Multi-Objective Diagnosis of Non-Permanent Faults in Many-Core Systems

Peter Waszecki[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*—Due to advancing technological processes, many-core systems integrate an ever-growing number of cores on one silicon die. At the same time, shrinking circuit geometries cause a higher susceptibility for hardware faults. This paper proposes a novel approach to detect defective cores in a many-core system which are showing an increased occurrence of intermittent faults. In contrast to 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 diagnosis allows the use of precautionary measures before a permanent fault can durably damage a component in a many-core system. In this paper, we present a multi-objective approach that can implicitly detect an affected core by diagnosing its intermittent faults and taking distributed applications and their dependencies into account. The implicit approach allows the waiving of explicit tests which considerably reduces the number of plausibility test functions and, thus, leads to a saving of resource load. We propose four different implementations of our early fault diagnosis which are compared and evaluated in terms of runtime and detectability. The experimental results give evidence of the feasibility and good scalability our approach.**

I. Introduction

Today, the multi- and many-core paradigm is not an exclusive characteristic of servers or workstations any more. Due to a higher performance at lower energy consumption, CPUs with multiple cores find their way into embedded System-on-a-Chip (SoC) architectures and, thus, into domains where safety aspects are of high relevance such as avionics and automotive. However, to fulfill the corresponding reliability requirements of safety-critical applications, efficient fault detection and fault tolerance mechanisms are inevitable. At the same time, Very Large Scale Integration (VLSI) processes with shrinking geometries and decreasing 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 many-core systems, a failure of one core can influence the behavior of the running application and, in the worst-case, lead to the failure of the entire system. It is therefore desirable to obtain knowledge about potential permanent faults before they actually happen, and apply precautionary measures, which can vary from restarting the affected task on a different (redundant) core to a controlled shutdown of the complete system. For such an early fault detection, an increased number of non-permanent faults is a suitable indicator to determine stressed cores.

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

Contributions of the paper. In this paper, we propose an approach to implicitly detect cores in a many-core system that show an increased number of non-permanent faults, i.e., the presence of so-called *intermittent* faults shall be verified. Assuming that in the course of the aging process deficient hardware causes the presence and accumulation of intermittent faults, a potential imminent permanent fault of a specific core can be projected by analyzing the results of a set of plausibility tests[1] running within regular tasks or as discrete applications. A major goal of the proposed approach is to perform such a detection implicitly in order to keep the additional resource utilization low. In contrast to an explicit detection which would require additional tests for each component, the proposed fault detection relies on existing plausibility tests being part of the many-core applications. Moreover, in a heterogeneous architecture we might often only be able to check the consistency of test results on specific cores. For this purpose, we will take the distribution of applications, the runtimes of tasks, and their data-dependencies into account to implicitly determine the core that shows an increased number of intermittent faults.

We propose four different implementations for an implicit intermittent fault detection in many-core systems. Two of these methods analyze linear dependencies within the system model and the other two use Integer Linear Programming (ILP) formulations to model confidence intervals or statistical tests, respectively. Additionally, the number of plausibility tests needed for the fault detection shall be as low as possible in order to, ideally, only use tests designed for the applications.

As the number of cores in many-core systems quickly increases and might reach ranges of 10^3 and higher within the next decade [2], the scalability of our approach is crucial. Based on this, we aim at optimizing two conflicting objectives: on the one hand, our method shall improve the detectability of faulty cores and on the other hand, it shall reduce the runtime of the detection. In the experimental results, we show the general feasibility of the proposed approach and compare the methods in terms of runtime and success rate of the detection. For the sake of simplicity, we consider a system with at most one stressed core in this paper. However, ILP approaches innately support the detection of multiple stressed resources and linear dependencies approaches might be adapted appropriately. Consequently, the extension of our approach to a concurrent detection of multiple faulty cores is part of future work.

[1] In the following, *test* and *plausibility test* refer to the plausibility tests as part of the system to be analyzed, whereas *test case* and *test run* refer to the different system specifications used for the experimental results.

Organization of the paper. The remainder of the paper is organized as follows: Section II discusses the related work. Section III proposes the system model, including a formal definition of the problem. The detection methods are described in Section IV and the experimental results are presented and discussed in Section V. Section VI concludes the paper.

II. Related Work

In the considered many-core systems, one core can have several periodic tasks assigned to it, but it only executes one task at a time. Nevertheless, the tasks might have data-dependencies across different cores, i.e., by using a Network-on-Chip (NoC) or shared memory for communication. Thus, efficient mapping of tasks to cores is an important factor to satisfy high performance and safety requirements. A survey and categorization of mapping methodologies for multi- and many-core systems is presented in [3], where the analysis differentiates between design-time and run-time optimization methods. In [4], static task mapping for embedded many-core SoCs is using an ILP-based and a greedy algorithm-based approach, respectively, in order to find the optimal number of cores for each task. To increase the reliability of a system, the most common fault tolerance strategies are, on the one hand, re-mapping of tasks as discussed in [5] and [6] for the use of NoC-based Multiprocessor System-on-Chips (MPSoCs). On the other hand, when intact cores are not able to take over tasks of the affected cores, a graceful degradation mechanism is necessary as for example described in [7]. Our approach is set at the level of fault detection and, thus, before potential fault tolerance mechanisms come into action. For that reason, it builds on an existing schedule and a given distribution of tasks to system cores.

A number of approaches have been presented for analyzing the reliability and providing fault diagnosis in distributed systems. In [8] three on-line self-testing policies for multi-processors are investigated in terms of performance and detection probability. Approaches to the problem on a system level are proposed in [9] and [10], 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, [11] considers not only reliability but also all other design objectives simultaneously. In [12], a lifetime reliability estimation of homogeneous many-core systems is presented, which analyzes different configurations and redundancy schemes. However, that paper mainly regards system faults that manifest in permanent faults.

None of these approaches considers the problem that an accumulation of transient faults might finally lead to a permanent fault of a resource or, in the worst case, the entire system. Our work analyzes and compares test failure rates in order to implicitly detect cores in a many-core system tending to fail through an increase of intermittent faults.

III. System Model

Our system model regards NoC-based many-core architectures on a higher level of abstraction as a number of interconnected resources processing tasks which are capable of exchanging data across the resources. In general, a resource, i.e., a core in a many-core system, can be affected by two types of faults: permanent and non-permanent. Permanent faults can durably damage the corresponding resource and non-permanent faults, which can further be divided into transient and intermittent faults only temporarily affect the system [13]. 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]. Similar but not equal to intermittent faults are transient faults, which are mostly caused by temporary environmental phenomena, like cosmic rays, electromagnetic interference, electrostatic discharge or radiation from lightning.

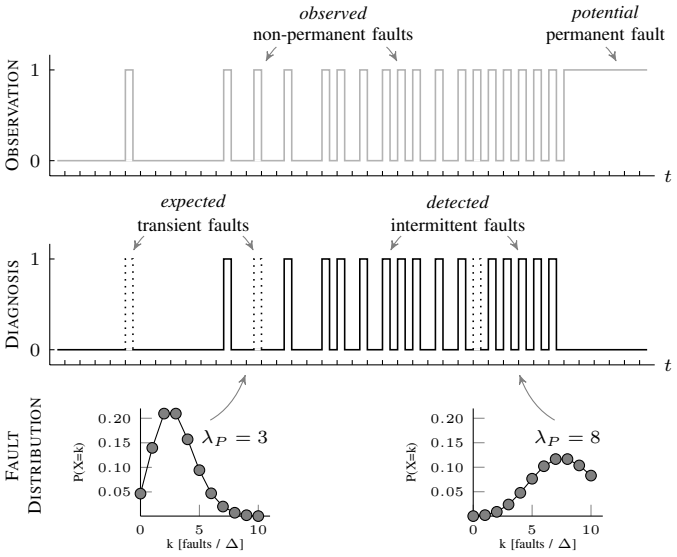

Fig. 1. The figure illustrates the occurrence of non-permanent faults on a resource, e.g., one core in a multi-core system. At first the resource is affected by transient faults only. Then, gradually, more and more intermittent faults occur which finally result in a permanent fault. The probabilistic distribution of the intermittent faults depicts an early ($\lambda_P = 3$) and a late ($\lambda_P = 8$) stage of the fault period.

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 detection of intermittent faults to avoid additional testing overhead. The occurrence of these faults can be detected with the help of plausibility tests which are evaluating the outcome of particular 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 transient single- or multi-bit flip [14]. 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 a fault in a preceding task originally caused by an intermittent fault on a resource. As, ideally, our approach shall only use already existing plausibility tests, the overhead introduced by additional tests is not considered within the scope of this paper, but might be taken into account for future work.

To model the probability of fault occurrence, the Poisson distribution is used. However, our approach is flexible enough to adopt other probabilistic fault models if required. As shown in Figure 1 a mere observation of faults affecting a resource allows no conclusion about their origin. However, with knowledge about the probabilistic distribution of the faults, it is possible to split the non-permanent faults on a resource 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 the detection of false positive faults.

Unlike in single-core architectures, the system performance in many-core systems arises from the number of cores rather than their complexity, which results in an absence of hardware-based fault tolerance mechanisms on each core [15]. Consequently, it becomes inevitable to implement this functionality in software, for which the computational overhead should be kept as low as possible. We use an implicit approach to determine task failures and, thus, resources with an increased number of intermittent faults. For this purpose, we run a number of plausibility tests at the end of particular task sequences and compare the expected and observed results of several tests. Ideally, this shall give us the right conclusions about the fault-causing resource. In a motivating example in Figure 2, an excerpt of four cores in a larger NoC-based many-core system is shown, where an increased rate of intermittent faults occurs on $core_0$. The mapping of eight application tasks τ_{x_i/y_i} and two tests $t_{x/y}$ to the four cores is indicated by the gray background areas and dashed arrows. Depending on the particular assignment and utilization of the tasks, a higher rate of intermittent faults on one core can be detected by analyzing the failure ratio of the tests. Given an equal utilization of the cores by all application tasks, a consistent fault propagation towards the tests, and a sufficient observation time, a faulty $core_0$ 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 $core_1$, whereas the occurrence of failures of just one test t_x or t_y shows a faulty $core_2$ and $core_3$, respectively.

In our system model, we consider a resource as one core in a many-core architecture. However, by adapting the expected fault rates, our fault diagnosis can also be used for other system granularity levels where a resource is regarded, for example, as a whole tile comprising a computation core, cache memory and router in a NoC or, in a fine grained model, just one of these elements.

Formal Problem Definition In the following, a formal definition of the problem is presented, which is mainly based on the sets listed below.

T	set of available tests
R	set of resources (i.e., cores) to be considered
$\mathcal{T}_r$	set of periodic tasks on a resource $r \in R$
e_τ	execution time of a task $\tau \in \mathcal{T}_r$
h_τ	period of a task $\tau \in \mathcal{T}_r$
O_t	observed failures of a test $t \in T$ in Δ
E_t	expected failures of a test $t \in T$ in Δ
Δ	time interval for a test failure observation

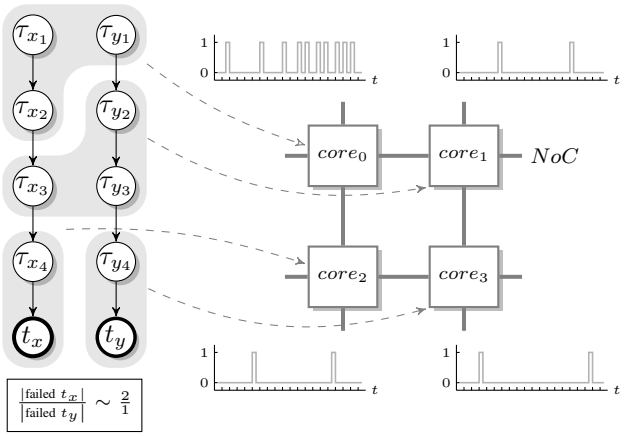

Fig. 2. The example illustrates the principle of our detection approach on four cores of a NoC-based many-core system. The application tasks τ_{x_1}, τ_{x_2}, τ_{y_1} are mapped to $core_0$ and τ_{x_3}, τ_{y_2}, τ_{y_3} are mapped to $core_1$, whereas the remaining τ_{x_4} and t_x as well as the τ_{y_4} and t_y are assigned to $core_2$ and $core_3$, respectively. The increased intermittent fault rate on $core_0$ results in a test failure ratio between t_x and t_y of $2:1$.

To detect a faulty resource in a non-trivial system with less tests than resources, as illustrated in the example in Figure 2, mere quantitative analysis of plausibility test failures is not sufficient. We regard the fault rate (i.e., number of faults per time interval) for a specific resource as being subject to the Poisson distribution with a known average value (see Figure 1). In Equation (1) it is defined as an independent random variable X_r following 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 hardware description. In the scope of this work $E[X_r]$ shall be assumed as a known value.

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

Furthermore, for each test $t \in T$ and each resource $r \in R$, we introduce a $\lambda(t, r)$ which represents the number of expected test failures per time interval which are caused by the respective resource. 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). The equation indicates whether the faulty resource runs any tasks whose failure would lead to a failure of test t which can be the test itself, or any of its predecessors. Here, $pred(t)$ represents the set of tasks that are predecessors of test t.

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

Finally, 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, as defined in Equation (3).

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

According to our fault diagnosis approach, Λ describes the expected fault rates under normal operation for transient faults only. In case of intermittent faults, the observations will deviate from this matrix and allow an implicit fault detection as proposed in Section IV.

IV. Fault Diagnosis Methods

The presented fault diagnosis is designed to indicate cores in a many-core system (also referred to as resources) experiencing an increased occurrence of intermittent faults and the goal to reach a good detectability at a reasonably low runtime leads to a multi-objective problem. As a matter of principle, the solution of this problem can be abstracted to an analysis of the quantitative relation between observed and expected test failures. Thus, based on the expectation matrix Λ introduced in Section III, four realizations of the general fault diagnosis principle shall be described and evaluated with respect to their correctness and performance. The methods are summarized in Table I and described below.

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 defined in Equation (4). The magnitude of this inequality is depending on the considered fault distribution in the tested system and was in the range of 10^3 for our test cases. This range has been chosen in order to evaluate and verify the general suitability and effectiveness of the four detection methods. In reality, the rate of intermittent faults caused by aging, rises slowly over time until a permanent fault occurs. This behavior is not considered in the experimental results and will be investigated in future work.

$$O_t \gg \Delta \cdot \sum_{r \in R} \lambda(t, r) \tag{4}$$

A. Method I

For the first two detection methods, 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 (5a))[2]. In addition, an observation vector $\mathbf{v}_{O_T}$ which consists of the observed failures of each plausibility test is used (Equation (5b)).

$$\mathbf{v}_{r_i} = [\lambda(t_1, r_i), \lambda(t_2, r_i), \cdots, \lambda(t_m, r_i)]^T \tag{5a}$$
$$\mathbf{v}_{O_T} = [O_{t_1}, O_{t_2}, \cdots, O_{t_m}]^T \tag{5b}$$

TABLE I. Four implementations of our fault diagnosis

Method	*Implementation*	*Test principle*	*Section*
I	Vector-based	Cosine Similarity	IV-A
II	Vector-based	Singular Value Decomposition	IV-B
III	ILP-based	Confidence Interval	IV-C
IV	ILP-based	Pearson's χ^2-Test	IV-D

Based on this, a cosine similarity analysis between the observation vector and the expectation vectors can indicate, if the corresponding resource is faulty. The closer the cosine value is to 1, the higher is the certainty that the observed faults stem from the resource belonging to the expectation vector. This is shown in Equation (6), where θ is the angle enclosed by the examined vectors and ϵ_{cos} is the maximum acceptable deviation from 1.

$$|1 - cos(\theta)| \begin{cases} \leq \epsilon_{\text{cos}} \Rightarrow O_T \text{ from } r_i, & \theta = \angle(\mathbf{v}_{O_T}, \mathbf{v}_{r_i}) \\ > \epsilon_{\text{cos}} \Rightarrow O_T \text{ not from } r_i, & \theta = \angle(\mathbf{v}_{O_T}, \mathbf{v}_{r_i}) \end{cases} \tag{6}$$

B. Method II

Alternatively, the second method calculates the Singular Value Decomposition (SVD) of a sub-matrix $\Lambda_s = (\mathbf{v}_{O_T}, \mathbf{v}_{r_i})$ to determine the linear dependence. This is shown in Equation (7)[3], where the SVD of Λ_s results in a product of the orthogonal matrix U, the diagonal matrix Σ and the transposed orthogonal matrix V^{T}. The matrices U and V, whose columns contain the eigenvectors of $\Lambda_s\Lambda_s^{\text{T}}$ and $\Lambda_s^{\text{T}}\Lambda_s$, respectively, shall receive no consideration during the further detection procedure. Σ has the same dimension as Λ_s (i.e., $|T| \times 2$) and consists of the singular values σ_1 and σ_2 on its diagonal and 0 otherwise. The observation O_T is thought of being caused by resource r_i, if the rank of the diagonal matrix Σ is less than 2, as shown in Equation (8).

$$\Lambda_s{}^{|T|\times 2} = U\Sigma V^{\text{T}}, \text{ with } \Sigma = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} \text{ and } |T| \geq 2 \tag{7}$$

$$rank(\Sigma) \begin{cases} < 2 \Rightarrow O_T \text{ from } r_i, & \text{if } \exists\sigma \leq \epsilon_{\text{svd}} \\ = 2 \Rightarrow O_T \text{ not from } r_i, & \text{if } \forall\sigma > \epsilon_{\text{svd}} \end{cases} \tag{8}$$

The performance and correctness of Methods I and II are highly dependable on the proper choice of the tolerance interval ϵ. A too big value is not sensitive enough and may cause false positive detections, whereas a too small value would not consider the noise caused by transient faults and might lead to false negative detections. 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

Considering the non-deterministic nature of fault occurrences, a fundamentally different detection approach can be applied. 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

[2]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.

[3]Note, that T represents the transpose of a matrix and should not be confused with $|T|$ which stand for the number of tests in the set T.

standard deviations from the expected value E_t, we define the interval according to the 3σ-rule as seen in Equation (9). Here, σ represents the standard deviation and μ the mean of the Poisson-distributed variable x.

$$P_r(\mu - 3\sigma \leq x \leq \mu + 3\sigma) \approx 0.9973 \tag{9}$$

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 (10a) and (10b). Accordingly, this leads to a confidence interval $[\lambda_{lo}, \lambda_{hi}]$ which comprises the expected test failures E_t during the observation time Δ.

$$\underset{\lambda_{lo} \in \mathbb{R}_0^+}{\text{minimize}} \ \lambda_{lo}, \quad \text{subject to} \ \ F_{\lambda_{lo}}(x \leq O_t) \leq \frac{P_r + 1}{2} \tag{10a}$$

$$\underset{\lambda_{hi} \in \mathbb{R}_0^+}{\text{maximize}} \ \lambda_{hi}, \quad \text{subject to} \ \ F_{\lambda_{hi}}(x \leq O_t) \geq \frac{1 - P_r}{2} \tag{10b}$$

Introducing a stress factor x_r allows us to formulate an optimization problem with the confidence interval as one constraint (Equation (11b)). The expected test failures E_t are defined as sum of weighted λ-values over all resources, where the weight factor is x_r (Equation (11c)). We use y_r as switch variable to decide whether x_r is significantly large ($y_r = 1$) or still in an acceptable range ($y_r = 0$) in order to keep E_t inside the confidence interval (Equations (11d) and (11e)). The threshold for this decision is set by the variable d_r. Based on these constraints, the objective function searches for cases where a minimal number of stressed resources is responsible for observed test failures (Equation (11a)).

$$\underset{y_r \in \{0,1\}}{\textit{minimize}} \quad \sum_{r \in R} y_r \tag{11a}$$

subject to:

$$\forall t \in T: \quad \lambda_{lo}(O_t) \leq \frac{E_t}{\Delta} \leq \lambda_{hi}(O_t) \tag{11b}$$

$$\forall t \in T: \quad E_t = \sum_{r \in R} x_r \cdot \lambda(t, r) \cdot \Delta \tag{11c}$$

$$\forall r \in R: \quad x_r \geq d_r \cdot y_r \tag{11d}$$

$$\forall r \in R: \quad x_r \leq d_r + 10^{10} \cdot y_r \tag{11e}$$

The optimization algorithm considers $\lambda(t, r)$ and O_t to be constants from the set of positive rational numbers $\mathbb{R}_0^+$. Similarly, the variables E_t and x_r contain values from $\mathbb{R}_0^+$, whereas y_r is defined in $\{0, 1\}$, respectively. The threshold d_r is crucial for the goodness of the fault diagnosis so that several different values have been selected for the experimental results. 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

The fourth implemented method 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 (Equation (12)).

$$\chi^2 = \sum_{t \in T} \frac{(O_t - E_t)^2}{E_t} \tag{12}$$

The null hypothesis H_0 for the test is defined in Equation (13). It indicates that all *observed* test failures O_t result from the *expected* test failures E_t.

$$H_0 : E_t \rightarrow O_t \tag{13}$$

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 (14) denotes this inequality where $F(\chi^2, |T| - 1)$ is the CDF of the χ^2-distribution with $|T| - 1$ degrees of freedom.

$$1 - F(\chi^2, |T| - 1) \geq 0.05 \tag{14}$$

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 are trying 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 (15b) and (15c) for the optimization problem formulated below.

$$\underset{y_r \in \{0,1\}}{\textit{minimize}} \quad \sum_{r \in R} y_r \tag{15a}$$

subject to:

$$\forall t \in T: \quad \chi^2 = \sum_{t \in T} O_t^2 \cdot R_t - 2 \cdot O_t + E_t \tag{15b}$$

$$\forall t \in T: \quad E_t \cdot R_t = 1 \tag{15c}$$

$$\forall t \in T: \quad E_t = \sum_{r \in R} x_r \cdot \lambda(t, r) \cdot \Delta \tag{15d}$$

$$\forall r \in R: \quad x_r \geq d_r \cdot y_r \tag{15e}$$

$$\forall r \in R: \quad x_r \leq d_r + 10^{10} \cdot y_r \tag{15f}$$

Similarly to Method III, the test has to be repeated to rule out alternative solutions.

V. Experimental Results

In the following, the implementation details and test cases are explained, followed by a comprehensive presentation and discussion of experimental results. All experiments were carried out on an Intel Core i5 with 2.6 GHz and 8GB RAM. For the ILP-based optimization, GUROBI [16] was used as solver.

Implementation and Test Case Modeling. The expected test failure rates for cores in a many-core system are represented as λ-values of a Poisson distribution and implemented in a Λ-matrix, as defined in Section III. To specify a stressed core, its λ-value is weighted with a factor s in the range of 10^3, which is equivalent to an increase of its initial (transient)

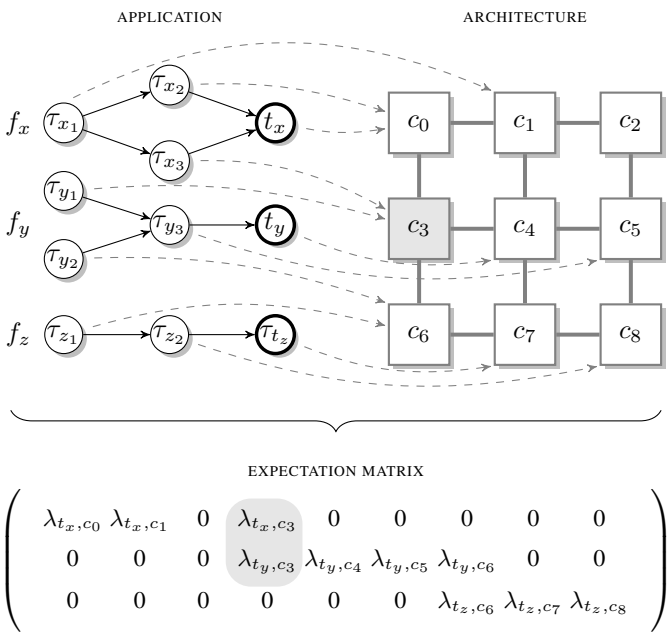

Fig. 3. The figure illustrates an exemplary system specification consisting of three application functions (f_x, f_y, f_z), cores (c_0 - c_8) and mappings between tasks and cores (⇢). The expectation matrix derived from the specification shows weighted λ-values for the faulty core c_3.

fault rate during correct operation. The implementation of the application, which consists of interdependent tasks, and the architecture, which consists of interconnected cores, is based on a previously presented graph-based model [17]. Figure 3 shows an exemplary system specification with three functions and a many-core system with nine cores as well as the corresponding expectation matrix. In this example, core c_3 is considered faulty, which affects the highlighted λ-values in the expectation matrix.

Based on the system implementation described above, 240 test cases of realistic sizes and topologies have been generated with architectures ranging from 3x3 to 48x48 cores. Thus, the test cases comprise various system specifications with 9-2304 cores executing 3, 5 and 10 tasks on a resource on average, respectively. Here, the ratio of plausibility tests to cores varies between $\frac{1}{2}$ and $\frac{1}{16}$. For each test case, the observation time was as well varied and we assume that only half of the test cases have actually a stressed core in order to detect also false positive results.

The four fault detection methods presented in Section IV have been applied to all test cases in three different sensitivity levels each. For this purpose, the values of the parameters ϵ_{cos} and ϵ_{svd}, in case of the two linear approaches, and d_r, in case of the ILP approaches, have been varied. Table II shows the parameter values used for the test cases.

Results and Discussion. First, we investigate the absolute detection rates of all methods, which are illustrated in Figure 4. Here, according to the labels on the y-axis, each bar represents one detection method with one specific parameter value. As each test outcome can be assigned to one of four different results, the bars are subdivided into four segments: *correct*, *false positive*, *false negative* and *timeout*, which correspond to

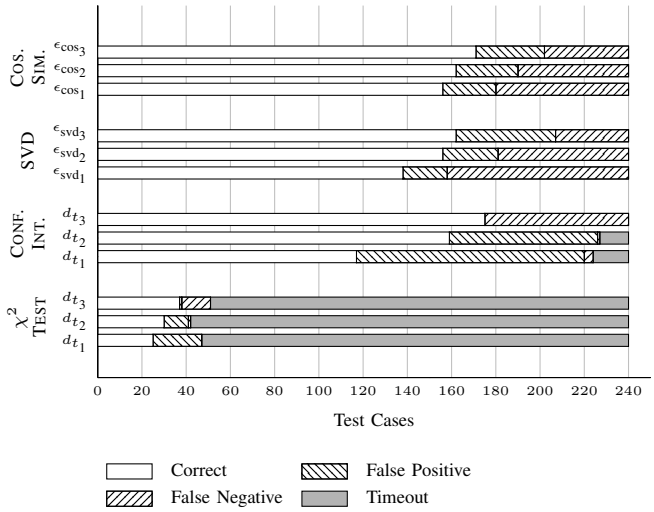

Fig. 4. The graph compares the numbers of different test case results with each bar representing the outcome for a fault detection method with one specific parameter value. False positive and false negative results comprise test cases which could not be solved correctly. The timeout for finding a solution for one single test case is 60 seconds.

a number of test cases between 0 and 240, as denoted on the x-axis. *Correct* results include both the detection of a stressed core and the proper diagnosis of a system without stressed cores. In contrast, a *false positive* result indicates the detection of a different than the stressed core as well as the detection of several stressed cores, including the requested one. Finally, for a *false negative* result, the actually stressed core has not been detected. When a test case run has not been able to find an unambiguous solution after 60 seconds, it is aborted and marked as *timeout*.

Inspecting the absolute numbers of the test case results shows a clear superiority of the two vector-based implementations as well as the *Confidence Interval* with at least half of all test cases solved correctly. However, the results of the χ^2-*Test* look comparably or even better, when not taking the timeout into account, which makes this implementation still interesting when used for an offline analysis on high-performance platforms. Regarding the different parameter values, the *Cosine Similarity* and the *SVD* show an increase of false positive results with higher ϵ_{cos} and ϵ_{svd}, respectively. This is, due to an increased tolerance scope for the comparison of the observed and expected vectors when the thresholds are higher. As, on the other hand, with a lower threshold the number of false negative results is increasing, these parameters must be chosen carefully, as already mentioned in Section IV-B. For the ILP-implementations this correlation is reversed and a higher d_r leads to more false negative detections, as this threshold proportionally mirrors the level of the assumed system noise.

TABLE II. LIST OF THE PARAMETER SETS USED BY THE DETECTION METHODS

Method	Parameter	*Set 1*	*Set 2*	*Set 3*
I	ϵ_{cos}	$3.0 \cdot 10^{-3}$	$6.0 \cdot 10^{-3}$	$1.2 \cdot 10^{-2}$
II	ϵ_{svd}	$4.3 \cdot 10^{-5}$	$8.50 \cdot 10^{-5}$	$1.70 \cdot 10^{-4}$
III & IV	d_r	1.1	1.5	1000

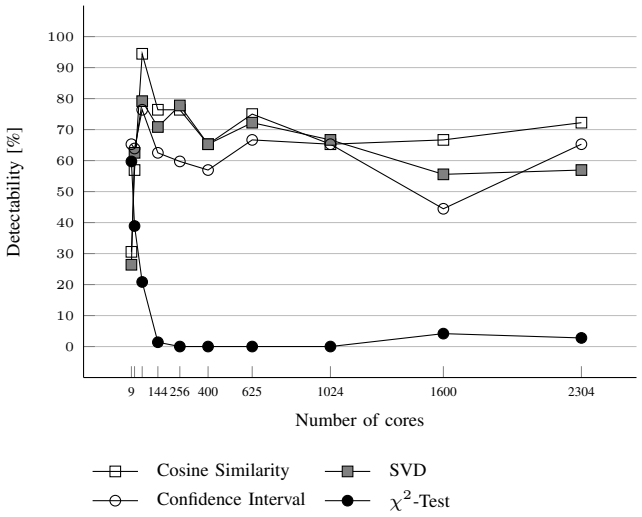

Fig. 5. Detectability as a function of the system size. The outcome of each method combines the results of all three parameter values.

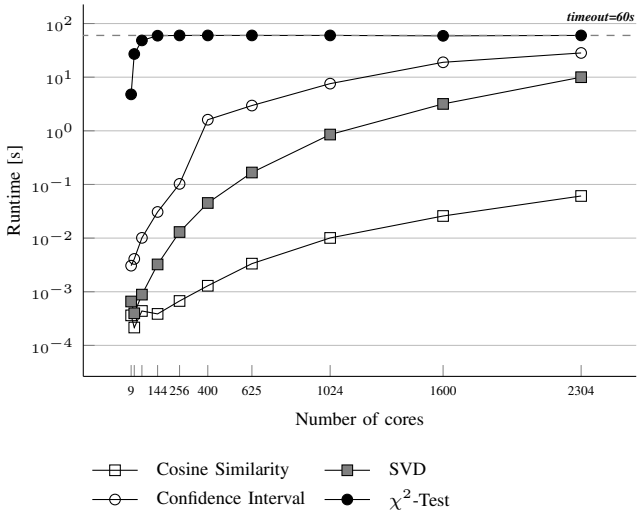

Fig. 6. Average test case runtime as a function of the system size. The outcome of each method combines the results of all three parameter values.

Besides the absolute test case results, we are interested in the scalability of our approach and, hence, in the behavior of the implementations with respect to different system sizes. Figure 5 illustrates the detectability as a function of the number of cores for all four methods, where the results from all three parameter values are combined. The detectability represents the percentage number of correct test results only and does not include false positive results.

Interestingly, the *Cosine Similarity* and the *SVD* which are both showing good results for large system sizes are outperformed by the ILP-based methods when only few cores are diagnosed. Here, the slightly scattered and unsteady developing of the result values towards higher core numbers might be improved by a system size specific choice of the parameter values. In general, all implementations except for the χ^2-*Test* provide a good detectability which lies clearly above 50% also for large core numbers. Regarding the χ^2-*Test*, the almost

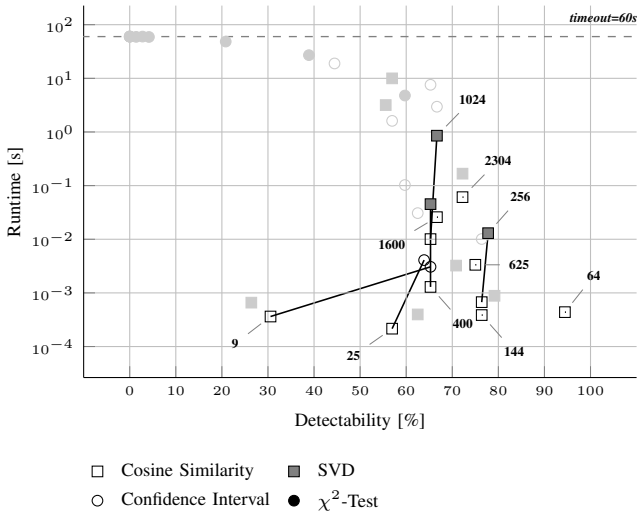

Fig. 7. Illustration of Pareto efficient points with respect to a high detection rate at low test case runtimes. The pinned labels show the core numbers considered by the particular results and two connected points represent a Pareto-front. The results for the different parameter values are combined.

complete lack of detectability for more than 144 cores is caused by the timeout, which can clearly be seen in Figure 6. In this graph, the test case runtime on the y-axis is displayed in a logarithmic scale, which indicates very short computational times for the *Cosine Similarity* for all system sizes and still fair results for the *SVD* and *Confidence Interval*. However, both of them are approaching the timeout limit for high core numbers.

As mentioned before, in our work we are interested in both improving the detectability and reducing the runtime of the diagnosis. To analyze the Pareto efficiency of these two objectives, we combine the results from the two previous graphs in Figure 7. Here, the black markers represent optimal results for a particular system size, with the number of cores pinned to the corresponding marker. In the case, where two detection methods are optimal for two different objectives, their markers are connected and the resulting line forms the Pareto-front for the corresponding system size. All grayed out points are dominated by the black ones and, hence, Pareto inefficient.

For the most system sizes, *Cosine Similarity* seems to be the best implementation method, although the *SVD* shows slightly better detection rates for 256, 400 and 1024 cores. When considering small systems with 9 and 25 cores, than the *Confidence Interval* is the better choice, as in this time range the difference to the vector-based method is negligible.

Finally, as our detection approach is implicit, and hence, shall utilize already existing tests, we want to investigate the influence of the number of plausibility tests on the detectability. In Figure 8 the y-axis represents the percentage number of correct test results, whereas the x-axis shows the tests-to-cores ratio. More precisely, the four values between 2^{-4} and 2^{-1} stand for one test per 16 cores, 8 cores, 4 cores, and 2 cores, respectively. Like for the previous three plots, the results for all parameters are combined in this graph.

As one might expect, the detectability is increasing with a higher tests-to-cores ratio for all methods except for the χ^2-

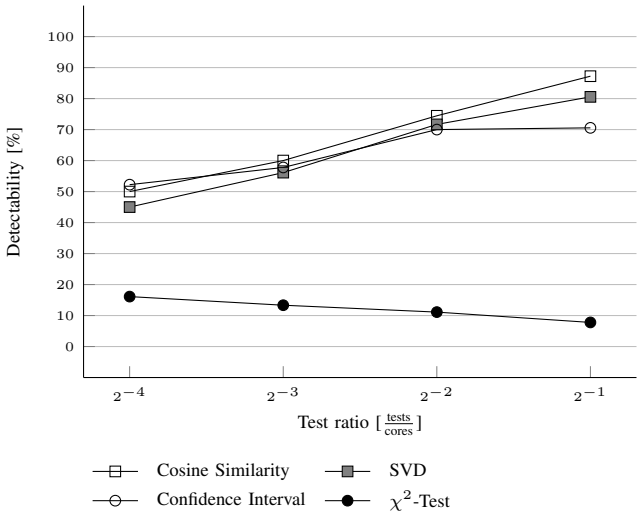

Fig. 8. Detection rate as a function of the tests-to-cores ratio. A higher number on the x-axis indicates that more tests have been used for the fault detection.

Test, where this correlation is reversed. The reason is that for a higher number of tests the problem becomes more complex and, thus, more χ^2-*Test* test cases run into the timeout, which is also responsible for the overall low detectability of this method. For the remaining three implementations, however, we can see that even a relatively low number of plausibility tests enables the fault detection to still being able to find a correct solution for every second test case.

Summary. In summary, we see that the experimental results clearly give evidence of the feasibility of our intermittent fault detection for many-core systems. Especially, regarding the wide range of the system sizes, the approach shows a good scalability and, hence, its suitability for SoCs with several thousand cores. While the runtimes of the detection methods are already satisfactory, when disregarding the χ^2-*Test*, it still might be possible to increase the detectability. Here, as part of future work, we want to investigate how the mapping of task and tests to different cores influences the detection and how it can be optimized.

VI. Conclusion

Today, we experience an unprecedented growth in the technological development of many-core SoCs, of which more and more are used in safety-critical domains, like automotive and avionics. However, the growing number of cores on a single die leads to new challenges in terms of system reliability. This paper proposes an approach that enables an early and implicit diagnosis of faulty cores in a many-core system. For the detection of cores affected by intermittent errors, two vector-based and two optimization-based methods are evaluated. The experimental results give evidence of the efficiency and effectiveness of the proposed approach. While we present only results for at most a single faulty core, the methods either support or can be extended to detect also multiple cores. The investigation of multiple faulty cores and optimal test distribution is a part of the future work. As a next step, we want to investigate the behavior of the detectability for different fault rates and also analyze different fault propagation models, for which an instruction set simulator shall be used.

References

[1] C. Constantinescu, "Trends and challenges in VLSI circuit reliability," *Micro, IEEE*, vol. 23, no. 4, pp. 14–19, 2003.

[2] G. Kurian, J. E. Miller, J. Psota, J. Eastep, J. Liu *et al.*, "Atac: a 1000-core cache-coherent processor with on-chip optical network," in *Proc. of PACT*, 2010, pp. 477–488.

[3] A. K. Singh, M. Shafique, A. Kumar, J. Henkel, A. Das *et al.*, "Mapping on multi/many-core systems: survey of current and emerging trends," in *Proc. of DAC*, 2013.

[4] J. Kaida, T. Hieda, I. Taniguchi, H. Tomiyama, Y. Hara-Azumi, and K. Inoue, "Task mapping techniques for embedded many-core socs," in *Proc. of ISOCC*, 2012, pp. 204–207.

[5] A. Amory, C. Marcon, F. Moraes, and M. Lubaszewski, "Task mapping on NoC-based MPSoCs with faulty tiles: Evaluating the energy consumption and the application execution time," in *Proc. of RSP*, 2011, pp. 164–170.

[6] A. Das and A. Kumar, "Fault-aware task re-mapping for throughput constrained multimedia applications on NoC-based MPSoCs," in *Proc. of RSP*, 2012, pp. 149–155.

[7] M. Glaß, M. Lukasiewycz, C. Haubelt, and J. Teich, "Incorporating graceful degradation into embedded system design," in *Proc. of DATE*, 2009, pp. 320–323.

[8] O. Héron, J. Guilhemsang, N. Ventroux, and A. Giulieri, "Analysis of on-line self-testing policies for real-time embedded multiprocessors in DSM technologies," in *Proc. of IOLTS*, 2010, pp. 49–55.

[9] Y. Xie, L. Li, M. Kandemir, N. Vijaykrishnan, and M. J. Irwin, "Reliability-aware co-synthesis for embedded systems," in *Proc. of ASAP*, 2004, pp. 41–50.

[10] S. Tosun, N. Mansouri, E. Arvas, M. Kandemir, and Y. Xie, "Reliability-centric high-level synthesis," in *Proc. of DATE*, 2005, pp. 1258–1263.

[11] M. Glaß, M. Lukasiewycz, T. Streichert, C. Haubelt, and J. Teich, "Reliability-aware system synthesis," in *Proc. of DATE*, 2007, pp. 409–414.

[12] L. Huang and Q. Xu, "On modeling the lifetime reliability of homogeneous manycore systems," in *Proc. of PRDC*. IEEE, 2008, pp. 87–94.

[13] V. Desmet, Y. Sazeides, and C. Vrioni, "Performance implications of hard-faults in non-architectural structures," in *Proc. of RAAW*, 2007.

[14] A. Herkersdorf, M. Engel, M. Glaß, J. Henkel, V. B. Kleeberger *et al.*, "Cross-layer dependability modeling and abstraction in system on chip," in *Proc. of SELSE*, 2013.

[15] F. N. Sibai, "Detecting matrix multiplication faults in many-core systems," in *Proc. of IIT*, 2011, pp. 330–335.

[16] Gurobi Optimizer, "Gurobi 5.0," http://www.gurobi.com/.

[17] M. Lukasiewycz, M. Glaß, C. Haubelt, and J. Teich, "Efficient symbolic multi-objective design space exploration," in *Proc. of ASP-DAC*, 2008, pp. 691–696.