Implicit Intermittent Fault Detection in Distributed Systems

Peter Waszecki[1], Matthias Kauer[1], Martin Lukasiewycz[1], Samarjit Chakraborty[2]

[1] TUM CREATE, Singapore, Email: peter.waszecki@tum-create.edu.sg
[2] TU Munich, Germany, Email: samarjit@tum.de

***Abstract*—This paper presents a novel approach to detect resources in distributed systems with an increased occurrence of intermittent faults that exceed the amount of unavoidable transient faults caused by environmental phenomena. Intermittent faults occur due to stressed resources and often are a precursor of permanent faults. The proposed early fault detection and diagnosis allows the use of precautionary measures before the permanent failure of a component in a distributed system occurs. In this paper, we present four methods that can implicitly detect intermittent faults by taking the distributed applications and their dependencies into account. Thus, explicit tests are not required which would lead to additional costs and resource load. On the other hand, the implicit approach may considerably reduce the number of plausibility tests compared to the conservative solution with one test per resource. We analyzed and evaluated implementations of the proposed fault detection principle. The experimental results give evidence of the feasibility of our approach and show a comparison of the implemented methods in terms of runtime and detection rate.**

I. Introduction

Today, the reliability of embedded systems and the associated safety aspects are of high relevance in many domains with strict real-time requirements such as in avionics and automotive. Also for non-safety critical applications, for example, in consumer electronics, efficient fault detection and fault tolerance mechanisms are important to fulfill the customers' quality expectations. On the other hand, ever-growing Very Large Scale Integration (VLSI) processes with shrinking geometries and decreasing power supply voltages result in devices which are increasingly susceptible to transient faults and, hence, might have a negative impact on the system reliability [1]. In distributed systems, a failure of a single component can influence the behavior of a multitude of applications. It is therefore desirable to detect potential failures of components before they actually happen and apply precautionary measures which can vary from graceful degradation to a replacement of the affected component. For such an early detection, an increased number of non-permanent faults is a suitable indicator to determine stressed components.

Contributions of the paper. In this paper, we propose an approach to implicitly detect components in a distributed system that show an increased number of non-permanent faults. Assuming that deficient hardware causes the presence and accumulation of so-called *intermittent* faults which lead to software errors, then a potential imminent failure of a specific resource can be projected by analyzing the results of a set of plausibility tests running within regular tasks or as discrete applications. A major objective of the proposed approach is to perform such a detection implicitly to keep the additional costs and resource utilization low. In contrast to an explicit detection which would require additional test tasks for each component, the proposed fault detection relies on existing plausibility tests which are part of the distributed applications. Also, due to the architectural constraints of the resources it is often only possible to check the consistency of test results at specific points, e.g., on particular Electronic Control Units (ECUs). For this purpose, we take the distribution of applications, the runtimes of tasks, and their data-dependencies into account to implicitly determine the component with an increased number of intermittent faults.

We propose four different implementations for an implicit intermittent fault detection in distributed systems. The first two methods are based on the analysis of linear dependencies within the system model, while the other two methods use Integer Linear Programming (ILP) formulations for an optimization-based approach. In the experimental results, we show the feasibility of the proposed fault detection and compare the different methods in terms of their runtime and detectability. For the sake of simplicity, in this paper we consider a system with at most one stressed resource. However, ILP approaches innately support the detection of multiple stressed resources and the methods based on the analysis of linear dependencies might be adapted appropriately. The extension of our approach to a concurrent detection of multiple faulty resources is part of future work.

In this paper, we consider the automotive domain, working on the level of Electrical and Electronic (E/E) architectures, i.e., we assume ECUs, sensors, and actuators as the basic components. However, the proposed methodology is also applicable to other levels of granularity, e.g., in case of a Multiprocessor System-on-Chip (MPSoC), basic components might be computing units, buses, switches, or memories. While for E/E architectures the number of basic components stays in a double-digit or lower three-digit range, other domains might easily reach ranges of 10^4 or even 10^5, which makes the scalability of our approach an important objective. Since the detection might also be performed offline by analyzing the detected failed tests within a certain time interval, the approaches do not have to satisfy strict runtime requirements, but have to be within a reasonable range with a good scalability.

Organization of the paper. The remainder of the paper is organized as follows: Section II discusses the related work. Section III describes the system model, including a formal definition of the system and problem. The detection approaches are proposed in Section IV. Experimental results are presented in Section V before the paper is concluded in Section VI.

II. Related Work

In the considered distributed systems a resource can execute one or multiple periodic tasks (or generally speaking manipulate data), while these tasks might have data-dependencies across

This work was financially supported in part by the Singapore National Research Foundation under its Campus for Research Excellence And Technological Enterprise (CREATE) programme.

different resources by using messages or shared memory for communication. The mapping of tasks to resources is a crucial point and is discussed, e.g., in [2]. To deal with errors in the system, the most common fault tolerance strategies are, on the one hand, re-mapping of tasks as discussed in [3] and [4] for the use of Network-on-Chip (NoC)-based MPSoCs. On the other hand, when intact resources do not have enough free capacity to take over tasks of the affected resources, a graceful degradation mechanism is necessary as, for example, described in [5]. Our approach is set at the fault detection stage and, thus, before the fault tolerance mechanisms come into action. For that reason, it builds on an existing schedule and a given task distribution.

A number of approaches are dealing with the reliability and fault diagnosis in distributed systems. In [6], a hardware based monitoring is proposed that enables the detection of transient faults in a Real-Time Operating System (RTOS). Three on-line self-testing policies for multi-processors are investigated by Héron et al. [7] in terms of performance and detection probability. Approaches to the problem on a system design level are presented in [8] and [9] where the system is modeled by a Data Flow Graph (DFG) and the authors maximize the reliability by exchanging the resources until the defined constraints are met. In contrast, [10] considers not only reliability but also all other design objectives simultaneously.

However, none of these approaches consider the problem that an accumulation of non-permanent faults might finally lead to a permanent failure of a resource or, in the worst case, the entire system. To the best of our knowledge, this is the first work analyzing and comparing test failure rates to implicitly detect resources tending to fail through the steady increase of intermittent faults.

III. System Model

In our system model, we assume that a resource can be affected by two types of faults: permanent faults, which durably damage the system and non-permanent faults, which can further be divided into transient and intermittent faults, and only temporarily affect the system [11]. Although permanent faults are going along with enduring physical changes of the affected hardware, they are often preceded by an increased number of intermittent faults which themselves are caused by malfunctioning hardware and occur with a high arbitrary frequency, see [1]. In contrast to intermittent faults, transient faults are caused by temporary environmental phenomena, like cosmic rays, electromagnetic interference, electrostatic discharge or radiation from lightning. The main purpose of this work is to identify intermittent faults early before a permanent fault occurs in order to apply appropriate precautions. Besides this, an important aspect is the *implicit* nature of the fault detection to avoid additional testing overhead. The occurrence of intermittent faults can be validated with the help of plausibility tests which are evaluating the outcome of particular periodic tasks. According to the hypothesis of the Resilience Articulation Point (RAP), all faults originating from a physical phenomenon, if not masked, will manifest as a permanent or non-permanent single- or multi-bit flip [12]. We assume that the errors caused by these bit flips are propagated between data-dependent tasks and finally result in plausibility test failures, with the probabilistic distribution of the intermittent faults remaining unaltered. Thus, a failed plausibility test indicates an error in an own or a preceding task originally caused by a fault on a resource. To model the

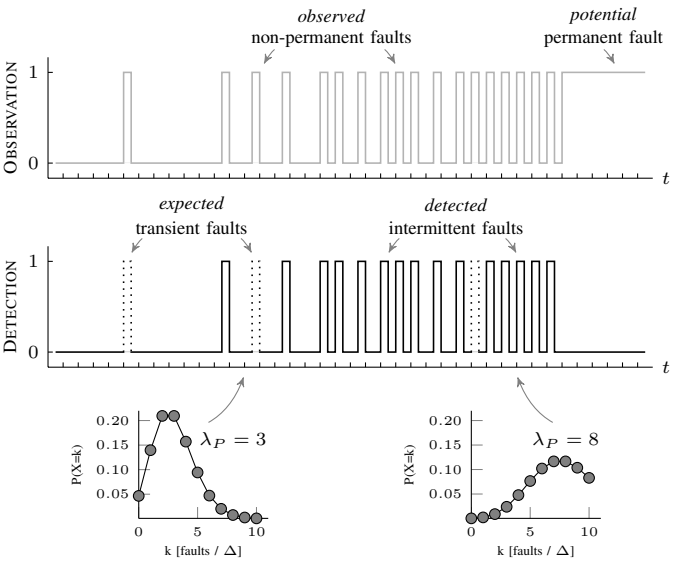

Fig. 1. Illustrative example for the occurrence of faults on a resource. In the beginning the operation is affected by transient faults only. Then, gradually, more and more intermittent faults occur which finally result in a permanent fault. The probabilistic distribution of the intermittent faults depicts an early ($\lambda_P = 3$) and a late ($\lambda_P = 8$) stage of the fault period.

probability of the fault occurrence, the Poisson distribution is used, however, our approach is flexible enough to adopt other probabilistic error models, if required. Figure 1 shows how non-permanent faults can be split into a transient and an intermittent part, with the latter becoming more and more predominant over time as illustrated by a higher mean value λ_P of the Poisson distribution. It is important to consider transient faults as possible noise in our analysis approach to avoid false positive fault detections as far as possible.

Generally, due to architectural and cost reasons not every resource can be equipped with specific tests to detect intermittent faults. Therefore, our implicit detection makes use of existing plausibility tests which are part of the user applications. In our approach, we analyze a number of plausibility tests at the end of particular task sequences and compare the expected and observed results of several tests. This allows us to draw the right conclusion about the fault-causing resource. A motivating example is shown in Figure 2, where eight application tasks τ_{x_i/y_i} and two test tasks $t_{x/y}$ are mapped to three ECUs as indicated by the gray background areas. Depending on the particular assignment and utilization of the tasks, a higher rate of intermittent faults on a resource can be detected by analyzing the failure ratio of the test tasks. Given an equal utilization among all application tasks and a consistent error propagation towards the test tasks, a faulty ECU_1 will cause a failure ratio between t_x and t_y of $\frac{2}{1}$, since it is running two tasks from the t_x-task chain and only one from the t_y-task chain. Note that correspondingly a failure ratio of $\frac{1}{2}$ indicates a faulty ECU_2 and a ratio of $\frac{1}{1}$ a faulty ECU_3.

Formal Problem Description. In the following, a mathematical description shall formally define the problem, which is mainly based on the sets and functions listed below.

T	set of available tests
R	set of resources to be considered
$\mathcal{T}_r$	set of tasks on a resource $r \in R$
Δ	time interval for test failure observation
O_t	observed failures of test $t \in T$ in Δ

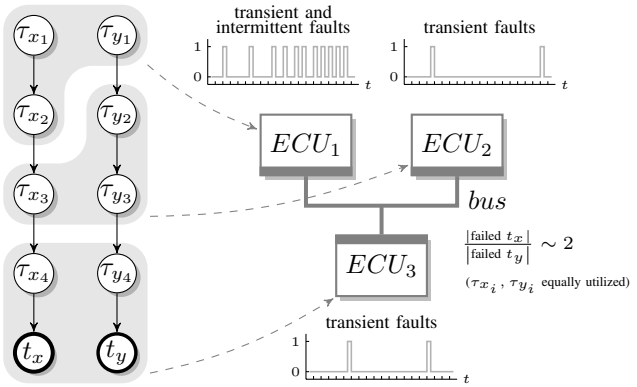

Fig. 2. Simplified example for the detection of a faulty resource. The application tasks τ_{x_1}, τ_{x_2}, τ_{y_1} are mapped to ECU_1 and τ_{x_3}, τ_{y_2}, τ_{y_3} are mapped to ECU_2 whereas the remaining τ_{x_4} and τ_{y_4} as well as the plausibility tests t_x and t_y are assigned to ECU_3. The increased intermittent fault rate on ECU_1 results in a test failure ratio of 2 between t_x and t_y.

E_t expected failures of test $t \in T$ in Δ

$\lambda : T \times R \to \mathbb{R}$ frequency of a test $t \in T$ failing due to a fault of $r \in R$

The mere quantitative analysis of plausibility test failures is not sufficient to detect a faulty resource. This is particularly the case, when considering the non-trivial example of having a system consisting of less tests than resources (see the example in Figure 2). Consequently, for each test $t \in T$ and each resource $r \in R$ we define a $\lambda(t, r)$ which represents the number of expected test failures per time interval caused by the respective resource.

The actual fault rate (i.e., number of faults per time interval) for a specific resource is a random variable following the Poisson distribution with a known average value. Figure 1 illustrates the increasing fault occurrence over time for a hypothetical resource with an increasing expectation value λ_P towards the permanent fault. In Equation (1) the fault rate is represented as an independent random variable X_r of the Poisson distribution with an expected value $E[X_r]$. The expected value is strongly dependent on the resource's susceptibility to intermittent faults and must be investigated experimentally or determined from the manufacturer's hardware description. In the scope of this work $E[X_r]$ shall be assumed to be known.

$$X_r \sim \text{Poi}(E[X_r]) \tag{1}$$

With e_τ describing the average execution time of a periodic task $\tau \in \mathcal{T}_r$ and h_τ describing its period, $\lambda(t, r)$ is defined as in Equation (2). Here, $pred(t)$ represents the set of tasks that are predecessors of the test t.

$$\lambda(t, r) = \sum_{\tau \in \mathcal{T}_r \cap pred(t)} \frac{e_\tau}{h_\tau} \cdot E[X_r] \tag{2}$$

The equation indicates whether the faulty resource runs any task in which an error would lead to a failure of test t. That can be the test task itself, or any of its predecessors.

The allocation of each test to the resources which can cause its failure leads to a test expectation matrix Λ which comprises all $\lambda(t, r)$ values as its elements (see Equation (3)).

$$\Lambda = \big(\lambda(t, r)\big)_{t,r} \tag{3}$$

This matrix describes the expected fault distribution under normal operation for transient faults only. In case of intermittent faults, the observations will deviate from this matrix and enable an implicit fault detection as described in Section IV.

Detectability Analysis. Before the actual detection process, it is important to initially ensure that the given distribution of test tasks leads to an unambiguous recognition of the affected resource. Given the presumption that only one resource can fail in the considered time interval a mutual comparison of the tests related to a specific resource enables a conclusion about the system's detectability. In detail, the so-called cosine similarity in Equation (4a) between two column vectors v_{r_i} and v_{r_j} in the Λ matrix will result in a number proportionally approximating 1 the smaller the angle θ between these two vectors is. It is lower than 1 otherwise, thus, indicating a better distinctness between two potentially faulty resources. Equation (4b) is illustrating this correlation in which ϵ_{det} is the maximum acceptable deviation of the cosine value from 1.

$$cos(\theta) = \left| \frac{\mathbf{v}_{r_i} \cdot \mathbf{v}_{r_j}}{\|\mathbf{v}_{r_i}\| \cdot \|\mathbf{v}_{r_j}\|} - 1 \right| \tag{4a}$$

$$cos(\theta) \begin{cases} \leq \epsilon_{det} \Rightarrow r_i \approx r_j, & \theta = \angle(\mathbf{v}_{r_i}, \mathbf{v}_{r_j}) \\ > \epsilon_{det} \Rightarrow r_i \neq r_j, & \theta = \angle(\mathbf{v}_{r_i}, \mathbf{v}_{r_j}) \end{cases} \tag{4b}$$

A valid definition of detectability can be formulated as follows.

Definition 1 (Fully Detecting Implementation): We call the allocation of plausibility tests to resources fully detecting, if each sub-matrix $\Lambda_s = (\mathbf{v}_{r_i}, \mathbf{v}_{r_j})$ of the expectation matrix Λ consists of linearly independent columns, i.e. has full rank (Equation (5)).

$$rank(\Lambda_s^{|T| \times 2}) = min(|T|, 2) \tag{5}$$

A trivial way to a fully detecting test setup would be to attach a plausibility test to every single resource. This leads to a diagonal matrix Λ_d which can always be solved and for this reason can be regarded as the reference solution. Therefore, one of the goal of this work is to beat the reference solution by reducing the number of required tests.

IV. Detection Approaches

Based on the presumption that the system to analyze is fully detectable, a potentially faulty resource shall be located. As a matter of principle, the solution of the problem can be abstracted to an analysis of the ratio between observed and expected test failures. Given the expectation matrix Λ, four possible realizations of the general fault detection principle shall be described and evaluated with respect to their correctness and performance. In the following, the sets O_t and E_t are representing observed and expected failures of a test $t \in T$, respectively, which occur within a time interval Δ. We assume that at least one resource $r \in R$ is faulty if the number of observed test failures originating from all possible resources is significantly higher than the number of the expected test failures within the time interval Δ, as shown in Equation (6).

$$O_t \gg \Delta \cdot \sum_{r \in R} \lambda(t, r) \tag{6}$$

A. Method I: Cosine Similarity

The first detection method uses a similar approach as the detectability analysis in Section III. We regard the expectation matrix as a set of column vectors, where each vector $\mathbf{v}_{r_i}{}^*$ is assigned to one resource and contains the corresponding utilization dependent expected test fault rates (Equation (7a)).

In addition, an observation vector $\mathbf{v}_{O_T}$ which consists of the observed failures of each plausibility test is used (Equation (7b)).

$$\mathbf{v}_{r_i} = [\lambda(t_1, r_i), \lambda(t_2, r_i), \cdots, \lambda(t_m, r_i)]^T \quad (7a)$$

$$\mathbf{v}_{O_T} = [O_{t_1}, O_{t_2}, \cdots, O_{t_m}]^T \quad (7b)$$

A cosine similarity analysis between the observation vector and the expectation vectors can detect the resource responsible for the failed tests, when the corresponding cosine value is close to 1. This is illustrated in Equation (8), where θ is the angle enclosed by the examined vectors and ϵ_{cos} is the maximum acceptable deviation from 1.

$$cos(\theta) \begin{cases} \leq \epsilon_{\text{cos}} \Rightarrow O_T \text{ from } r_i, & \theta = \angle(\mathbf{v}_{O_T}, \mathbf{v}_{r_i}) \\ > \epsilon_{\text{cos}} \Rightarrow O_T \text{ not from } r_i, & \theta = \angle(\mathbf{v}_{O_T}, \mathbf{v}_{r_i}) \end{cases} \quad (8)$$

B. Method II: SVD-TEST

Alternatively, the second method calculates the Singular Value Decomposition (SVD) of the sub-matrix $\Lambda_s = (\mathbf{v}_{O_T}, \mathbf{v}_{r_i})$ to determine the linear dependence, as shown in Equation (9). The observation O_T is thought of being caused by resource r_i, if the rank of the diagonal matrix Σ containing the singular values σ_1 and σ_2 is less than 2 (Equation (10)).

$${\Lambda_s}^{|T|\times 2} = U\Sigma V, \text{ with } \Sigma = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} \text{ and } |T| \geq 2 \quad (9)$$

$$rank(\Sigma) \begin{cases} < 2 \Rightarrow O_T \text{ from } r_i, & \text{if } \exists\sigma \leq \epsilon_{\text{svd}} \\ = 2 \Rightarrow O_T \text{ not from } r_i, & \text{if } \forall\sigma > \epsilon_{\text{svd}} \end{cases} \quad (10)$$

Regarding the performance and correctness, both methods lead to comparable results when using appropriate threshold values and, thus, are highly dependable on the correct choice of the tolerance interval ϵ. Numerous test runs proved $\epsilon_{\text{cos}} \sim 10^{-3}$ and $\epsilon_{\text{svd}} \sim 10^{-5}$ as good ranges for the analyzed system sizes.

C. Method III: CONFIDENCE INTERVAL

A fundamentally different detection approach can be applied by making use of the non-deterministic nature of fault occurrences. To mirror the stochastic uncertainty of the expected intermittent faults caused by a resource, we can create a confidence interval around E_t inside which an observation is thought to conform to its corresponding λ-values. Assuming, that the faults are following a Poisson distribution and that during normal operation all observed failures O_t lie within three standard deviations from the expected value E_t, we can define the interval according to the 3σ-rule as seen in Equation (11). Here, σ represents the standard deviation and μ the mean of the Poisson-distributed variable x.

$$P_r(\mu - 3\sigma \leq x \leq \mu + 3\sigma) \approx 0.9973 \quad (11)$$

Given the Cumulative Distribution Function (CDF) of x as $F_\lambda(x \leq k)$, the values λ_{lo} and λ_{hi} can be determined by approximating the corresponding CDFs to the limits resulting from the 3σ-rule, as shown in Equations (12a) and (12b). Accordingly, this leads to a confidence interval $[\lambda_{lo}, \lambda_{hi}]$ which is located around E_t.

$$\underset{\lambda_{lo}\in\mathbb{R}_0^+}{\text{minimize}} \ \lambda_{lo}, \quad \text{subject to } F_{\lambda_{lo}}(x \leq O_t) \leq \frac{P_r + 1}{2} \quad (12a)$$

$$\underset{\lambda_{hi}\in\mathbb{R}_0^+}{\text{maximize}} \ \lambda_{hi}, \quad \text{subject to } F_{\lambda_{hi}}(x \leq O_t) \geq \frac{1 - P_r}{2} \quad (12b)$$

As a next step, introducing a stress factor allows us to formulate an optimization problem with the confidence interval as one constraint (Equation (13b)). We define the expected observations E_t as sum of weighted λ-values over all resources, where the stress factor is x_r (Equation (13c)). Additionally, in Equations (13d) and (13e) we use y_r as a switch variable to decide whether x_r is significantly large ($y_r = 1$) or still in an acceptable range ($y_r = 0$), so that E_t stays inside the confidence interval. The threshold for this decision is set by x_{var}. The objective function (Equation (13a)) tries to find cases where only a minimal number of stressed resources is responsible for a failed test.

$$\underset{y_r\in\{0,1\}}{\textit{minimize}} \quad \sum_{r\in R} y_r \quad (13a)$$

subject to:

$$\forall t \in T: \quad \lambda_{lo}(O_t) \leq \frac{E_t}{\Delta} \leq \lambda_{hi}(O_t) \quad (13b)$$

$$\forall t \in T: \quad E_t = \sum_{r\in R} x_r \cdot \lambda(t, r) \cdot \Delta \quad (13c)$$

$$\forall r \in R: \quad x_r \geq x_{\text{var}} \cdot y_r \quad (13d)$$

$$\forall r \in R: \quad x_r \leq x_{\text{var}} + 10^{10} \cdot y_r \quad (13e)$$

For the optimization algorithm $\lambda(t, r)$ and O_t are considered to be constants from the set of positive rational numbers $\mathbb{R}_0^+$. Similarly, the variables E_t and x_r are defined in $\mathbb{R}_0^+$, whereas y_r contains only two values, $\{0, 1\}$.

To rule out additional resources being responsible for the failed tests, the optimization algorithm must be started a second time with the first solution excluded from the test. Only if the latter test reveals no solution, one can be sure about the correctness of the first run.

D. Method IV: χ^2-TEST

The fourth method to detect faulty resources is based on a statistical hypothesis test. To determine how well the expectation fits an observation the *Pearson's χ^2-Test* shall be used as a statistical model to describe the goodness of fit (see Equation (14)).

$$\chi^2 = \sum_{t\in T} \frac{(O_t - E_t)^2}{E_t} \quad (14)$$

The null hypothesis H_0 for the test is defined in Equation (15) and it indicates that all *observed* test failures O_t result from the *expected* test failures E_t.

$$H_0 : E_t \rightarrow O_t \quad (15)$$

H_0 shall be accepted if the p-value of the χ^2-Test is higher than a predefined significance level and rejected otherwise. Equation (16) denotes this inequality where $F(\chi^2, |T| - 1)$ is the CDF of the χ^2-distribution with $|T| - 1$ degrees of freedom.

$$1 - F(\chi^2, |T| - 1) \geq 0.05 \quad (16)$$

*Note, that the expectation vectors $\mathbf{v}_{r_i}$ are not scaled with the observation interval Δ, and hence do not represent the actual expected values E_t. This has no influence on the analytical result of the cosine similarity analysis, as it is magnitude independent.

To find solutions for the χ^2-test we can formulate an optimization problem analogue to that, used for the confidence interval approach. Again, we try to find cases where the sum of all switch variables y_r is minimal. However, the χ^2-Test puts E_t in a quotient term and hence, leads to a non-linear problem. In order to still be able to use linear solvers, the variable R_t serves as a binary representation of the reciprocal value E_t^{-1}. This is defined in the Equations (17b) and (17c) for the optimization problem formulated below.

$$\underset{y_r \in \{0,1\}}{\textit{minimize}} \quad \sum_{r \in R} y_r \tag{17a}$$

subject to:

$$\forall t \in T: \quad \chi^2 = \sum_{t \in T} O_t^2 \cdot R_t - 2 \cdot O_t + E_t \tag{17b}$$

$$\forall t \in T: \quad E_t \cdot R_t = 1 \tag{17c}$$

$$\forall t \in T: \quad E_t = \sum_{r \in R} x_r \cdot \lambda(t,r) \cdot \Delta \tag{17d}$$

$$\forall r \in R: \quad x_r \geq x_{\text{var}} \cdot y_r \tag{17e}$$

$$\forall r \in R: \quad x_r \leq x_{\text{var}} + 10^{10} \cdot y_r \tag{17f}$$

Similarly to the previous fault detection method, the test has to be repeated to rule out alternative solutions.

V. Experimental Results

In this section the experimental results are presented. All experiments were carried out on an Intel Core i5 with 2.6 GHz and 4GB RAM. For the methods that required an ILP solver, Gurobi [13] was used. In the following, the implementation details and the test cases will be presented. The results will be discussed afterwards.

Implementation and Test Case Modeling. Expected test failure rates for all resources are represented as λ-values of a Poisson distribution and implemented as a Λ-matrix, as defined in Section III. To specify a stressed resource, its λ-value is weighted with a factor s in the range of 10^3, which is equivalent to an increase of its initial fault rate. The implementation of the application, which consists of interdependent tasks, and architecture, which consists of interconnected resources, is based on a previously presented graph-based model [14]. Figure 3 shows an exemplary system specification with two functions and five resources as well as the corresponding expectation matrix. In this example, resource $r3$ is considered faulty, which affects the highlighted λ-values in the expectation matrix.

240 tests cases of realistic sizes and topologies have been generated. These test cases comprise various system specifications with 10-100 resources and 3, 5 and 10 tasks executed on each resource, respectively. The ratio of tests to resources varies between $\frac{1}{4}$ and 1. For each test case, the observation time was as well varied. Here, we assume that only half of the test cases have actually a stressed resource to detect also false positive results.

Results. All test cases have been applied to the fault detection methods presented in Section IV with each being subject to three different values of the parameters x_{var} in case of the two ILP approaches and $\epsilon_{\cos}$ and ϵ_{svd} in case of the two linear approaches. Table I shows the parameter values used for the tests.

First, we investigate the absolute detectability in Figure 4. We make the distinction between *correct*, *false positive*, *false negative*, and *timeout*. *Correct* results include proper detection of stressed resources as well as the right detection of systems without a stressed resource. In the case of a *false positive* result, a different resource than the stressed one has been detected or one or several other resources, including the stressed one, have been found. A test result is *false negative* when a stressed resource has not been detected. A *timeout* aborts the run after 60 seconds marking the corresponding test case.

Figure 5 visualizes the test runtimes. Each cross stands for the computation time of one single test case where the lowest

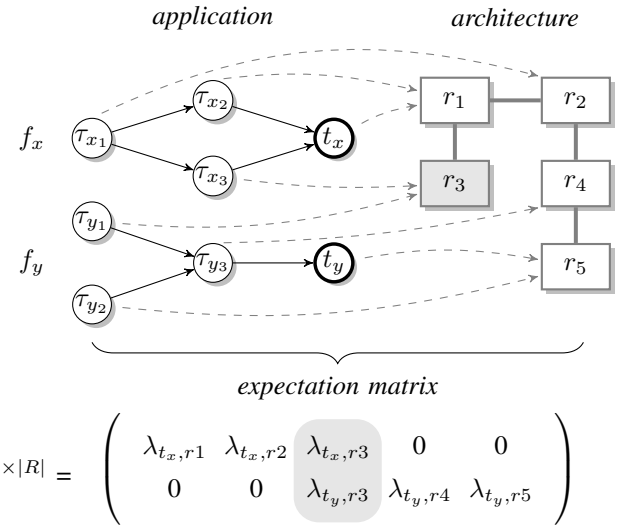

Fig. 3. The figure illustrates an exemplary systems specification consisting of the application functions (f_x, f_y), the resources (r_1 - r_5) and mappings between tasks and resources (⇢). The expectation matrix derived from the specification highlights the affected λ-values of the faulty resource $r3$.

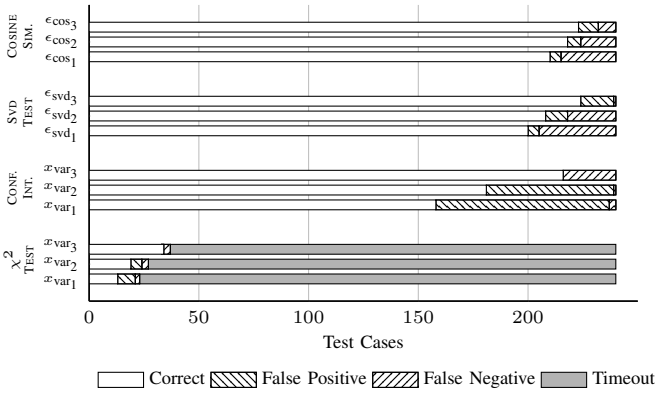

Fig. 4. The graph compares the absolute numbers of the test case results. Each bar represents the result for one fault detection method with one specific parameter value. False positive and false negative results refer to test cases which could not be solved correctly. The timeout for finding a solution for one single test case was set to 60 seconds.

TABLE I
List of parameter values used for the tests

	Value 1	*Value 2*	*Value 3*
$\epsilon_{\cos}$	$3.0 \cdot 10^{-3}$	$6.0 \cdot 10^{-3}$	$1.2 \cdot 10^{-2}$
ϵ_{svd}	$4.3 \cdot 10^{-5}$	$8.50 \cdot 10^{-5}$	$1.70 \cdot 10^{-4}$
x_{var}	1.1	1.5	1000

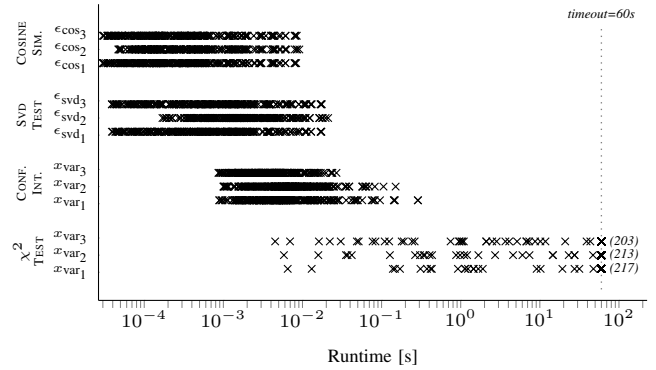

Fig. 5. The figure shows the computation times of each test case with a timeout at 60 seconds. The numbers of runs that are aborted due to a timeout are annotated in brackets.

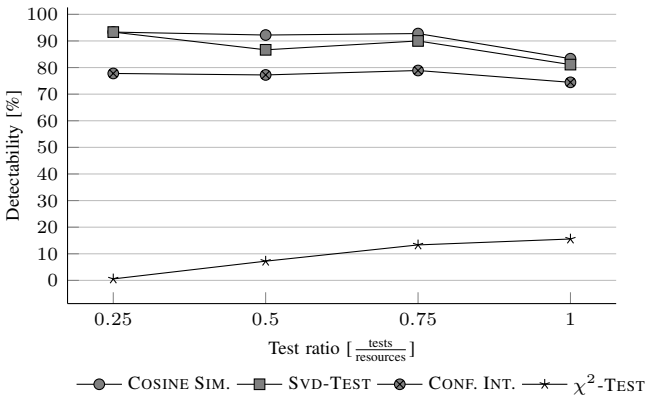

Fig. 6. The graph illustrates the percentage of detectability for each method depending on the relative number of tests.

values are in the range of microseconds and the timeout is set to 60 seconds.

Finally, the detectability of the system with respect to the relative number of tests is shown in Figure 6. Here, the y-axis represents the number of correct test results as percentage values and the x-axis marks four points with different test to resource ratios between 0.25 and 1. For this plot, the results of all three parameter values have been combined.

Discussion. The presented test case results clearly give evidence of the feasibility of our intermittent fault detection approach. The *Cosine Similarity* and the *SVD-Test* show a very good overall detection rate when using an appropriate tolerance interval ϵ. On the other hand, the *Confidence Interval* method can achieve similar detectability results with the right choice of the decision limit x_{var}. The results from the χ^2-*Test* look promising, when not taking the timeout into account. It is expectable that it would deliver detection rates comparable to the other methods or even better with a higher timeout for finding the solution.

The computation time of each test is in a millisecond range for the two linear approaches and still in a sub-second range for the *Confidence Interval* method. The runtime for the χ^2-*Test* might be reduced significantly by using a non-linear solver, as this would eliminate the additional computationally expensive constraint for the expectation variable E_t. Additionally, it should be noted that the runtime does not have to support any detection within real-time since already the observation time of a system can be rather long and compared to that the presented methods are very fast.

When splitting the results with reference to different tests to resources ratios all methods except for the χ^2-*Test* show a good detectability in a high two-digit percentage range with only small deviations between the different ratios. The decline of the linear methods curves at the $1:1$ ratio derives from a lightly increased number of false positive detections, when many tests are applied, and might be reduced by adapting the parameter values accordingly. As for the χ^2-*Test*, a higher number of tests seems to improve the detectability. However, it must be considered that this correlation might be distorted due to the limited number of test results caused by the timeout, which is also responsible for the low detection rate. It therefore appears, that the choice of the right parameter values has a greater impact on the fault detection than the number of plausibility tests, proving that our approach works well also for a low number of tests with respect to the system size.

VI. Conclusion

In this paper, an approach was proposed that enables an early and implicit detection of resources in a distributed system affected by intermittent faults . The experimental results give evidence of the efficiency and effectiveness of the proposed methods. While we presented only results for at most one faulty resource, the methods either support or can be extended to detect also multiple faults. The investigation of multiple faulty resources is a part of the future work.

References

[1] Cristian Constantinescu. Trends and challenges in VLSI circuit reliability. *Micro, IEEE*, 23(4):14–19, 2003.

[2] Martin Lukasiewycz and Samarjit Chakraborty. Concurrent architecture and schedule optimization of time-triggered automotive systems. In *Proc. of CODES*, pages 383–392, 2012.

[3] Alexandre Amory, César Marcon, Fernando Moraes, and Marcelo Lubaszewski. Task mapping on NoC-based MPSoCs with faulty tiles: Evaluating the energy consumption and the application execution time. In *Proc. of RSP*, pages 164–170, 2011.

[4] Anup Das and Akash Kumar. Fault-aware task re-mapping for throughput constrained multimedia applications on NoC-based MPSoCs. In *Proc. of RSP*, pages 149–155, 2012.

[5] Michael Glaß, Martin Lukasiewycz, Christian Haubelt, and Jürgen Teich. Incorporating graceful degradation into embedded system design. In *Proc. of DATE*, pages 320–323, 2009.

[6] Dhiego Silva, Letícia Bolzani, and Fabian Vargas. An intellectual property core to detect task schedulling-related faults in RTOS-based embedded systems. In *Proc. of IOLTS*, pages 19–24, 2011.

[7] Olivier Héron, Julien Guilhemsang, Nicolas Ventroux, and Alain Giulieri. Analysis of on-line self-testing policies for real-time embedded multiprocessors in DSM technologies. In *Proc. of IOLTS*, pages 49–55, 2010.

[8] Yuan Xie, Lin Li, Mahmut Kandemir, Narayanan Vijaykrishnan, and Mary Jane Irwin. Reliability-aware co-synthesis for embedded systems. In *Proc. of ASAP*, pages 41–50, 2004.

[9] Suleyman Tosun, Nazanin Mansouri, Ercument Arvas, Mahmut Kandemir, and Yuan Xie. Reliability-centric high-level synthesis. In *Proc. of DATE*, pages 1258–1263, 2005.

[10] Michael Glaß, Martin Lukasiewycz, Thilo Streichert, Christian Haubelt, and Jürgen Teich. Reliability-aware system synthesis. In *Proc. of DATE*, pages 409–414, 2007.

[11] Veerle Desmet, Yiannakis Sazeides, and Costas Vrioni. Performance implications of hard-faults in non-architectural structures. In *Proc. of RAAW*, 2007.

[12] Andreas Herkersdorf, Michael Engel, Michael Glaß, Jörg Henkel, Veit B Kleeberger, et al. Cross-layer dependability modeling and abstraction in system on chip. In *Proc. of SELSE*, 2013.

[13] Gurobi Optimizer. Gurobi 5.0. http://www.gurobi.com/.

[14] Martin Lukasiewycz, Michael Glaß, Christian Haubelt, and Jürgen Teich. Efficient symbolic multi-objective design space exploration. In *Proc. of ASP-DAC*, pages 691–696, 2008.