Multi-Layer Diagnosis for Fault-Tolerant Networks-on-Chip

Schley, Gert; Dalirsani, Atefe; Eggenberger, Marcus; Hatami, Nadereh; Wunderlich, Hans-Joachim; Radetzki, Martin

IEEE Transactions on Computers Vol. 66(5) 1 May 2017

doi: http://dx.doi.org/10.1109/TC.2016.2628058

Abstract: In order to tolerate faults that emerge in operating Networks-on-Chip, diagnosis techniques are employed for fault detection and localization. On various network layers, diverse diagnosis methods can be employed which differ in terms of their impact on network performance (e.g. by operating concurrently vs. pre-empting regular network operation) and the quality of diagnostic results. In this contribution, we show how diagnosis techniques of different network layers of a Network-on-Chip can be combined into multi-layer solutions. We present the cross-layer information flow used for the interaction between the layers and show the resulting benefit of the combination compared to layer-specific diagnosis. For evaluation, we investigate the diagnosis quality and the impact on system performance to explore the entire design space of layer-specific techniques and their multi-layer combinations. We identify pareto-optimal combinations that offer an increase of system performance by a factor of four compared to the single-layer diagnosis.

Preprint

Multi-Layer Diagnosis for Fault-Tolerant Networks-on-Chip

Gert Schley, Atefe Dalirsani, Marcus Eggenberger, *Student Member, IEEE,* Nadereh Hatami, Hans-Joachim Wunderlich, *Fellow, IEEE*, Martin Radetzki, *Senior Member, IEEE*

***Abstract*—In order to tolerate faults that emerge in operating Networks-on-Chip, diagnosis techniques are employed for fault detection and localization. On various network layers, diverse diagnosis methods can be employed which differ in terms of their impact on network performance (e.g. by operating concurrently vs. pre-empting regular network operation) and the quality of diagnostic results. In this contribution, we show how diagnosis techniques of different network layers of a Network-on-Chip can be combined into multi-layer solutions. We present the cross-layer information flow used for the interaction between the layers and show the resulting benefit of the combination compared to layer-specific diagnosis. For evaluation, we investigate the diagnosis quality and the impact on system performance to explore the entire design space of layer-specific techniques and their multi-layer combinations. We identify pareto-optimal combinations that offer an increase of system performance by a factor of four compared to the single-layer diagnosis.**

***Index Terms*—Networks-on-Chip, NoC, Diagnosis, Performance, Multi-layer, Design Space Exploration.**

I. Introduction

FOR THE LAST 50 YEARS, chip integration density has increased exponentially, enabling the implementation of multi-core chips such as Intel's Xeon Phi Coprocessor [1]. As technology scaling continues from today's 16 nm feature size to 1 nm in 2028 [2], thousands of processing cores on a single chip will become feasible. With the increasing number of cores, higher communication bandwidth will be required. Conventional bus-based systems will no longer meet these requirements and would thus become a performance bottleneck. To avoid this bottleneck, Networks-on-Chip (NoC) have been proposed as communication structure for manycore systems [3].

A negative aspect of technology scaling, however, is the increasing probability of chip defects introduced during manufacturing [4] or caused during system operation by wear-out effects such as BTI, HCI or electromigration [5]. Such defects are modeled as permanent faults in the structure of a NoC. When a fault leads to a deviation of information from the fault-free case, we say that an error is activated. Different from transient errors caused by radiation or crosstalk, errors resulting from permanent faults cannot be corrected with a simple retransmission. Without proper handling, they may lead to system malfunction or even to complete system failure.

To bypass faults, they have to be located and classified as permanent. For this purpose, diagnostic techniques can be designed on various network layers from physical layer to transport layer. Each layer has its specific tradeoff between quality (granularity) and cost (overhead) of diagnosis. On the transport layer, diagnosis is implemented in software, without additional hardware cost, and is conducted concurrently with regular system operation. However, the available information is largely abstracted from the underlying hardware. Therefore, faults cannot be attributed to individual gates and wires. Granularity increases towards lower network layers at the expense of an increased diagnosis effort (e.g., the need to shut down the network for diagnosis or the cost of built-in self diagnosis hardware).

With this article, we contribute the following:

- NoC-specific diagnostic techniques for three individual layers: transport layer, network layer, and physical layer,
- the fusion of any combination of these techniques into a multi-layer approach with cross-layer information exchange,
- and the investigation of all possible combinations with respect to diagnostic quality and performance impact, so as to identify pareto-optimal solutions.

The remainder of this article is organized as follows: In Section II we present related work relevant to this contribution. The cross-layer interaction for multi-layer diagnosis is presented in Section III. Detailed descriptions of the layer-specific diagnosis techniques and their cross-layer interfaces can be found in Sections IV to VI. Evaluation results are presented and discussed in Section VII. Section VIII concludes our contribution.

II. Related Work

A. Software-Based Diagnosis

A probabilistic method for locating permanent faults in link wires based on a software protocol is presented in [6]. All data packets are equipped with an error locality aware error correction code. At receiver side each packet is checked for errors and the positions of erroneous bits are identified. The bit positions are sent to a central network node where they are accredited to the wires of the links situated on the path the packet has taken.

In [7], faulty links are localized by means of heartbeat messages, which are sent by each network node to dedicated

This work has been supported by the German Research Foundation (DFG) under grant RA 1889/4-1 (ROCK) and WU 245/12-1 (ROCK).

G. Schley, M. Eggenberger, and M.Radetzki are with the Chair of Embedded Systems, University of Stuttgart, Stuttgart 70569, Germany. (e-mail: {schley, eggenbms, radetzki}@informatik.uni-stuttgart.de).

A. Dalirsani, N. Hatami, and H.-J. Wunderlich are with the Chair of Computer Architecture, University of Stuttgart, Stuttgart 70569, Germany. (e-mail: {dalirsani, hatami, wu}@informatik.uni-stuttgart.de).

fault detection units. Because of the applied scheduling and routing, heartbeat messages are expected to arrive at a fault detection unit after a specific duration. If a heartbeat message is delayed or does not arrive at all, the corresponding path is marked as suspicious. A link is considered to be faulty if it was marked suspicious by multiple heartbeat messages.

In [8], links and switches are tested on system startup by a distributed built-in self-test mechanism and all faulty components are disabled. A software-based localization method is used to provide the exact location of faulty components to a central routing reconfiguration mechanism. Each network node tests by means of ping messages if all other nodes can be reached. If a node can be reached, the corresponding path is added to a local log. A central node collects all logs and analyzes them to determine the location of faulty components.

All the software-based diagnosis approaches mentioned above have in common that they have only a coarse-grained network view and thus permanent faults can only be tracked down to a link or switch. Consequently, fault tolerance is achieved by shutting down faulty components (e.g. [6]) and by triggering routing reconfiguration (e.g. [9]). However, the shutdown of a complete switch, which also leads to the loss of the connected processing core, may be inappropriate when the fault affects just a small part of the switch's functionality. To salvage the remaining intact functionality, diagnosis techniques with a more fine-grained diagnosis granularity such as functional diagnosis or structural diagnosis have to be used.

B. Functional Diagnosis

A framework to support functional diagnosis in NoCs is proposed in [10]. The considered functional faults are: deadlock, livelock, misrouting, and packet starvation. During a logging phase, each network node takes a snapshot containing information about the packets in the node's input buffers and the current timestamp. In the subsequent local check phase, network operation is pre-empted and the snapshots are used to identify the occurrence of functional faults. If a functional fault is found, the snapshots of all network nodes are forwarded to a central node where they are used to derive diagnostic information. Evaluations show that the observability of faults depends on the packet injection rate and the snapshot rate.

A distributed online functional diagnosis approach to detect stuck-at, bridging, and delay faults in a NoC switch is presented in [11]. Each switch is equipped with a test packet generator and a diagnosis unit. During diagnosis, test packets are sent by the local network interface and by all neighbors of a switch under test (SUT). To detect and locate faults in the data path and control path of the SUT, the diagnosis process is separated into nine phases with different test packet destinations. A token-based control mechanism ensures that only a single switch is diagnosed at each point in time.

Having determined the failure of a switch function, the usage of the affected functionality can be avoided, e.g. by adapting the routing [12]. Alternatively, one can attempt to restore that functionality, e.g. by invoking redundant components or by reconfiguring the data path [13]. However, functional diagnosis cannot attribute the misbehavior to a particular location of a structural fault but only to a set of locations that all cause this misbehavior. To find the exact location of a fault, which enables precisely directed countermeasures, structural diagnosis has to be applied.

C. Structural Diagnosis

The fine-granular fault localization of structural diagnosis is achieved by adding a dedicated test access mechanism (TAM) or built-in self-test (BIST) to the hardware design. The most widely used methodology for structural testing is scan design [14].

In [15], a scan-based TAM is proposed to test the logic and input buffers of NoC switches. The TAM's architecture corresponds to the minimal spanning tree of the network topology. Test patterns are sent by the source network node of the spanning tree via a dedicated line to all other nodes. The test responses of each switch are sent back to the source node via the same line. Another approach for NoCs that makes use of scan chains is presented in [16]. It deals with the optimized transport of patterns to scan chains using flits.

An IEEE 1500 compliant test wrapper for NoCs is proposed in [17]. The wrapper facilitates the parallel test of all switches in the network as well as diagnosis of individual switches. This work considers the complete NoC as a flat core and exploits its regularity. This results in smaller area overhead and reduced test time compared to other scan approaches.

A distributed BIST architecture for diagnosing NoCs is proposed in [18]. The goal of this work is to reuse the NoC's interconnection infrastructure as TAM and to exploit the inherent parallelism of NoCs to reduce test time. A strategy is proposed that allows test data to be produced by a global test data generator and to be transmitted to the switches by means of special test packets.

All the above structural diagnosis approaches have in common that the network has to be switched to offline mode partly or in full before diagnosis is carried out, causing service interruption and significant performance penalty. On the other hand, due to their gate level granularity, these approaches allow the most fine-grained reconfiguration on the level of individual wires [19] or switch ports [20].

D. Multi-Layer Diagnosis

Multi-layer diagnosis combines diagnosis techniques of different layers to locate hardware defects. Multi-layer approaches have been published for various kinds of systems ranging from FPGA systems to all-optical networks. They can reduce the time required for localization [21] and reconfiguration [22] by combining diagnostic information from the individual layers and may even reduce implementation overhead [23].

In [21], the benefits and challenges of bottom-up multi-layer fault localization in all-optical multi-layer networks are discussed. Considerations for designing multi-layer fault localization under the aspects of efficient fault detection and fast localization are presented.

A hardware/software integrated diagnosis technique for processors is proposed in [24]. In this approach fine-grained

hardware information is provided to the software diagnosis technique, which uses this information to diagnose intermittent faults.

Multi-layer online diagnosis for FPGA-based systems is presented in [22] and [25] to detect faults in programmable logic blocks (PLBs). On the highest layer, the set of the possibly defective PLBs is determined before a more fine-grained diagnosis is carried out on the next lower layer.

The work of [23] proposes a model for cross-layer resilience for on-chip systems featuring diagnosis. The authors state that cross-layer resilient systems have the benefit that by shifting functionality to the software layers, hardware overhead is reduced. Furthermore, using detailed information provided by the different layers helps to make optimal decisions how to handle a fault. Since diagnosis is tightly coupled to the hardware, contribution from application/software layer is stated to be difficult. However, the application can e.g. help to schedule the diagnosis depending on the number of errors it observes.

While prior research confirms the usefulness of multi-layer diagnosis for more traditional systems, there is hardly any NoC related work available so far. A NoC diagnosis approach involving multiple layers is presented in [26]. Based on the work of [6], faulty links and crossbar connections are located in software. On network layer, functional faults such as misrouting or flit/packet dropping are diagnosed by means of hardware counters at each switch port and router logic. The information about the occurrence of functional faults in a switch is forwarded to a central network node where it is used by a probabilistic method to determine permanent faults. However, both diagnosis techniques are strictly separated from each other and there is no cross-layer interaction.

Therefore, we investigate subsequently how diagnosis techniques of multiple NoC layers can interact in order to achieve a superior tradeoff between diagnosis quality and performance impact compared to isolated solutions. To the best of our knowledge, this is the first attempt towards a true multi-layer diagnosis architecture with cross-layer interaction in the NoC domain.

III. Multi-Layer Diagnosis Overview

Diagnosis techniques can be employed on various network layers to locate permanent faults in NoCs. Each layer has its own functionality, with which diagnosis must be integrated, and its specific information that is subject of diagnostic processes:

- On transport layer, a software-based *diagnostic protocol* (DP) can locate faulty network components by analyzing incoming data packets.
- On network and data link layer, *functional diagnosis* (FD) identifies functional misbehavior and remaining functionality of a network switch using dedicated test packets.
- On the physical layer[1], *structural diagnosis* (SD) determines faulty gates or wires in the logic of a network link or switch.

[1]Whereas the Open Systems Interconnection Model (ISO/IEC 7498-1) definition of network layer only includes the physical interconnection between two switches, we interpret the term physical layer in a broader way to include the implementation of switches as digital circuits.

A. Combination of Diagnostic Techniques

In the following, we show how interaction between layer-specific diagnostic techniques is achieved by exchanging diagnostic information between the layers and by using this information to optimize the respective diagnostic processes. All possible combinations of the three layer-specific techniques will be considered. They are denoted as follows:

1) DP+FD,
2) DP+SD,
3) DP+FD+SD,
4) and FD+SD.

Figure 1 depicts the proposed information flow for multi-layer diagnosis. It consists of a top-down information flow in order to narrow down the position of a fault and of a bottom-up information flow to provide diagnostic feedback from FD or SD to DP. As we will discuss in more detail in Section IV, this is required to adapt DP to avoid cases where intact switches are erroneously diagnosed as faulty (*false positives*). In the limits of their respective fault models, FD and SD do not suffer from false positives due to the applied test patterns. Top-down and bottom-up information exchange are explained in the next subsection.

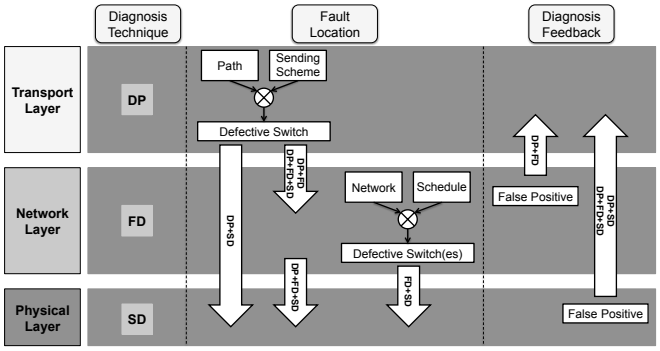


Fig. 1. Interaction of multi-layer techniques.

B. Cross-Layer Information Flow

When a fault has only been detected, without further diagnostic information, the set of potentially defective switches, Φ, is the set of all switches of the NoC, S. A diagnosis technique $t \in \{DP, FD, SD\}$ determines the subset of faulty switches (including the connected links) out of this set. This is described with the diagnosis function $d_t : S \rightarrow S$. Details on the implementation of this function are described in subsequent chapters for each technique t. Here we show the multi-layer flow of diagnosis for DP, FD, and SD. This flow is modeled by the algorithm presented in Listing 1. This algorithm does not have a centralized implementation, but it describes the behavior that results from the distributed interaction of the layer-specific techniques. The distributed interaction points will be highlighted in the layer-specific chapters.

The top-down diagnostic process starts with the transport layer protocol, DP, followed by functional diagnosis (FD) and

Listing 1 Flow of Diagnosis.

Φ: set of potentially faulty switches
Φ_t: set of faulty switches identified by technique t
Δ: set of all *false positives*
Σ: set of all faulty switches reported by DP

```
1:  while true do
2:    wait for fault detection
3:
4:    —Top-Down—
5:    if DP implemented then
6:      Φ_DP := d_DP(Φ ∩ Φ_path)   ▷ where 0 ≤ |Φ_DP| ≤ 1
7:      Σ := Σ ∪ Φ_DP                    ▷ add switch to list
8:      Φ := Φ_DP              ▷ update set of faulty switches
9:    end if
10:
11:   if FD implemented then
12:     Φ_FD := d_FD(Φ)                  ▷ where Φ_FD ⊆ Φ
13:     Δ := Φ \Φ_FD             ▷ determine false positives
14:     Φ := Φ_FD
15:   end if
16:
17:   if SD implemented then
18:     Φ_SD := d_SD(Φ)                  ▷ where Φ_SD ⊆ Φ
19:     Δ := Δ ∪ (Φ \Φ_SD)
20:     Φ := Φ_SD
21:   end if
22:
23:   —Bottom-Up—
24:   if DP implemented then
25:     if Δ ≠ ∅ then          ▷ false positive was reported
26:       adaptation of DP
27:       Σ := Σ \Δ          ▷ remove false positive from list
28:     end if
29:   end if
30: end while
```

structural diagnosis (SD). If any of these techniques is not implemented in the system, it is skipped. Each diagnostic technique receives the set of potentially faulty switches, Φ, as input and passes a potentially reduced set on to the lower layers. Assuming a deterministic routing, the set of switches that have to be diagnosed by DP can be reduced due to the knowledge about the path a packet has taken from sender to receiver. Thus, the diagnostic function is invoked only for the switches situated on that path, Φ_{path} (line 6). Diagnosis of FD and SD is carried out for set Φ (line 12 and 18) received from the preceding technique.

For each technique t, the diagnosis result Φ_t contains those switches identified as faulty. While the result of DP is, at each point in time, at most one faulty switch, the result of FD and SD can be any subset of Φ. A switch that was identified as faulty by DP is added to the list Σ of faulty switches and, as a result of this, is deactivated (line 7). Those potentially faulty switches that are not confirmed as such by FD or SD are added to the set of false positives, Δ (line 13 and 19). Each diagnostic technique t updates the set Φ according to its own, more precise diagnostic result Φ_t (line 8, 14, and 20). Thereby it also reduces the search space of faults for the subsequent technique(s).

The bottom-up information flow communicates any false positives found by FD or SD back to DP. If the set Δ is not empty, we know that DP erroneously diagnosed an intact switch as faulty and deactivated it. This is reversed by removing the switch from list Σ and thus reactivating it (line 27). Furthermore, the software protocol DP is adapted to reduce future false positives (line 26). This adaptation requires a deeper understanding of DP and is detailed in the next chapter.

IV. Software-based Diagnosis Protocol

The transport layer protocol, DP, is an enhanced version of our software-based end-to-end protocol presented in [27]. It has an increased localization granularity: besides faulty links, faulty crossbar connections of a switch can be diagnosed as well. The failure of a complete switch is diagnosed as the failure of all its links or crossbar connections.

Protocol DP consists of two parts: the base protocol and the diagnosis protocol. The base protocol (cf. Subsection IV-A) is responsible for acknowledging the receipt of packets to the corresponding senders and for retransmitting packets in case they were received with errors or if they were lost. We employ end-to-end retransmission as switch-to-switch retransmission protocols have been found inferior unless unrealistically high error rates are assumed [28][29]. The diagnosis protocol (cf. Subsection IV-B) is used to locate the faulty network component in case of a permanent fault. It is activated for such packets that could not be successfully delivered to their destinations after a number of attempts.

Beside the localization feature, the end-to-end protocol includes a software-based routing mechanism for fault tolerance. If, due to a localized fault, a switch cannot forward a packet using the regular routing, the packet is consumed by the processing element (PE) connected to that switch. The PE calculates an intermediate destination that results in a path that bypasses the fault. It updates the packet header with this intermediate information and reinjects the packet into the network. This software routing feature is not subject of this article; details can be found in [27]. Base protocol and diagnosis protocol are described in the following subsections.

A. Base Protocol

The base protocol uses the *Selective Repeat ARQ* mechanism, i.e. all packets received correctly at their destination are kept and only erroneous packets are retransmitted. To identify errors, each packet is equipped with a parity bit. Alternatively, any other error detection code (EDC) could be used. We further assume that on sender side, a retransmission buffer exists where a packet is stored until it is positively acknowledged. To reorder packets on receiver side, additional reorder buffers are required. Retransmission buffers and reorder buffers are assumed to be shared with the PE so that they can be managed in software.

The base protocol handles packet corruption as well as packet loss. When a packet arrives at its destination it is checked for errors. If no errors are found, a positive acknowledgement (*ACK*) is transmitted back to the sender. Otherwise, a negative acknowledgement (*NACK*) is given. *ACK*s and *NACK*s are transmitted as single-flit protocol packets. When a sender receives an *ACK*, the corresponding packet is removed from the sender's retransmission buffer, and in case of a *NACK*, the packet is retransmitted. In case a packet is lost, neither *ACK* nor *NACK* are generated and thus the packet will neither be removed nor retransmitted. To handle this situation, the base protocol makes use of an adaptable timer δ. If no acknowledgement is received for a packet within time t_δ, the packet is considered as lost and it is automatically retransmitted. This is also the case if a protocol packet with *ACK*s and *NACK*s was lost. The function $t_\delta(l)$ used by DP to adapt timeout t_δ is shown in Equation 1:

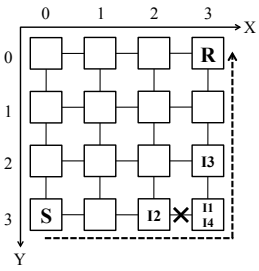
$$t_\delta(l) = t_{init} * (\lceil l^l_{max} * \frac{1}{l!} \rceil) \quad (1)$$

Function $t_\delta(l)$ allows the gradual increase of timeout t_δ up to a defined maximum. Starting from the initial timeout t_{init}, the adaptation function calculates the new timeout $t_\delta(l)$ depending on control parameter l ($0 \leq l < l_{max}$). For small values of l, $t_\delta(l)$ increases exponentially to achieve a large change in timer value. Towards large values of l, the timeout increase becomes linear to avoid excessively large timer values. The control parameter l is incremented by one and a new timeout is calculated whenever FD or SD report a false positive or the roundtrip time exceeds t_δ. The roundtrip time is the time between sending out a data packet and receiving the corresponding acknowledgement. It is considered in order to take the network load and potential congestion into account. As soon as roundtrip time falls significantly, the control parameter is decremented, leading to a shorter timeout.

The base protocol handles the corruption and loss of packets caused by transient errors. However, if packet retransmission fails multiple times (in our investigations: twice) we assume that a permanent fault is present. In this case, all retransmitted packets are corrupted again since they use the same path from sender to receiver; thus, they always either arrive at the receiver with an error or get lost. To restore a fault-free communication, the permanent fault has to be localized by the diagnosis protocol.

B. Diagnosis Protocol

On transport layer, fault localization relies on analysis of packet data. To be able to locate faulty network components, protocol DP must determine the path a packet has taken and thus requires knowledge about the routing. Moreover, fault localization relies on packets being retransmitted via the same path as the original packet. Therefore, the routing has to behave deterministically at least when the diagnosis protocol is activated. The routing may be reconfigured after diagnosis. The protocol finds the faulty network component by continuously narrowing down the fault position on the path. It can locate a single faulty network component at a time.



#	Sender X/Y	Receiver X/Y	d [hops]	ACK/NACK
1	0/3 (S)	3/0 (R)	6	NACK
2	0/3 (S)	3/3 (I1)	3	NACK
3	0/3 (S)	2/3 (I2)	2	ACK
4	2/3 (I2)	3/0 (R)	4	NACK
5	2/3 (I2)	2/3 (I3)	2	NACK
6	2/3 (I2)	3/3 (I4)	1	NACK
7	2/3 (I2)	3/3 (I4)	1	NACK
8	2/3 (I2)	3/3 (I4)	1	NACK

Fig. 2. Example of sending scheme.

When multiple faulty components coincide, these components are found by distinct diagnosis runs.

The diagnosis protocol is activated when a permanent fault has been detected with a packet. That packet is no longer sent to the receiver directly but to a so called intermediate node determined by the sender. Intermediate nodes are network nodes that are situated halfway on the path between sender and receiver. When a packet arrives at an intermediate node, it is consumed by that node and checked for faults. If the packet passes the check, i.e. no fault is found, this implies that the path from the original sender to the intermediate node is fault-free and the fault must reside in the second half of the path. In this case, the intermediate node takes over the role as sender for this packet, stores it in its retransmission buffer, and positively acknowledges the packet to the sender. On receipt of the acknowledgement, the sender removes the corresponding packet from its buffer. On the other hand, if the packet does not pass the check at the intermediate node, the fault is located between sender and intermediate. In this case, the intermediate node discards the packet and sends a negative acknowledgement to the sender. The sender now selects another intermediate node that is situated halfway to the old intermediate node.

An example of locating permanent faults on links and crossbar connections is shown in Figure 2: A packet is sent by sender S to receiver R using XY routing. Because of an undiscovered permanent fault on the link between network nodes $2/3$ and $3/3$ the packet becomes corrupted and thus, upon receipt, R sends back *NACK*. After a number of unsuccessful retransmissions, the diagnosis protocol is activated. The sender determines the first intermediate node $I1$ situated in the middle of the path between S and R. As the packet passes the faulty link, it is corrupted again. At $I1$, the packet is consumed, checked, and a *NACK* is sent back to S. Due to the fault, the NACK gets corrupted and is not considered at S. Thus, time t_δ has to elapse before the packet is retransmitted. S determines $I2$ to be the next intermediate node. This time, the packet arrives at $I2$ without errors, is stored in $I2$'s retransmission buffer, and an ACK is sent back to S. Source S removes the packet from its retransmission buffer, and $I2$ becomes the new sender of the packet. Instead of further bisecting the path between $I2$ and R, the packet is first sent to R. This is necessary for our protocol to be able to locate switch-internal faults. The reason for this is explained subsequently to this example. When $I2$ receives a *NACK* from R, it bisects the path and chooses first $I3$ and then $I4$ as next intermediate

nodes. The distance between $I2$ and $I4$ is only one hop. If a *NACK* is received in this situation, it implies that the fault has to be between $I2$ and $I4$.

We now explain the need for an intermediate node to send the packet to receiver R first before further bisecting the path. Assume that, instead of the link, the west-to-east crossbar connection of node $I2$ is faulty. Thus a packet becomes corrupted on its way from S to receiver R. The diagnosis protocol is activated for this packet, and as a result of path bisection, node $I2$ becomes an intermediate node eventually. However, this causes the fault to be bridged because node $I2$ uses crossbar connection west-to-local to consume the packet and crossbar connection local-to-east to re-inject the packet. Thus, the fault has no impact on the packet. If in the following the path is continuously bisected, the packet will be forwarded to further intermediate nodes and will arrive at R without faults eventually. The fault, however, is not localized.

To detect a fault on a crossbar connection, an intermediate node always sends a packet to its designated destination R first, before possibly further bisecting the path. If the intermediate node now receives an *ACK* from the receiver it can conclude that its switch must have a crossbar fault. This conclusion is based on the following statements:

1) The intermediate node has accepted the packet earlier, thus, the path between the sending node and intermediate node has to be fault-free.
2) As the intermediate node receives an *ACK* from the receiver, the path between intermediate node and receiver has to be fault-free.
3) Since the diagnosis protocol was activated, at least one fault must exist on the path from sender to receiver.

When combined with the FD or SD techniques, protocol DP maps a link or crossbar fault to the corresponding switch. The reason for this is that FD and SD always have to be carried out for an entire switch. In case of a link fault, Φ_{DP} is the neighbor of the switch that has diagnosed the fault. Thus, the faulty link is identified with its destination node: in the above example $\Phi_{DP} := \{I4\}$. If a crossbar fault has been diagnosed, Φ_{DP} is the switch that has diagnosed and contains the fault ($\Phi_{DP} := \{I2\}$).

V. Functional Diagnosis

Functional diagnosis (FD) is performed on the network layer in order to identify more precisely the switch functionality affected by a fault. To this end, FD considers six functional failure classes as defined in [30]:

1) *Misrouting*: A packet is routed to a wrong output port.
2) *Data corruption*: Data within one or more flits is altered.
3) *Packet loss*: At least one packet is lost on its way from the input port to an output port of a switch.
4) *Garbage packet*: A new packet is generated and forwarded to an output port. This includes packet duplication.
5) *Flit loss*: At least one flit of a packet is lost on its way from the input port to an output port of a switch.
6) *Garbage flit*: A new flit is generated and forwarded to an output port.

Some structural faults result in none of the above functional failures. These faults constitute the *residue class*. For all faults of this class, FD cannot identify the affected switch function and the complete switch is considered to be faulty.

FD is carried out online with the exception of the switch under diagnosis (SUD), which is set offline. All other switches in the network remain operative and continue transmitting data packets. The NoC itself is used as test access mechanism (TAM), thus avoiding the cost of a dedicated TAM. The online diagnosis process of FD is described in detail in Subsection V-A.

At design time, structural faults are classified according to the functional failure(s) they cause. Appropriate functional test patterns are generated to diagnose these functional failures with a high structural fault coverage. Both, fault classification and pattern generation, are described in detail in Subsection V-B. Additional input constraints are presented in Subsection V-C. They ensure that test patterns conform to the format of a valid packet. We refer to such patterns as test packets in the following.

A. Functional Online Diagnosis

In combination with the diagnosis protocol DP, FD is performed only for an individual switch reported potentially faulty by DP. For the diagnosis of the SUD, all adjacent switches are involved, sending test packets to the SUD's input ports. Additionally, the SUD receives test packets via its local port from the local processing element. While sending test packets, the neighbor switches remain operative and can transmit data packets via their other ports that are not connected to the SUD.

If FD is implemented *without DP*, all switches have to be considered as potentially faulty. Therefore, FD has to be performed for the complete network. Obviously, all switches of a network can be diagnosed consecutively, but this would result in significant latency. To reduce latency, as many switches as possible should be diagnosed in parallel. As neighbors of an SUD supply the test packets, they cannot be diagnosed at the same time. Hence, a scheduling is required that specifies the diagnosis start for each switch. We constrain the scheduling so that each node may only send packets to one SUD at a time. The resulting diagnosis schedule for an 8x8 mesh is depicted in Figure 3. The number shown for each switch corresponds to the iteration in which it is diagnosed. In total the schedule consists of five iterations. Note that the number of iterations does not increase for larger mesh networks.

As result of FD, out of the set of all diagnosed switches (set Φ) the faulty switches and their affected functions are determined. If a diagnosed switch by DP is not faulty, the switch is part of the set of false positives (set Δ). Any false positives are reported back to DP in order to reduce DP's pessimism. If FD is used without DP, multiple faulty switches can be determined by FD at the same time; however, it is unlikely that multiple permanent faults emerge simultaneously during operation.

B. Fault Classification and Pattern Generation Method

1) Overview: We make use of the satisfiability solver (SAT) based offline method presented in [30] to generate patterns

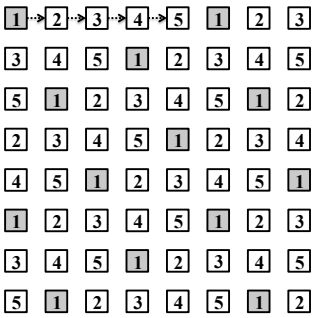


Fig. 3. Diagnosis scheduling for functional diagnosis.

for FD. An overview of the structural fault classification and pattern generation method is shown in Figure 4. For this work, we have enhanced the pattern generation so that the NoC can be used as TAM concurrently with regular network operation. Our method makes use of SAT instances of the fault-free switch and of the faulty switch. The faulty switch is a copy of the fault-free one including the literals for the injected structural fault. To model a functional failure of a switch, we add appropriate clauses that identify a functional mismatch between the fault-free and faulty switch. Using this extended SAT instance, the fault classification algorithm iteratively searches for primary input assignments that make injected structural faults observable through functional failures.

2) Modeling of Functional Failures: A structural fault f is testable if an input sequence exists for which, under consideration of the primary input constraints of the switch, a defined functional mismatch is observable. A functional failure is modeled as characteristic boolean functions that capture input constraints, f_{in}, and the output mismatch, f_{out}, by which the failure class can be identified, over a number of cycles T. The corresponding SAT instance Φ_{FF}^{T} is a conjunction of the CNF of f_{in} and f_{out}:

$$\Phi_{FF}^{T} = CNF(f_{in}) \wedge CNF(f_{out}). \tag{2}$$

Function f_{in} specifies the input constraints for the duration of T for the primary inputs of the switch. Hence, f_{in} for input port $p \in$ *Ports* is defined as:

$$f_{in}(p,t) = f(Din_{p,t}, HSin_{p,t}) \tag{3}$$

where $Din_{p,t}$ and $HSin_{p,t}$ are the input data lines and handshake signals of port p at cycle $1 \leq t \leq T$.

The output characteristic function f_{out} defines the functional mismatch of the switch for the primary outputs. This function takes the functional circuit responses of the fault-free switch and those of the switch with fault f as parameters:

$$f_{out}(p,t) = f(Dout_{p,t}, Dout_{p,t}^{f}, HSout_{p,t}, HSout_{p,t}^{f}) \tag{4}$$

where $Dout_{p,t}$ and $HSout_{p,t}$ are the data lines and handshake signals of the fault-free copy of the switch, and $Dout_{p,t}^{f}$ and $HSout_{p,t}^{f}$ belong to the faulty copy. For modeling the six functional fault classes introduced at the beginning of this chapter, we consider a send signal being used as handshake signal *HS* which we denote $send_{p,t}$. The output characteristic functions for the functional fault classes are listed in Table I.

3) Fault Classification: The relation between the functional failure classes and structural faults is established by a fault classification step. The classification algorithm receives the target functional failure class F and the list of structural faults as inputs. It then creates the SAT instances of the sequential fault-free switch Φ_{s}^{T} and of the targeted functional failure class Φ_{FF}^{T}. The algorithm iteratively picks a structural fault f from the fault list and creates the corresponding instance of the faulty switch $\Phi_{SF}^{f,T}$. By conjunction of these three parts, the fault classification SAT instance Φ_R is constructed:

$$\Phi_R = \Phi_{FF}^{T} \wedge \Phi_{s}^{T} \wedge \Phi_{SF}^{f,T} \tag{5}$$

The algorithm searches for an input assignment that satisfies Φ_R, i.e. the targeted functional failure F becomes observable. If an assignment is found, fault f causes the functional failure and thus f is related to the corresponding functional failure class. A structural fault may belong to multiple functional failure classes. If a fault cannot be related to any of the classes, the fault is part of the residue class. Upon termination of the algorithm, the subset of structural faults that cause F is known. The classification results for all the defined functional failure classes constitute what we call the golden classification reference.

If an input assignment leads to an observable functional failure for a structural fault f, it can be used as test pattern. However, this would result in a large pattern set that contains one pattern per structural fault f for each functional failure F.

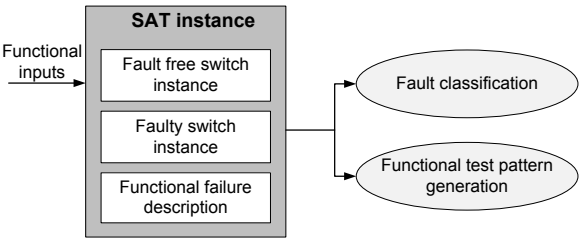


Fig. 4. Overview of pattern generation and fault classification method.

TABLE I
OUTPUT CHARACTERISTICS OF THE FUNCTIONAL FAILURE CLASSES

Functional Failure Class	f_{out}
Misrouting	$\bigvee_{\substack{p,q \in Ports,\\ p \neq q}} \bigwedge_{t=c_h}^{T} (\overline{send}_{p,t} \wedge send_{p,t}^{f}) \wedge (send_{q,t} \wedge \overline{send}_{q,t}^{f}) \wedge (Dout_{p,t} = Dout_{q,t}^{f})$ where $c_h = T - num\_flits + 1$
Data corruption	$\bigvee_{p \in Ports} \bigvee_{t=1}^{T} (Dout_{p,t} \neq Dout_{p,t}^{f}) \wedge (send_{p,t} \wedge send_{p,t}^{f})$
Flit loss	$\bigvee_{p \in Ports} \bigvee_{t=1}^{T} (send_{p,t} \wedge \overline{send}_{p,t}^{f})$
Packet loss	Flit loss holds for the packet length
Garbage flit	$\bigvee_{p \in Ports} \bigvee_{t=1}^{T} (\overline{send}_{p,t} \wedge send_{p,t}^{f})$
Garbage packet	Garbage flit holds for the packet length

As diagnosis performance relates to the number of test patterns it is favorable to remove redundant patterns. We employ fault dropping to minimize the pattern set. However, it must be ensured that the minimized pattern set does not change the structural fault coverage of the functional failure classes.

C. Input Constraints for Functional Online Diagnosis

In order to use the NoC's communication infrastructure as TAM, test patterns determined by SAT have to meet the format requirements on valid packets. In our case, a test packet must start with a head flit and end with a tail flit. Between head and tail, an arbitrary number of body flits may exist (fpp: number of flits per packet):

$$\bigwedge_{p \in Ports} \bigwedge_{t=1}^{T-fpp+1} (Din_{p,t} = head) \Leftrightarrow (Din_{p,t+fpp-1} = tail) \tag{6}$$

During diagnosis, test packets have to be applied to the inputs of an SUD in exactly the same way as determined by the SAT solver. If this requirement is violated, the functional failure may no longer be observable. This is a problem especially for online diagnosis because it implies that all neighbors of the SUD have to synchronize the injection of test patterns so that they arrive at the SUD's inputs at the correct point in time. To avoid this issue, we constrain all switch input ports $p \in Ports$ but one, i.e. the test port $p_{test} \in Ports$, to constant values and find a test packet for the test port that causes the functional failure. For online diagnosis, this has the effect that only one neighbor is sending an actual test packet at a time and thus there is no time dependency between the test packets of different neighbors. To find appropriate test patterns, an additional clause (equivalence) is added to the SAT instance that enforces constant flits and handshake control signals at all non-test ports:

$$\forall p \neq p_{test}, \forall t \in [2, T] : (b_{p,1} \vee \overline{b_{p,t}}) \wedge (\overline{b_{p,1}} \vee b_{p,t}) \tag{7}$$

The clause ensures that all the bits $b_{p,t}$ of all flits arriving at $t \geq 2$ have the same value as the bits $b_{p,1}$ in the first flit at $t = 1$. The clauses to ensure that a pattern corresponds to a valid packet and to enforce constant flits on non-test ports are specified in the input characteristic function of the SAT instance.

VI. Structural Diagnosis

On the physical layer, structural diagnosis (SD) is performed, which is the most granular diagnosis approach in our multi-layer scheme. In order to perform structural logic diagnosis, the gate level structure of the switch, test patterns targeting structural faults and their responses are required. Scan design [14], the most widely used structural test method also for NoCs [15][16], is employed and used for offline or online testing. The test patterns are applied to the scan chains via dedicated test ports, and a central test controller handles the test scheduling for the shift-in, shift-out, and capture/load phases. If a pure offline test is sufficient, the test control can be executed by an external tester. An embedded built-in self-test (BIST) controller is required for an online test strategy which switches off only a part of the network for testing and migrates tasks in a similar way as used in CASP (Concurrent Autonomous Chip Self-Test Using Stored Test Patterns, [31]).

A. Structural Online Diagnosis

The purpose of structural diagnosis is identifying the remaining functionalities of a switch which has already been recognized as faulty by the higher level approaches DP or FD. A central diagnosis service point analyzes the test responses, uses a dictionary created beforehand for defect location and then extracts the remaining functionalities of the switch in presence of this fault. The structural diagnosis approach helps to remove the false positives of the multi-layer approach as well. If the deterministic test patterns do not indicate a fault, the switch belongs to the set of false-positives. This may happen only if the software-based diagnosis approach (DP) has erroneously indicated the switch as faulty.

B. Remaining Switch Functionality

Usually, failing test patterns and their responses are used as input to a fault dictionary for fault location. The approach presented here is based on [32], skips this step, but creates a dictionary of affected functionalities beforehand. For a given fault f, two failures may occur:

1) A routing path from input port A to output port B is broken iff there exists a test pattern that makes the fault observable at B while the routing path $A \rightarrow B$ has been selected by the router.
2) An output port O is damaged iff there exists a test pattern that makes the fault observable at O for at least one routing path $x \rightarrow y$, $y \neq O$.

The size of such a dictionary is limited by the number of faults detected during constrained automatic test pattern generation (ATPG). To identify an affected functionality, ATPG is constrained to the input and output ports involved and determines those faults which corrupt a certain routing path or output port. Input constraints specify the routing signals for each routing path. In addition, all output ports except those on which the fault effect should be observed, are masked by constraints. When the ATPG is able to generate a test pattern, the respective routing path and the output port are affected by the fault and should be ruled out. The routing paths and output ports which are not ruled out by the ATPG can be used for normal operation [20]. A fault tolerant architecture may support deactivation of defective switch functions or ports so that the degraded switch with degraded functionality and performance is used for normal operation. When only deactivation of defective ports is supported, a vertex cover algorithm [20] finds the minimum number of ports to avoid the defective routings as well.

VII. Evaluation

In this section, we evaluate each diagnosis technique and their combinations with respect to diagnosis quality (Subsection VII-B) and impact on system performance (Subsection VII-C). The definition of both evaluation criteria is given

in the respective subsections. As they are in conflict with each other, we investigate their tradeoff and determine pareto-optimal solutions in Subsection VII-D.

For the quality evaluation of functional and structural diagnosis we have used the netlist synthesized from our RTL design of a typical switch with wormhole switching, five in/out ports, a crossbar, an XY router, and Stall/Go flow control. For performance evaluation a SystemC model of the complete NoC has been simulated, which includes a cycle-true switch model. The simulation model is parameterized with NoC mesh size, network load and traffic model. All simulations were carried out for an 8x8 mesh with moderate network load (flit injection rate 0.14 flits/node/cycle) [2] and high network load (flit injection rate 0.28, close to saturation) using uniform random traffic. Additional results are provided for varying network load, with transposed traffic, and for a larger 16x16 NoC. Moreover, we have investigated the NoC in a system context with application traffic.

The effect of faults is simulated by injecting permanent faults during operation. Results are averaged over 25 randomly generated fault patterns with 1 to 5 faults. Each fault models the defect of a switch by corrupting the packets that pass through the switch. Each fault pattern is simulated for 500,000 cycles. Within this period, the faults are injected one after another at random points in time. Once a permanent fault is diagnosed in the simulation, the faulty component is shut down. During the diagnosis process, whose duration is discussed subsequently (cf. Subsection VII-C2), further packets may suffer from the faulty component. These packets would be kept in their retransmission buffers until diagnosis is finished. To bypass a shut down component, software routing (cf. Section IV) is employed.

A. Cost breakdown

In this section we document the cost overhead of additional logic (gate count) and memory required for our diagnostic techniques:

- SD requires additional storage for 116,802 bit (approx. 14 kByte) of test patterns. These are applied by resources (test controller, scan chain) that have to be implemented for production test anyway, thus not causing additional logic overhead.
- FD requires 2,972,299 bit (approx. 362 kByte) of storage for the functional test program. Applying these patterns requires no logic overhead as patterns are injected by the processing elements attached to the NoC switches.
- DP requires a retransmission buffer of 10 packets (200 Byte) per network interface. The code size for the diagnostic protocol causes an overhead of 3.3 kByte of object code, 9.7% of the total protocol code size of 33.9 kByte. Both, packet and code storage can be implemented without additional memory instances using the local memory of the processing element.

[2]Note that the injection rate includes the injection of data flits and acknowledgement flits.

B. Diagnosis Quality

1) Evaluation methodology: Diagnosis quality is rated by whether a fault can be detected and how accurately it can be localized. It is composed of the following three characteristics:

a) the fault coverage, i.e. the ability to detect structural faults,
b) localization and classification accuracy, i.e. the ability to pinpoint the location of a fault and to classify its functional effect, and
c) the occurrence of false positives.

While these individual characteristics can be assessed quantitatively, overall quality has multiple dimensions and cannot be measured with a single, easily comparable number. We therefore order diagnosis techniques according to their quantitative performance with respect to the three criteria, and assign them quality levels in the range from 1 (least quality) to 5 (highest quality).

2) Layer-Specific Diagnosis Techniques: For functional diagnosis (FD) and structural diagnosis (SD), *structural fault coverage* close to 100% can be achieved with our pattern generation methods. This makes FD and SD superior compared to DP. For both, FD and SD, *accuracy* depends only on the test patterns used and is not affected by the network load; thus the accuracy of both techniques is the same for moderate and high load. In contrast to FD, which classifies the functionality affected by a structural fault, SD can narrow down a structural fault to a single signal or gate of the netlist switch (cf. Section VI), thus achieving 100% localization accuracy. Evaluation of FD [30] reveals that it succeeds in mapping 79.8% of the structural faults to one of the functional fault classes presented in Section V. For the remaining 20.2% of faults, however, it can only be concluded that the switch has a permanent fault but not which functionality is affected. Therefore, FD has a lower localization accuracy than SD. With valid test patterns, this technique does not produce *false positives*. This is another indication of higher diagnosis quality in comparison to DP.

The software-based diagnosis protocol, DP, can only diagnose faults that are functionally testable and that become manifest in loss or corruption of data packets. Thus, from the 79.8% of functionally testable faults, we have to subtract the 8.5% of faults that only lead to a misrouting, yielding 71.3% of faults than can be covered (detected). Exactly these

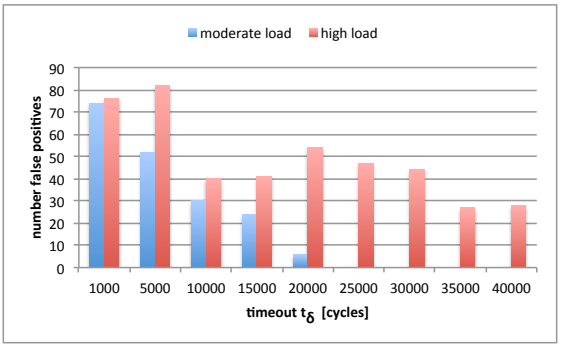


Fig. 5. Number of false positives of DP.

TABLE II
DIAGNOSIS QUALITY

Criterion	DP(high)	DP(mod)	DP+FD	DP+SD	DP+FD+SD	FD	FD+SD	SD
coverage [%]	71.3	71.3	71.3	71.3	71.3	100	100	100
accuracy [%]	71.3	71.3	79.8	100	100	79.8	100	100
false positives	27...40	0	0	0	0	0	0	0
quality level	1	2	2	3	3	4	5	5

faults can be classified according the DP fault model, i.e. fault classification accuracy also amounts to 71.3%. With respect to *false positives*, the value of parameter t_{init} used to calculate timeout t_δ (cf. Equation 1) is crucial for DP's diagnostic quality. If the value is chosen too small, the diagnosis protocol may be activated due to timeouts that are unrelated to faults. This in turn may lead to a shutdown of an intact link or crossbar connection. The number of diagnosed false positives for different values of t_{init} is shown in Figure 5. For the simulation it was assumed that timeout t_δ can be increased twice, i.e. $l_{max} = 3$. The results show that, for moderate network load, no false positives are diagnosed by DP when a proper value is chosen for t_δ. For high network load, however, DP produces false positives at all considered timeout values, and reasonable settings lead to false positives in the range from 27 to 40. Since the diagnosis quality of DP depends on the network load, we subsequently distinguish between moderate load (DP_{mod}) and high load (DP_{high}).

In summary, quality of the individual diagnosis techniques is ranked as follows: $DP_{high} < DP_{mod} < FD < SD$. The quantitative assessment of the three criteria is included in Table II.

3) Multi-Layer Diagnosis Techniques: We now assess the diagnosis quality of the multi-layer combinations, DP+FD, DP+SD, FD+SD, and DP+FD+SD. Due to the multi-layer interaction (cf. Section III), the technique used on highest layer identifies the faulty switch and reports it to the lower layer technique. Thus, *fault coverage* is determined by the technique on highest layer. On the other hand, *accuracy* is defined by the technique on lowest layer. Simulations have shown that combining DP with one of the lower layer techniques reduces the number of *false positives* to zero. Although false positives are still diagnosed by DP, in the experiments they are identified and corrected by the lower layer technique. The resulting quantiative assessment of the multi-layer techniques is included in Table II.

The table also shows the assignment of quality levels from 1 (low) to 5 (high) to all individual techniques and their multi-layer combinations. In this ranking, highest priority is given to the ability to detect faults (fault coverage). The other two criteria are used to differentiate cases of identical fault coverage. The diagnostic quality of DP under high load is inferior to all other techniques, and thus, we rank DP_{high} at the lowest quality level. Diagnostic quality benefits from eliminating false positives, which is achieved by operating DP at moderate load or by combining it with FD. Thus, DP_{mod} and and DP+FD are ranked at quality level 2, ignoring the insignificant difference with respect to classification accuracy. By further adding SD, accuracy is significantly improved. Therefore, DP+SD (where SD additionally eliminates false positives independent of network load) and DP+FD+SD are ranked at the next higher quality level, 3. By removing DP from the investigated combinations, faults undetectable by DP are no longer masked from the more detailed diagnosis techniques. Hence, taking also into account the relationship between FD and SD established above, they are ranked at level 4 and 5, respectively. Note that this does not disqualify DP+ techniques since they offer a potentially reduced performance impact, which will be investigated in the next subsection. Finally, due to its high coverage and accuracy, the quality of FD+SD is similar to SD. In summary, this results in the quality ranking shown in Table II. We can conclude at this point that a multi-layer combination of techniques always offers better quality than the individual technique used on highest layer.

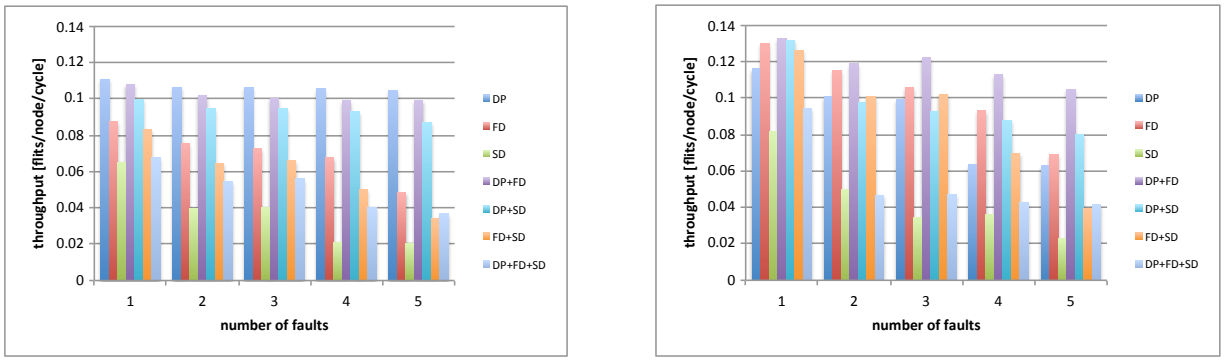
C. Performance Impact

1) Definitions: We define *fault localization latency* as time between the occurrence (in simulation: injection) of a permanent fault and its localization by a diagnosis technique. This includes all protocol and retransmission overhead as well as the time required for EDC checking. *Data throughput* is measured as the average number of flits received per node and cycle. It takes into account only data flits, including retransmissions, but no acknowledgement or protocol flits. Thus, it is a fair measure of NoC payload throughput, and does not count protocol overhead as useful data. The *injection rate* is the average number of flits injected into the network per node and cycle, including all acknowledgement / protocol flits.

2) Fault Localization Latency: To evaluate the impact of diagnosis techniques on system performance, we first investigate how long it takes each diagnosis technique to localize a permanent fault in the network. For this purpose, we have simulated the NoC under moderate and high network load. For the first 20,000 cycles the network is filled with flits before a single permanent fault is injected to the network. The measured fault localization latencies are shown in Table III.

TABLE III
FAULT LOCALIZATION LATENCY [CYCLES]

Diagnosis technique	Moderate load	High load
DP	360	680
FD	32,610	32,880
SD	99,864	100,134
DP+FD	6,859	7,179
DP+SD	50,237	50,557
FD+SD	82,487	82,757
DP+FD+SD	56,736	57,056



(a) Data throughput under moderate network load

(b) Data throughput under high network load

Fig. 6. Data throughput achievable with the different diagnosis techniques in an 8x8 mesh under uniform random traffic.

The results show that localization latency generally increases slightly for high network load. This is because a higher latency of acknowledgements of the base protocol delays the start of diagnosis. Among all diagnosis techniques, protocol DP offers the least localization latency in both load situations. Beside the reduced diagnosis effort, this can be attributed to the limitation of diagnosis to the switches on the path between sender and receiver. FD and SD in isolation, however, have to be performed for the entire NoC. In case of FD, latency is composed of the time required for the five diagnosis iterations (cf. FD scheduling in Subsection V-A). The total localization latency for SD is composed of the time required to identify and test the faulty switches with test patterns and the time to analyze the test responses. When FD or SD are combined with DP, DP reduces the set Φ of potentially faulty switches to a single one. As a result of this, the five-iteration schedule for FD and the identification phase of SD can be omitted and the localization latency is considerably reduced for both load situations. The latency of DP+FD is 6,859 cycles for moderate load and 7,179 cycles for high load. This corresponds to approximately 1/5 of the latency required for FD, i.e. the latency for a single FD execution. For DP+SD, fault localization latency is reduced by about 50% (in comparison to SD) to 50,237 and 50,557 cycles, respectively. This is because a single test run, instead of two, is sufficient when the defective switch has already been identified. The fault localization latencies of FD+SD and DP+FD+SD are composed of:

- FD+SD: full FD + 50% SD.
- DP+FD+SD: DP + one iteration FD + 50% SD.

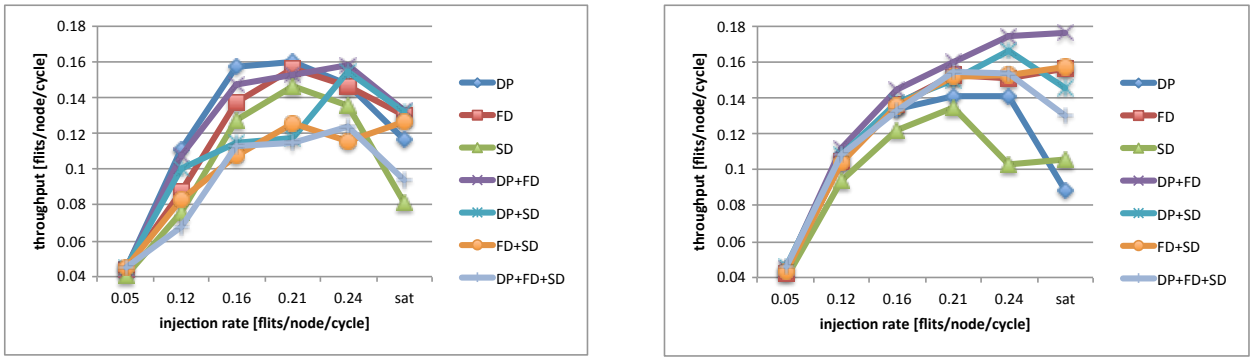
We can observe that, independent of network load, the combination of a lower layer technique with one on higher layer results in a reduction of localization latency compared to the lower layer technique used in isolation.

3) Data Throughput: During diagnosis, for the duration given by the fault localization latency, system performance is reduced because, depending on the diagnosis technique(s) used, a number of switches are not or not fully available for payload data transmission. The impact on system performance is measured in terms of the achievable data throughput. Results are shown in Figure 6 for all diagnosis techniques.

Under moderate load (cf. Figure 6a), throughput generally decreases with an increasing number of permanent faults. This decrease is related to the quality level of a technique: Due to larger fault localization latency, higher diagnosis quality correlates with larger performance reduction. Comparing DP, FD, and SD, the results show that DP offers the largest throughput for any number of faults since its localization latency is significantly smaller (cf. Subsection VII-C2). Furthermore, in contrast to FD and SD, the throughput decreases just slightly compared to the fault-free case when DP is used in isolation. This is because DP involves only a limited number of switches situated on a faulty path, while FD and SD are always performed for the entire network. The large performance impact of FD and SD can be mitigated by combining them with DP. DP+FD and DP+SD offer better throughput compared to FD and SD, respectively, because in their setting the costly lower layer diagnosis is only performed for a single switch that was identified as faulty by DP. The combination DP+FD+SD provides less throughput than DP+FD or DP+SD due to the sequential execution of FD and SD. However, for more than one fault, DP+FD+SD still has a higher throughput than SD because only individual switches are diagnosed each time a fault occurs. The combination FD+SD also leads to better throughput than SD alone because during a diagnosis iteration of FD all switches but the diagnosed ones remain operative and SD is only performed for individual switches identified as faulty by FD.

The throughput results for high network load are shown in Figure 6b. In this scenario, the DP no longer offers superior performance. The reason for this is the increased number of diagnosis protocol activations because more timeouts t_δ occur due to congestion. In our simulation setting, this results in approximately 30 shutdowns of components erroneously identified as faulty (cf. Subsection VII-B). Only when DP is combined with FD or SD, these false positives are detected so that the permanent shutdown of the corresponding components can be avoided. With increasing number of faults, the throughput of SD, FD+SD, and DP+FD+SD falls below DP's throughput due to SD's long localization latency. FD, DP+FD and DP+SD exhibit good throughput results and outperform DP independent of the number of faults.

Future CMOS technologies will enable the production of



(a) Uniform random traffic

(b) Transposed traffic

Fig. 7. Data throughput over injection rate in an 8x8 mesh with a single fault.

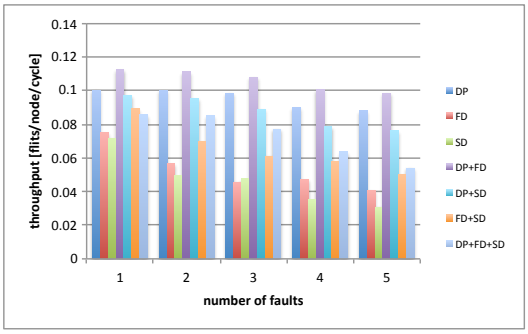


Fig. 8. Data throughput in a 16x16 mesh (uniform random traffic).

larger NoCs. Figure 8 shows throughput achievable under faults in a 16x16 mesh, where the injection rate is 0.14, close to saturation[3]. The relative performance of the diagnosis techniques shows a trend similar to Figure 8, confirming their applicability in larger NoCs. Throughput is slightly lower, compared to the 8x8 case, because more nodes generate more protocol packets that limit the bandwidth available for data packet throughput. On the other hand, thoughput degradation with increasing number of faults is less pronounced because the larger number of nodes and alternative routes reduces the relative impact of each fault.

The evolution of throughput with increasing injection rate is depicted in Figure 7. For random uniform traffic (Figure 7a), the protocol overhead (protocol packets) due to retransmissions rises strongly as the injection rate approaches saturation (sat), leading to a reduction of data packet throughput. With transposed traffic (Figure 7b), where each node communicates only with a single receiver, protocol overhead is reduced, leading to slightly higher achievable throughput, whereas general trends are similar to the other traffic model.

4) Application Execution Time: In addition to synthetic traffic, we have simulated the diagnosis techniques with application traffic using traffic patterns for an 8x8 mesh from the *MCSL Benchmark Suite* [33]. The execution time and the average throughput achieved with a fault-free network are shown in Table IV for each application. For simulations with faults, we have again used sets with five random permanent faults (cf. Subsection VII-C3) and measured the average overhead in execution time caused by diagnosis. The results are shown in Figure 9. We conclude from these results that the impact of diagnosis also depends on the traffic pattern. For example, the overhead is negligible (below 1%) for the H264 decoder and Newton-Euler Control applications. This is due to the high computation to communication ratio of these two applications. There are long periods with very low network load, thus, in these periods diagnosis has almost no impact on the communication. For the other applications, however, an overhead can be observed. This is the case especially for the SD technique. As in the case of uniform random traffic, again, the techniques DP, FD, and DP+FD generally exhibit the smallest overhead.

TABLE IV
EXECUTION TIME AND THROUGHPUT

Application	Execution Time [million cycles]	Throughput [flits/node/cycle]
Reed-Solomon Encoder	28.83	0.018
Reed-Solomon Decoder	29.99	0.01
H264 Decoder	29.98	0.005
Newton-Euler Control	30.48	0.001
Spec95 Fpppp	29.91	0.015
Sparse Matrix Solver	29.95	0.01

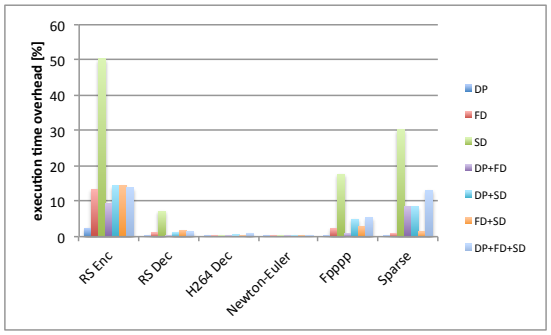
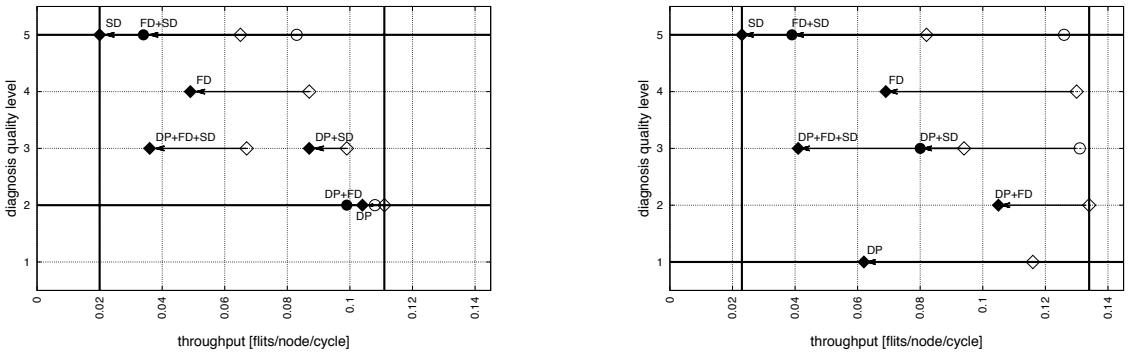


Fig. 9. Overhead application execution time.

[3]In meshes, saturation rate is inversely proportional to the square root of the number of nodes.



(a) Moderate network load

(b) High network load

Fig. 10. Diagnosis quality vs. throughput

D. Pareto Analysis

As detailed diagnosis takes time during which system performance is degraded, the design goals of high diagnosis quality and high performance are in conflict with each other. This can also be seen from the experimental results: For instance, DP offers the best performance under moderate network load, but its diagnosis quality is limited. To identify those diagnosis techniques which offer a preferable tradeoff between quality and performance, we now perform a pareto analysis. The design spaces spanned by quality and throughput for moderate and high network load are shown in Figure 10. Symbols represent the different combinations of diagnostic techniques, where the arrow from a contour symbol to a solid symbol represents the throughput trend from a single fault to five faults. Square and circle symbols are used for better differentiation between techniques with overlapping trendlines. A symbol represents a pareto-optimal point (at a given number of faults) if no other entry offers the same or better throughput and diagnosis quality. The unachievable ideal design goal (best performance and best diagnosis quality) is situated in the upper right corner of the design space.

In both load situations, FD, DP+SD, and FD+SD are pareto-optimal. For moderate load, additionally, protocol DP is pareto-optimal. In comparison with the other techniques, DP achieves a considerably higher throughput due to its reduced diagnosis effort. However, for high load DP is no longer pareto optimal due to the high number of false positives. As no feedback from FD or SD is available when DP is employed in isolation, links or crossbar connections erroneously diagnosed as faulty are shut down, thus reducing the communication resources of the NoC. The DP+FD combination, in which DP can use feedback from FD, is a pareto optimum for high load. If the load situation is unknown at design time, DP+FD should be given preference over DP as the results of DP+FD and DP are similar for moderate load.

Under both load scenarios, the throughput of all pareto optimal techniques is similar for one permanent fault. If the number of faults increases, however, the throughput gap widens; in particular, FD and FD+SD suffer from reduced throughput. For the techniques with protocol support, DP+SD and DP+FD, system performance is considerably higher as detailed diagnosis is only performed for individual switches.

From the results for both load situations we conclude that the combination of two techniques leads to a performance improvement, enabled by the reduction of the set of potentially faulty switches Φ by the higher-layer technique. This comes, however, at the expense of a reduced diagnosis quality. The mutual benefit for combined techniques can be seen especially for DP+FD and DP+SD. While DP reduces reduces set Φ, FD or SD identify false positives and prevent the shutdown of the corresponding components. This is also reflected in the results of these techniques. While the performance is higher than with FD and SD alone, the diagnosis quality is increased compared to DP.

In summary, the results show that the combination of protocol DP on transport layer with FD or SD on network layer and physical layer, respectively, increases the system performance noticeably compared to the techniques in isolation. The choice of one of the pareto optimal techniques, however, is application-dependent. For systems where performance is of concern, those techniques with protocol support can be applied. In safety-critical systems, the techniques with a higher diagnosis quality, i.e. FD and FD+SD, should be used. For systems that require the detection of latent faults, SD has to be used despite its low performance.

VIII. Conclusion

The combination of layer-specific diagnosis techniques into a multi-layer diagnosis approach is essential to find pareto-optimal solutions that offer a good tradeoff between system performance and diagnosis quality. The performance can be increased by combining a lower layer technique with a second one on higher layer. In this setting, the technique on higher layer helps achieve good performance by limiting the performance impact of diagnosis thanks to a reduced fault localization effort. On the other hand, the technique on lower layer ensures a high diagnostic quality, enabled by the more detailed information available on the lower layer. The combination of a software based diagnosis protocol on transport layer with either functional diagnosis on network layer or structural diagnosis on the physical layer turn out to be the most promising multi-layer diagnosis techniques.

REFERENCES

[1] Intel Xeon Phi Coprocessor, visited March 2015. [Online]. Available: https://software.intel.com/de-de/mic-developer

[2] International Technology Roadmap For Semiconductors, visited March 2015. [Online]. Available: http://www.itrs.net/

[3] W. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proc. of Design Automation Conf. (DAC)*, 2001, pp. 684–689.

[4] S. Borkar, "Designing reliable systems from unreliable components: the challenges of transistor variability and degradation," *IEEE Micro*, vol. 25, no. 6, pp. 10–16, 2005.

[5] J. Lienig, "Electromigration and its impact on physical design in future technologies," in *Proc. of ACM Int'l Symp. on Physical Design (ISPD)*, 2013, pp. 33–40.

[6] S. Shamshiri, A. Ghofrani, and K.-T. Cheng, "End-to-end error correction and online diagnosis for on-chip networks," in *Proc. of IEEE Int'l Test Conf. (ITC)*, 2011, pp. 1–10.

[7] A. Garbade, S. Weis, S. Schlingmann, B. Fechner, and T. Ungerer, "Fault localization in NoCs exploiting periodic heartbeat messages in a many-core environment," in *Proc. of 27th IEEE Int'l Parallel and Distributed Processing Symp. Workshops & PhD Forum (IPDPSW)*, 2013, pp. 791–795.

[8] Z. Zhang, D. Refauvelet, A. Greiner, M. Benabdenbi, and F. Pecheux, "Localization of damaged resources in NoC based shared-memory MP2SOC, using a distributed cooperative configuration infrastructure," in *Proc. of 29th IEEE VLSI Test Symp. (VTS)*, May 2011, pp. 229–234.

[9] D. Lee, R. Parikh, and V. Bertacco, "Brisk and limited-impact NoC routing reconfiguration," in *Proc. of the Conf. on Design, Automation & Test in Europe (DATE)*, 2014, pp. 306–306.

[10] R. Abdel-Khalek and V. Bertacco, "Post-silicon platform for the functional diagnosis and debug of networks-on-chip," *ACM Trans. on Embedded Computing Systems (TECS)*, vol. 13, no. 3s, pp. 1–25, Mar 2014.

[11] M. Kakoee, V. Bertacco, and L. Benini, "At-speed distributed functional testing to detect logic and delay faults in NoCs," *IEEE Trans. on Computers (TC)*, vol. 63, no. 3, pp. 703–717, March 2014.

[12] S. Rodrigo, J. Flich, A. Roca, S. Medardoni, D. Bertozzi, J. Camacho, F. Silla, and J. Duato, "Addressing manufacturing challenges with cost-efficient fault tolerant routing," in *Proc. of the 4th ACM/IEEE Int'l Symp. on Networks-on-Chip (NOCS)*, 2010, pp. 25–32.

[13] C. Liu, L. Zhang, Y. Han, and X. Li, "A resilient on-chip router design through data path salvaging," in *Proc. of 16th IEEE Asia and South PacificDesign Automation Conf. (ASP-DAC)*, 2011, pp. 437–442.

[14] M. Williams and J. Angell, "Enhancing testability of large-scale integrated circuits via test points and additional logic," *IEEE Trans. on Computers (TC)*, vol. C-22, no. 1, pp. 46–60, Jan 1973.

[15] M. Hosseinabady, A. Dalirsani, and Z. Navabi, "Using the inter- and intra-switch regularity in NoC switch testing," in *Proc. of Design, Automation & Test in Europe Conf. (DATE)*, 2007, pp. 1–6.

[16] M. Li, W.-B. Jone, and Q.-A. Zeng, "An efficient wrapper scan chain configuration method for network-on-chip testing," in *in Proc. IEEE Computer Society Annual Symp. on Emerging VLSI Technologies and Architectures (ISVLSI)*, 2006, pp. 147–152.

[17] A. Amory, E. Briao, E. Cota, M. Lubaszewski, and F. Moraes, "A scalable test strategy for network-on-chip routers," in *Proc. of IEEE Int'l Test Conf. (ITC)*, Nov 2005, pp. 590–599.

[18] C. Grecu, P. Pande, A. Ivanov, and R. Saleh, "Bist for network-on-chip interconnect infrastructures," in *Proc. of 24th IEEE VLSI Test Symp. (VTS)*, 2006, pp. 6 pp.–35.

[19] T. Lehtonen, D. Wolpert, P. Liljeberg, J. Plosila, and P. Ampadu, "Self-adaptive system for addressing permanent errors in on-chip interconnects," *IEEE Trans. on Very Large Scale Integration Systems (VLSI)*, vol. 18, no. 4, pp. 527–540, April 2010.

[20] A. Dalirsani, S. Holst, M. Elm, and H. Wunderlich, "Structural test for graceful degradation of NoC switches," in *Proc. of 16th IEEE European Test Symp. (ETS)*, 2011, pp. 183–188.

[21] C. Pinart, "A multilayer fault localization framework for IP over all-optical multilayer networks," *IEEE Network*, vol. 23, no. 3, pp. 4–9, May 2009.

[22] J. Emmert, C. Stroud, and M. Abramovici, "Online fault tolerance for fpga logic blocks," *IEEE Trans. on Very Large Scale Integration Systems (VLSI)*, vol. 15, no. 2, pp. 216–226, Feb 2007.

[23] N. P. Carter, H. Naeimi, and D. S. Gardner, "Design techniques for cross-layer resilience," in *Proc of the Conf. on Design, Automation and Test in Europe (DATE)*, 2010, pp. 1023–1028.

[24] M. Dadashi, L. Rashid, K. Pattabiraman, and S. Gopalakrishnan, "Hardware-software integrated diagnosis for intermittent hardware faults," in *Proc. of 44th Annual IEEE/IFIP Int'l Conf. on Dependable Systems and Networks (DSN)*, June 2014, pp. 363–374.

[25] M. Abramovici, C. Stroud, and J. Emmert, "Online BIST and BIST-based diagnosis of FPGA logic blocks," *IEEE Trans. on Very Large Scale Integration Systems (VLSI)*, vol. 12, no. 12, pp. 1284–1294, Dec 2004.

[26] A.-A. Ghofrani, R. Parikh, S. Shamshiri, A. DeOrio, K.-T. Cheng, and V. Bertacco, "Comprehensive online defect diagnosis in on-chip networks," in *Proc. of 30th IEEE VLSI Test Symp. (VTS)*, 2012, pp. 44–49.

[27] G. Schley, N. Batzolis, and M. Radetzki, "Fault localizing end-to-end flow control protocol for networks-on-chip," in *Proc. of 21st Euromicro Int'l Conf. on Parallel, Distributed and Network-Based Processing (PDP)*, 2013, pp. 454–461.

[28] D. Bertozzi, L. Benini, and G. De Micheli, "Error control schemes for on-chip communication links: The energy-reliability tradeoff," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 24, no. 6, pp. 818–831, 2005.

[29] A. Jantsch, R. Lauter, and A. Vitkowski, "Power analysis of link level and end-to-end data protection in networks on chip," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'05)*, 23-26 May 2005, pp. 1770–1773Vol.2.

[30] A. Dalirsani, N. Hatami, M. Imhof, M. Eggenberger, G. Schley, M. Radetzki, and H.-J. Wunderlich, "On covering structural defects in NoCs by functional tests," in *Proc. of 23rd IEEE Asian Test Symposium (ATS)*, Nov 2014, pp. 87–92.

[31] Y. Li, S. Makar, and S. Mitra, "Casp: Concurrent autonomous chip self-test using stored test patterns," in *Proc. of Design Automation and Test in Europe (DATE)*, March 2008, pp. 885–890.

[32] S. Holst and H.-J. Wunderlich, "Adaptive debug and diagnosis without fault dictionaries," *Journal of Electronic Testing*, vol. 25, no. 4-5, pp. 259–268, 2009.

[33] W. Liu, J. Xu, X. Wu, Y. Ye, X. Wang, W. Zhang, M. Nikdast, and Z. Wang, "A NoC traffic suite based on real applications," in *Proc. of IEEE Computer Society Annual Symp. on VLSI (ISVLSI)*, 2011, pp. 66–71.

Gert Schley received the Dipl-Ing (FH) degree in electrical engineering and information technologies in 2007 and the M.S. degree in embedded systems engineering in 2009 from the University of Applied Sciences, Pforzheim, Germany. Since 2009, he has been a Research Scientist with the Embedded Systems Group, University of Stuttgart. His research interests include hierarchical architectures and cross-layer fault tolerance for Network-on-Chip.

Atefe Dalirsani received her B.S. and M.S. degree in Computer Engineering from University of Tehran, Iran, in 2005 and 2007, respectively, and her Ph.D. from University of Stuttgart, Germany, in 2015. Her research interests include self-diagnosis and self-test of NoCs, design for testability, reliability and fault tolerance.

Marcus Eggenberger (S'13) received the Dipl.-Inform. degree from the Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, in 2012 and is currently pursuing a Ph.D. degree at the University of Stuttgart, Germany. His research interests include fast and accurate parallel simulation of network-on-chip.

Nadereh Hatami received her M.Sc. from the University of Tehran in 2008 and received her Ph.D. degree from the University of Stuttgart in 2014 with her thesis about Multi-level Analysis of Non-Functional Properties.

Hans-Joachim Wunderlich (F'09) is a full professor of Computer Science at the University of Stuttgart in Germany and Director of the Institute of Computer Architecture and Computer Engineering. His research interests include test, reliability and fault tolerance of microelectronic systems. He is a Fellow of IEEE.

Martin Radetzki (SM'12) is Professor of Embedded Systems Engineering with the University of Stuttgart. He received the Dipl.-Inform. and Dr.-Ing. degrees from the University of Oldenburg, Germany, in 1996 and 2000, respectively. His research interests include modelling and parallel simulation of embedded systems, design of robust systems, and architecture of fault-tolerant networks-on-chip.