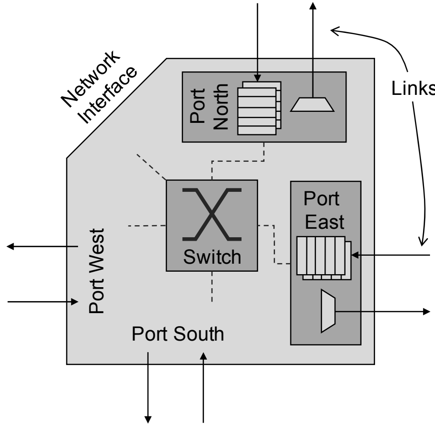

Figure 5. Structure of a NOC router with network interface and four ports

Domain-specific implementations of self-checking design may incur much lower overhead than generic implementations based on duplication with comparison or parity prediction. For NOCs, an efficient hybrid error detection scheme has been proposed in [DKW14], which combines parity-based error checking of transmitted data in the links with a dedicated error checking logic synthesized for yet uncovered errors in the router. This combination results in a low area overhead of only 52% and covers errors resulting from all single combinational and transition delay faults of the links and routers in the communication structure of the NOC.

While this self-checking detects errors with low latency, it does not provide diagnostic information. An in-field self-test of the NOC is required to determine if a detected error had a permanent or transient cause. Scan-based application of the test disrupts the operation of parts or even the whole NOC and is not desirable [TBH+10]. In [DaWu16], functional test stimuli are generated based on a structural fault model, which detect all testable faults and can be easily applied in the field without access to scan chains, e.g. by a processor.

Reconfigurable NOCs allow fine-grained management of their hardware resources, for example by disabling known faulty parts for graceful degradation. A challenge is, however, to decide whether an observed error is caused by a fault that merely affects a single link or port of a router or multiple parts. In the former case, the affected part can be disabled, and the remaining functionality of the NOC can be still used. For the location and classification of faults, functional tests with unique failure signatures for faults affecting only single resources are generated. The signatures of faults that can be masked by disabling a port or link of a router are then stored in the system for autonomous in-field diagnosis and NOC reconfiguration [DaWu16].

B. Runtime-Reconfigurable Architectures

FPGA-based runtime-reconfigurable architectures offer a high potential for energy-efficient acceleration of complex applications [CaHu11]. Their reconfigurability has also been exploited to adapt to the environment and hardware state for increased dependability. The architecture in [ZKI+14, ZBK+16] is an example of a self-aware system that dynamically schedules self-test and self-checking methods tailored for FPGAs: A resource manager monitors the state and usage of the computational resources and autonomously triggers self-tests, selects the required level of concurrent checking for errors, and balances the workload to increase system lifetime. The involved components and interactions are shown in Figure 6.

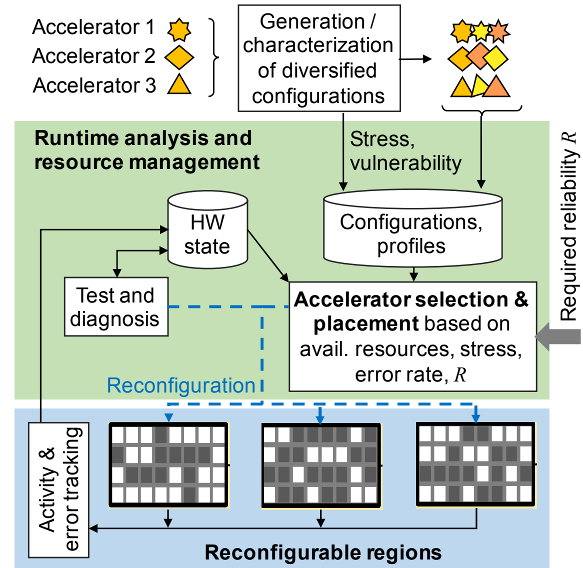

Figure 6. Self-aware resource management in a runtime reconfigurable system [ZKI+14, ZBK+16]

The regularity of the fabric, consisting of reconfigurable interconnect as well as combinational and sequential components, lends itself for efficient self-tests and in-field diagnosis [ASE04]. Periodic and on-demand execution of structural and functional self-tests achieve a high fault coverage and low fault detection latency with negligible performance impact [BBI+13].

If permanent faults are detected and localized, special module configurations with diversified resource usage are employed, which are generated at design time such that certain fault types are tolerated in at least one configuration [ZBK+16]. In addition, the diversity of used resources in such configurations can also be exploited to balance the stress induced by their operation [SKM+08]. The resource manager tracks at runtime for each resource the frequency, duration and type of usage to select the most appropriate module configuration such that the maximum stress in all resources is minimized [ZBK+16].

If the runtime-reconfigurable system is aware of the environmental radiation, e.g. by external sensors or by monitoring the error rate in protected memory arrays, the level of self-checking can be adjusted to the required degree of reliability. The resource manager dynamically determines the optimal implementation of configured modules, resorting to duplication or triplication only for the most vulnerable parts and time periods with high radiation. Since this avoids over-protection, the resulting system maximizes application performance and minimizes energy dissipation [ZKI+14].

IX. Conclusions

In-field detection and diagnosis of errors in self-aware systems and their causes in the hardware platform require self-checking, test and diagnosis at different levels. The overhead in terms of area, power, and performance is determined by the required error detection latency and diagnostic resolution. Two case studies demonstrate the application of these methods in the context of a network-on-chip and a runtime reconfigurable system.

Acknowledgment

This work is supported in parts by the German Research Foundation (DFG) under grant WU 245/17-1 (ACCESS) and as part of the priority program "Dependable Embedded Systems" (SPP 1500 – http://spp1500.itec.kit.edu).

References

[ALRL04] A. Avizienis, J.-C. Laprie, B. Randell, C. Landwehr: Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE Trans. Dep. Secure Computing, Vol. 1(1), pp. 11-33, Jan. 2004.

[AM10] F. Ahmed, L. Milor: Reliable cache design with on-chip monitoring of NBTI degradation in SRAM cells using BIST. Proc. IEEE VLSI Test Symposium (VTS), 2010, pp. 63-68.

[APZM07] M. Agarwal, B. Paul, M. Zhang, S. Mitra: Circuit Failure Prediction and Its Application to Transistor Aging. Proc. IEEE VLSI Test Symposium (VTS), 2007, pp. 277-286.

[ASE04] M. Abramovici, C. E. Stroud, J. M. Emmert: Online BIST and BIST-based diagnosis of FPGA logic blocks. IEEE Trans. Very Large Scale Integration (VLSI) Systems, vol. 12(12), pp. 1284-1294, Dec. 2004.

[BaMi09] A. Baba, S. Mitra: Testing for Transistor Aging. Proc. IEEE VLSI Test Symposium (VTS), 2009, pp. 215-220.

[BardMS87] P. Bardell, W. H. McAnney, J. Savir: Built-in Test for VLSI: Pseudorandom Techniques, Wiley, 1987.

[BBI+13] L. Bauer, C. Braun, M. E. Imhof, M. A. Kochte, E. Schneider, H. Zhang, J. Henkel, H.-J. Wunderlich: Test Strategies for Reliable Runtime Reconfigurable Architectures. IEEE Transactions on Computers, Vol. 62(8), August 2013, pp. 1494-1507.

[BCI+13] R. Baranowski, A. Cook, M.E. Imhof, C. Liu, H.-J. Wunderlich: Synthesis of Workload Monitors for On-Line Stress Prediction. Proc. IEEE Symp. Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2013, pp. 137-142.

[BCZ07] M. Boule, J. S. Chenard, Z. Zilic: Assertion Checkers in Verification, Silicon Debug and In-Field Diagnosis. Proc. IEEE International Symposium on Quality Electronic Design (ISQED), 2007, pp. 613-620.

[BeBl12] M. Beckler, R. D. Blanton: On-chip diagnosis for early-life and wear-out failures. Proc. IEEE International Test Conference (ITC), pp. 1-10, 2012.

[BKW15] R. Baranowski, M. A. Kochte, H.-J. Wunderlich: Fine-Grained Access Management in Reconfigurable Scan Networks. IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems (TCAD), Vol. 34(6), June 2015, pp. 937-946.

[BSS+06] P. Bernardi, E. Sanchez, M. Schillaci, G. Squillero, M. S. Reorda: An Effective Technique for Minimizing the Cost of Processor Software-Based Diagnosis in SoCs. Proc. Design Automation & Test in Europe Conference (DATE) 2006, pp. 1-6.

[CaHu11] J. Cardoso, M. Huebner: Reconfigurable Computing: From FPGAs to Hardware/Software Codesign. Springer, 2011.

[CMAB07] K. Constantinides, O. Mutlu, T. Austin, V. Bertacco. Software-Based Online Detection of Hardware Defects Mechanisms, Architectural Support, and Evaluation. Proc. IEEE/ACM International Symposium on Microarchitecture (MICRO), 2007, pp. 97-108.

[DaWu16] A. Dalirsani, H.-J. Wunderlich: Functional Diagnosis for Graceful Degradation of NoC Switches. Proc. IEEE Asian Test Symposium (ATS'16), 21-24 November 2016.

[DJS15] N. Dutt, A. Jantsch, S. Sarma: Self-Aware Cyber-Physical Systems-on-Chip, Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 46-50, 2015.

[DKW14] A. Dalirsani, M. A. Kochte, H.-J. Wunderlich: Area-Efficient Synthesis of Fault-Secure NoC Switches. Proc. IEEE International On-Line Testing Symposium (IOLTS), 7-9 July 2014, pp. 13-18.

[EKD+03] D. Ernst, N. Kim, et al.: Razor: a low-power pipeline based on circuit-level timing speculation. Proc. IEEE/ACM International Symposium on Microarchitecture, pp. 7–18, 2003.

[ElWu10] M. Elm, H.-J. Wunderlich: BISD: Scan-Based Built-In Self-Diagnosis. Proc. ACM/IEEE Design Automation and Test in Europe (DATE), 2010, pp. 1243-1248.

[FRJ+07] R. Franch, P. Restle, N. James, W. Huott, J. Friedrich, R. Dixon, S. Weitzel, K. Van Goor, G. Salem: On-chip timing uncertainty measurements on IBM microprocessors. Proc. IEEE International Test Conference (ITC), 2007, pp. 1-7.

[GMK91] T. Grüning, U. Mahlstedt, H. Koopmeiners: DIATEST: a fast diagnostic test pattern generator for combinational circuits. Proc. IEEE International Conference on Computer-Aided Design (ICCAD), 1991, pp. 194-197.

[GNW10] P. Girard, N. Nicolici, X. Wen (Eds.): Power-Aware Testing and Test Strategies for Low Power Devices. Springer, 2010.

[GoVe90] A. J. van de Goor, C. A. Verruijt: An Overview of Deterministic Functional RAM Chip Testing. ACM Computing Surveys, Vol. 22(1), March 1990, pp. 5-33.

[GPA+11] D. Gizopoulos, M. Psarakis, S. V. Adve, P. Ramachandran, S. K. S. Hari, D. Sorin, A. Meixner, A. Biswas, X. Vera: Architectures for online error detection and recovery in multicore processors. Proc. Design, Automation & Test in Europe (DATE), 2011, pp. 1-6.

[GRP15] M. Gaudesi, M. Sonza Reorda, I. Pomeranz: On Test Program Compaction. Proc. IEEE European Test Symposium (ETS), 2015, pp. 1-6.

[HeTe14] M. T. He, M. Tehranipoor: SAM: A comprehensive mechanism for accessing embedded sensors in modern SoCs. Proc. IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2014, pp. 240-245.

[HIK+14] S. Hellebrand, T. Indlekofer, M. Kampmann, M. A. Kochte, C. Liu, H.-J. Wunderlich: FAST-BIST: Faster-than-at-Speed BIST targeting hidden delay defects. Proc. IEEE International Test Conference (ITC), 2014, paper 29.3.

[HuAb84] K.-H. Huang and J. A. Abraham: Algorithm-Based Fault Tolerance for Matrix Operations," in IEEE Transactions on Computers, vol. C-33, no. 6, pp. 518-528, June 1984.

[ImWu14] M. E. Imhof, H.-J. Wunderlich: Bit-Flipping Scan - A Unified Architecture for Fault Tolerance and Offline Test. Proc. ACM/IEEE Design Automation and Test in Europe (DATE), 2014, pp. 1-6.

[JDS13] A. Jutman, S. Devadze, K. Shibin: Effective Scalable IEEE 1687 Instrumentation Network for Fault Management. IEEE Design & Test of Computers, 2013, Vol. 30, No. 5, pp. 26-35.

[KDB+15] M. A. Kochte, A. Dalirsani, A. Bernabei, M. Omana, C. Metra, H.-J. Wunderlich: Intermittent and Transient Fault Diagnosis on Sparse Code Signatures. Proc. IEEE Asian Test Symposium (ATS'15), 22-25 November 2015, pp. 157-162.

[KLCD02] A. Krstic, Wei-Cheng Lai, Kwang-Ting Cheng, L. Chen, S. Dey: Embedded software-based self-test for programmable core-based designs. IEEE Design & Test of Computers, vol. 19, no. 4, pp. 18-27, 2002.

[KSG+17] M. A. Kochte, M. Sauer, L. R. Gomez, P. Raiola, B. Becker, H.-J. Wunderlich: Specification and Verification of Security in Reconfigurable Scan Networks. Proc. IEEE European Test Symposium (ETS), 2017, pp. 1-6.

[KZW09] M. A. Kochte, C. G. Zoellin, H.-J. Wunderlich: Concurrent self-test with partially specified patterns for low test latency and overhead. Proc. IEEE European Test Symposium (ETS), 2009, pp. 53-58.

[LCH05] K.-J. Lee, C.-Y. Chu, Y.-T. Hong: An embedded processor based SOC test platform. Proc. IEEE International Symposium on Circuits and Systems (ISCAS), 2005, pp. 2983-2986.

[LMM08] Y. Li, S. Makar, S. Mitra: CASP: Concurrent Autonomous Chip Self-Test Using Stored Test Patterns. Proc. ACM/IEEE Design Automation and Test in Europe (DATE), 2008, pp. 885-890.

[LKW17] C. Liu, M.A. Kochte, H.-J. Wunderlich: Aging Monitor Reuse for Small Delay Fault Testing. Proc. IEEE VLSI Test Symposium (VTS), 2017.

[LTK16] K-J. Lee, P-H. Tang, M. Kochte: An On-Chip Self-Test Architecture with Test Patterns Recorded in Scan Chains. Proc. IEEE International Test Conference (ITC), 2016, Paper 16.3.

[MaMc88] A. Mahmood, E. J. McCluskey: Concurrent Error Detection Using Watchdog Processors - A Survey. IEEE Transactions on Computers, vol. 37, pp. 160-174, Feb. 1988.

[MiMc00] S. Mitra, E. J. McCluskey: WHICH CONCURRENT ERROR DETECTION SCHEME TO CHOOSE? Proc. IEEE International Test Conference (ITC), 2000.

[Nico92] M. Nicolaidis: Transparent BIST for RAMs. Proc. IEEE International Test Conference (ITC), 1992, pp. 598-607.

[Nico07] M. Nicolaidis: GRAAL: a new fault tolerant design paradigm for mitigating the flaws of deep nanometric technologies. Proc. IEEE International Test Conference (ITC), 2007, pp. 1–10.

[OSM02] N. Oh, P. P. Shirvani, E. J. McCluskey: Control Flow Checking by Software Signatures. IEEE Transactions on Reliability, vol. 51, pp. 111-122, Mar. 2002.

[RCS+14] A. Riefert, L. Ciganda, M. Sauer, P. Bernardi, M. S. Reorda, B. Becker: An effective approach to automatic functional processor test generation for small-delay faults. Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE), 2014, pp. 1-6.

[PGSS10] M. Psarakis, D. Gizopoulos, E. Sanchez, M. Sonza-Reorda: Microprocessor software-based self-testing. IEEE Design & Test of Computers, vol. 27(3), 2010, pp. 4-19.

[RoKa10] K. Rosenfeld, R. Karri: Attacks and Defenses for JTAG. IEEE Design & Test, Vol. 27(1), January 2010, pp. 36-47.

[SBK+15] A. Schöll, C. Braun, M. A. Kochte, H.-J. Wunderlich: Low-Overhead Fault-Tolerance for the Preconditioned Conjugate Gradient Solver. Proc. IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2015, pp. 60-65.

[SCT+16] M. Sadi, G. Contreras, D. Tran, J. Chen, L. Winemberg, M. Tehranipoor: BIST-RM: BIST-assisted reliability management of SoCs using on-chip clock sweeping and machine learning. Proc. IEEE International Test Conference (ITC), 2016, paper 15.1.

[SGH+07] J. Smolens, B. Gold, J. Hoe, B. Falsafi, K. Mai: Detecting Emerging Wearout Faults. IEEE Workshop on Silicon Errors in Logic – System Effects (SELSE), 2007.

[SHQ10] A. Singh, C. Han, X. Qian: An Output Compression Scheme for Handling X-States from Over-Clocked Delay Test. Proc. IEEE VLSI Test Symposium (VTS), 2010, pp. 57-62.

[SKM+08] S. Srinivasan, R. Krishnan, et al.: Toward increasing FPGA lifetime. IEEE Transactions on Dependable. and Secure Computing. vol. 5(2), pp. 115–127, 2008.

[SKY+12] Y. Sato, S. Kajihara, T. Yoneda, K. Hatayama, M. Inoue, Y. Miura, S. Ohtake, T. Hasegawa, M. Sato, K. Shimamura: DART: Dependable VLSI test architecture and its implementation. Proc. IEEE International Test Conference (ITC), 2012, pp. 1-10.

[TBH+10] T. D. Ter Braak, S. T. Burgess, H. Hurskainen, H. G. Kerkhoff, B. Vermeulen, X. Zhang: On-line dependability enhancement of multiprocessor SoCs by resource management. Proc. International Symposium on System on Chip, 2010, pp. 103-110.

[TKD+07] J. Tschanz, N. S. Kim, S. Dighe, et al.: Adaptive Frequency and Biasing Techniques for Tolerance to Dynamic Temperature-Voltage Variations and Aging. Proc. IEEE International Solid-State Circuits Conference (ISSCC). 2007, pp. 292-604.

[VoEf14] I. Voyiatzis, C. Efstathiou: Input Vector Monitoring Concurrent BIST Architecture Using SRAM Cells. IEEE Trans. Very Large Scale Integration (VLSI) Systems, vol. 22, no. 7, pp. 1625-1629, July 2014.

[ZBK+16] H. Zhang, L. Bauer, M. A. Kochte, E. Schneider, H.-J. Wunderlich, J. Henkel: Aging Resilience and Fault Tolerance in Runtime Reconfigurable Architectures. IEEE Transactions on Computers, vol. 66(6), pp. 957-970, June 2017.

[ZhKe14] Y. Zhao, H. G. Kerkhoff: Design of an Embedded Health Monitoring Infrastructure for Accessing Multi-processor SoC Degradation. Proc. IEEE Euromicro Conference on Digital System Design (DSD), 2014, pp. 154-160.

[ZhWu06] J. Zhou, H.-J. Wunderlich: Software-Based Self-Test of Processors under Power Constraints. Proc. ACM/IEEE Design Automation and Test in Europe (DATE), 2006, pp. 430-435.

[ZMM+06] M. Zhang, S. Mitra, T. M. Mak. N. Seifert, N. Wang, Q. Shi, K. Kim, N. Shanbhag, S. Patel: Sequential Element Design with Built-In Soft Error Resilience. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 14, no. 12, pp. 1368-1378, Dec. 2006.

[ZNL16] F. G. Zadegan, D. Nikolov, E. Larsson: A self-reconfiguring IEEE 1687 network for fault monitoring. Proc. IEEE European Test Symposium (ETS), 2016, pp. 1-6.

[ZKI+14] H. Zhang, M. A. Kochte, M. E. Imhof, L. Bauer, H.-J. Wunderlich, J. Henkel: GUARD: GUAranteed Reliability in Dynamically Reconfigurable Systems. Proc. ACM/EDAC/IEEE Design Automation Conference (DAC), 1-5 June 2014.

Table 1: Qualitative comparison of self-testing and self-checking approaches for self-aware systems

Class	**Periodic / on-demand self-test**				**Self-checking design & concurrent error detection**			
Type	**Pseudo-random patterns**	**Compressed deterministic patterns**	**Memory BIST**	**Software-based self-test (SBST)**	**Concurrent BIST**	**Self-checking design**	**Error control coding**	**Algorithm-based fault-tolerance (ABFT)**
Targeted structure	Logic	Logic	Logic, memory	Logic, Memory	Logic	Logic	Memory	Logic, memory
Fault coverage	Medium to high	High	High	Medium to high	Medium	Medium to high	High	Medium to high
Fault / error detection latency	Fault detection:				Error detection:			
	Depending on test frequency				Medium	Low	Low	Low to medium
Diagnostic resolution	Module-level	Module-level	High	Low	Module-level	Module-/interface level, high temporal resolution	Memory location	Module/operation-level, moderate temporal resolution
Test time overhead	High	Low	Variable	High	N/A			
Performance overhead	Depending on test length and frequency, lower for short interruptible / resumable test sessions.				Very low	Very low to high (tradeoff between time / structural redundancy)	(Very) low	Low to high (depending on coverage and detection latency)
Area cost	Low	Medium to high	Low	None	Medium	Medium to high	Low to medium	None
Memory requirements	Memory for seeds, expected signature	ROM or non-volatile memory for pattern seeds, signature	Memory for test programs	Memory for test programs	None	None	Low to medium	Low to medium
Power overhead Avg.	Medium to high	Low to high	Functional power	Functional power	Low to medium	Medium to high	Low	Low to medium
Power overhead Peak	Medium to high	Medium to high						

Author biographies and contact:

Michael A. Kochte received his Dr. rer. nat. (Ph.D.) degree from the University of Stuttgart, Germany, in 2014, where he currently leads the research group for Dependable Hardware of the Institute for Computer Architecture and Computer Engineering. His research interests include reconfigurable computing, hardware test and reliability, and hardware security.

kochte@iti.uni-stuttgart.de , Tel. +49-711- 685 88 361

Hans-Joachim Wunderlich is a full professor of computer science at the University of Stuttgart. He obtained his Ph.D. degree (Dr. rer. nat.) from the University of Karlsruhe and studied mathematics and philosophy at the universities of Konstanz and Freiburg. His research interests include design, test and fault tolerance of digital systems. He is a Fellow of IEEE.

wu@informatik.uni-stuttgart.de , Tel. +49-711- 685 88 391, Fax +49- 711 685 88 288