# Structural Test and Diagnosis for Graceful Degradation of NoC Switches

Dalirsani, Atefe; Holst, Stefan; Elm, Melanie; Wunderlich, Hans-Joachim

Journal of Electronic Testing: Theory and Applications (JETTA) Vol. 28(6) October 2012

doi: http://dx.doi.org/10.1007/s10836-012-5329-9

**Abstract:** Networks-on-Chip (NoCs) are implicitly fault tolerant and due to their inherent redundancy they can overcome defective cores, links and switches. This effect can be used to increase yield at the cost of reduced performance. In this paper, a new diagnosis method based on the standard flow of industrial volume testing is presented, which is able to identify the intact functions of a defective network switch rather than providing only a pass/fail result for the complete switch. To achieve this, the new method combines for the first time the precision of structural testing with information on the functional behavior in the presence of defects. This allows to disable defective parts of a switch after production test and use the intact functions. Thereby, only a minimum performance decrease is induced while the yield is increased. According to the experimental results, the method improves the performability of NoCs since 56.86% and 72.42% of defects in two typical switch models only impair one switch port. Unlike previous methods for implementing fault tolerant switches, the developed technique does not impose any additional area overhead and is compatible with many common switch designs.

**Noname manuscript No.**
(will be inserted by the editor)

# Structural Test and Diagnosis for Graceful Degradation of NoC Switches

**Atefe Dalirsani · Stefan Holst · Melanie Elm · Hans-Joachim Wunderlich**

the date of receipt and acceptance should be inserted later

**Abstract** Networks-on-Chip (NoCs) are implicitly fault tolerant and due to their inherent redundancy they can overcome defective cores, links and switches. This effect can be used to increase yield at the cost of reduced performance.

In this paper, a new diagnosis method based on the standard flow of industrial volume testing is presented, which is able to identify the intact functions of a defective network switch rather than providing only a pass/fail result for the complete switch. To achieve this, the new method combines for the first time the precision of structural testing with information on the functional behavior in the presence of defects. This allows to disable defective parts of a switch after production test and use the intact functions. Thereby, only a minimum performance decrease is induced while the yield is increased.

According to the experimental results, the method improves the performability of NoCs since 56.86% and 72.42% of defects in two typical switch models only impair one switch port. Unlike previous methods for implementing fault tolerant switches, the developed technique does not impose any additional area overhead and is compatible with many common switch designs.

**Keywords** Network-on-Chip · Graceful Degradation · Logic Diagnosis · Performability

A. Dalirsani, S. Holst, M. Elm, H-J. Wunderlich
Institute of Computer Architecture
and Computer Engineering
University of Stuttgart
Pfaffenwaldring 47; D-70569 Stuttgart, Germany
E-mail: {dalirsani, holst, wu}@iti.uni-stuttgart.de,
melanie.elm@ieee.org

## 1 Introduction

Networks-on-Chip (NoCs) have emerged as a new message passing infrastructure to supersede the traditional bus structures and meet the communication requirements in large SoCs [1–3]. An NoC contains a large number of switches and interconnects that form a structure spanning across the chip. To maintain acceptable yield, such large scale structures must provide redundancy and tolerate spot defects.

For many components, yield may be traded off against performance. The most popular example is speed binning for high-end processor chips. Flash memories may be delivered, even if a few blocks are not functional and not accessible. Cache structures are inherently fault tolerant, and defects can be mastered by disabling the corresponding lines and reducing the cache capacity [4].

In a similar way, defective switches of an NoC can be discarded after testing as long as the available redundancies ensure connectivity, perhaps with degraded performance. The hardware structure of each switch in an NoC is tested after production [5–7], at power-up or on demand [8]. Switches that fail these structural tests can be disabled and isolated [9]. The NoC can compensate for this loss to a certain degree by fault-tolerant routing. Packets are routed over alternative paths to ensure connectivity between as many cores as possible [9–12].

By disabling defective switches, the overall performance of the system decreases because cores get isolated from the network and the diverged traffic can cause congestion. Figure 1 gives an example for such a scenario. Switch $(1, 1)$ is identified as defective by production test and subsequent diagnosis. If it is disabled, Core B will be isolated and in addition three network links become unusable.

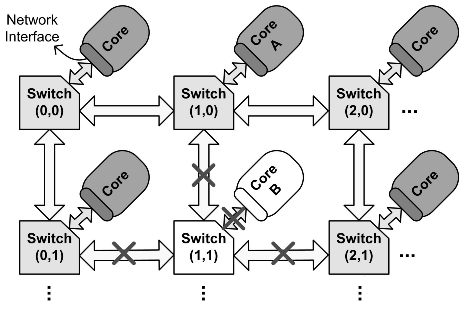


**Fig. 1** Performability loss by disabling switches.

The ability of preserving performance while tolerating a certain number of defects in a design is called *performability*. The performability can be increased, if not only complete switches but also defective links and ports can be disabled individually [13]. Let us assume it is possible to prove that the defect in switch $(1, 1)$ in figure 1 only affects the northern port. In this case, the northern port can be disabled while the others can still be used. Therefore, only one link in the network becomes unusable and core B is not isolated. One might argue, that this fine–grained reconfiguration may show diminishing returns in larger NoCs. However, as the network size increases, it takes more chip area at comparable defect densities. On the other hand, if technology scales down, even the defect density itself increases. The expected number of defective switches in the network increases accordingly and fine–grained reconfiguration will not loose its advantage. Moreover, detaching the healthy cores from the system due to defective switches will degrade the system performance inappropriately even in large systems. This is approved by the results from [14], as they indicate that reliability and yield decreases with the size of an NoC.

Recently, a few works have discussed similar types of such a graceful degradation in NoCs. In [13] a CRC encoding of the packets is presented to identify defective switch ports. However, the packet encoding is used specifically for fault detection in the datapath. In [8] a BIST architecture is presented that distributes the test patterns among the individual ports in order to identify the defective ones. In [15] a dual-channel architecture is introduced to deal with defective ports. Nevertheless, it does not discuss how to identify a defective port. Looking into the literature clearly reveals the gap of diagnosing defective NoC switches. None of the previous studies have used so far the standard test flow and fine-grained structural diagnosis to identify fault-free and faulty switch ports, which is the main contribution of the current work. The technique presented here for the first time is based on structural test and diagnosis methods for identifying and retaining the fault-free switch ports for system use and therefore improving performability and yield.

The approach here is based on the standard testing scheme and can deal with an arbitrary class of switches in order to handle graceful degradation. The method does not impose any additional hardware overhead for the diagnosis since we assume the standard flow and architecture of industrial volume test to determine defect locations inside the NoC switches. Besides, the same capabilities which are needed to isolate a defective switch from the network, are reused to support individual port deactivation. By mapping the structural defect information to NoC switch functions, we provide detailed information for fault tolerant routing.

The rest of this paper is organized as follows: The next section introduces the general concept of the approach and compares it with the state of the art. Section 3, 4, and 5 describe the identification of the remaining functionality of a switch. Section 6 applies this method to a typical switch and presents a thorough performability analysis with NoCs of various configurations.

## 2 Overview and Preliminaries

This chapter discusses the overall test, diagnosis and configuration flow for graceful degradation of NoC switches with maximum performability.

### 2.1 Overall Flow

The overall flow of the approach is depicted in figure 2. The main challenge is to identify the remaining capabilities of a defective switch from structural test data. It must be guaranteed that deactivating certain ports indeed confines all fault effects of the defect. The method to achieve this consists of three steps:

1. The structural test data is analyzed by an efficient logic diagnosis algorithm to identify the defect location within the random logic of the switch (Logic Diagnosis block in figure 2).
2. For each possible function of the switch, it is determined if it may be affected by the identified defect (Functional Mapping block of figure 2).
3. Finally, the ports associated with the affected functions are disabled (Function Lookup & Switch Reconfiguration block in figure 2).

Now, suppose the diagnosis outcome reports that the faulty responses generated by one specific switch in the NoC can be explained by a single fault $f$. Let us assume, the functional mapping described in chapter

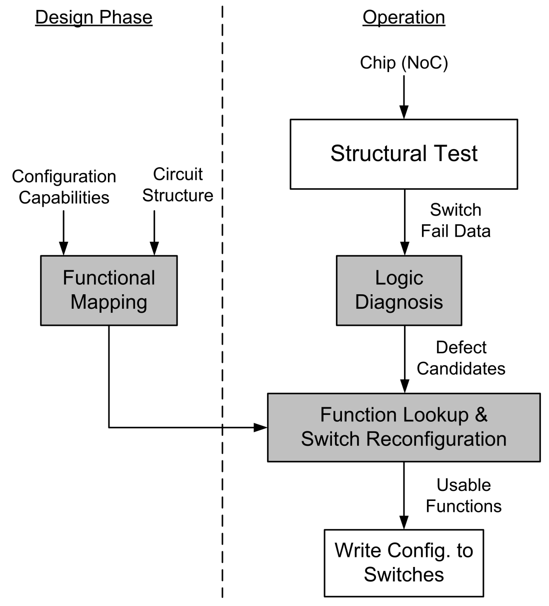


**Fig. 2** Overall flow

4 (step 2) has identified, which switch functions are affected by $f$. The affected functions and ports are looked up and the NoC is reconfigured to avoid the affected functions (step 3).

Figure 3 presents an example of using a semi-faulty switch in the NoC. In this 2D mesh, the outcome of our method indicates that defects only affect the southern output port of switch $(1, 0)$ and the northern input port of switch $(1, 1)$. Both are deactivated. If core $A$ sends a packet to $B$, the data will be rerouted via switches $(2, 0)$, $(2, 1)$ or $(0, 0)$, $(0, 1)$ alternatively using an appropriate fault tolerant routing like the one described in [13].

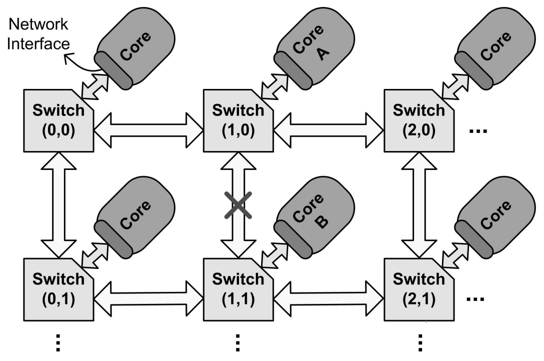


**Fig. 3** 2D grid topology

## 2.2 Switch Architecture Preliminaries

The method is not design-dependent and hence can be applied to any NoC architecture with switch designs having the following properties.

- A high fault coverage must be achievable in test mode. As a consequence, each individual switch must be accessible by ATE, test data must be transported to the switch and test responses have to be propagated to the outside. Many efficient NoC test strategies [6,16,7,17] already satisfy these requirements. This condition generally is fulfilled for a proper production test of an NoC.
- To efficiently utilize the test results for degradation, a switch must allow independent deactivation of any input or output port. This feature is already supported by many fault tolerant NoCs to isolate defective switches [9,11,8]. Even designs targeting only the deactivation of complete switches need to support the individual deactivation of ports to properly isolate a defective switch. All the neighbors of a defective switch are configured such that the output port connected to the defective switch is deactivated. Moreover, all the requests coming from the input port connected to the defective switch need to be ignored as well. A single bit for every input and output port indicates the faulty / non-faulty status of that port. A faulty input is deactivated by configuring the router to ignore any data and requests coming from the respective input port. A faulty output port is deactivated by redirecting packets destined for the respective port to an alternative output port. Since the same capabilities are needed to isolate a defective switch from the network, the more fine–grained port deactivation involves no additional overhead.
- The NoC must implement a fault tolerant routing scheme like [18,13,19,9] to guide diverged traffic over alternative routes to the correct destinations.

Case studies with actual implementations are presented in the experimental results.

## 2.3 Functional Switch Model

To formalize the functionality of a switch in general, we define the set of $P$ ports of the switch and we denote a switch function $x \triangleright y$ as the routing of data from input port $x$ to output port $y$ $(x, y \in P)$. As an example, consider a switch in a 2D mesh with four ports to neighboring switches and one port to the connected core. In this case $P = \{N, E, W, S, C\}$. Such a NoC switch can forward data from every input port to every other output port (if 180° turns are not allowed), this results in $5 \cdot 4 = 20$ distinct functions. This sort of function definition in the switch has already been used as functional fault model for NoCs [17,20–22].

Compared to the previous fault tolerant approaches for identifying switch portions to be deactivated [11,8], the method here does not introduce any hardware overhead and utilizes the available test structures already been added for production test. Also, it is not designed for online operation but to be applied after production test. Thus, it can make use of recent, powerful diagnosis techniques, which are able to track down defects to signals instead of defective units.

The following chapters provide a detailed description of the diagnosis flow and the mapping of structural defect information to NoC functions.

## 3 Structural Fault Diagnosis

According to section 2, the first step of the proposed method consist of the structural test and a logic diagnosis technique. If the structural test fails for a switch, well-established diagnosis techniques can be used to find a defect. The application at hand however has slightly higher demands on the diagnosis results than other applications as explained below.

Structural fault diagnosis [23–26] uses the available test fail data to locate the defects within the random logic of faulty switches. Often it is sufficient to locate a fault site, i.e. some signals or structures affected by a defect, to guide physical failure analysis or to provide data for volume diagnosis. This is not sufficient here, because additional fault sites may be present in other parts of the circuit which are not part of the diagnosis result, and we must have evidence that the remaining circuit structures are defect-free. To achieve this, the logic diagnosis algorithm has to classify the faulty behavior of the circuit to be either completely explainable by the identified defects or not. If the signature of the circuit cannot be completely explained, some unknown interactions among multiple fault sites are present and the switch is completely disabled. If the identified defects can explain the signature of the circuit completely, there is no indication that the remaining circuit structure is defective.

Studies have shown that most production defects disturb only a single internal signal directly [26] and diagnosis algorithms based on the single location at-a-time (SLAT) paradigm [25] can locate such defects with high precision. We will now show, how fault-model independent diagnosis approaches can be used to determine, whether a faulty behavior of a switch can be attributed to a single defective signal or gate within the circuit. Clearly, single stuck-at faults are insufficient to reason about arbitrary defects. Therefore, we will use the conditional line flip (CLF) model [27], which has been proven quite effective in this regard.

A defect can disturb a victim signal $v$ in arbitrary ways. In any case the victim signal $v$ will carry a faulty value at least in some situations. A CLF $v \oplus [cond]$ describes a victim signal $v$, which carries a faulty value (i. e. the opposite of the fault-free value or a "flipped" value), if the condition $[cond]$ is true. The victim signal $v$ is also called *fault site* and the condition $[cond]$ is of arbitrary (Boolean, temporal, or even random) nature. For example, $v \oplus [v]$ is a stuck-at 0 fault, $v \oplus [vw]$ is an AND-bridge fault from $w$ to $v$, and $v \oplus [v\overline{v_{-1}}]$ is a slow to rise transition fault with $v_{-1}$ evaluating to the previous value of $v$.

In the worst case, the victim signal $v$ will *always* generate a faulty value. This can be modeled as *unconditional line flip* $v \oplus [1]$. Fault simulation of this worst case condition may generate more failing responses than observed in the switch containing a defect $v \oplus [cond]$ with arbitrary condition. But every observed fail will be perfectly explained by a simulation of $v \oplus [1]$. In other words, locating a line flip $v \oplus [1]$ that is able to explain all *faulty* patterns, is sufficient to conclude that a defect $v \oplus [cond]$ explains the complete test. If a tool cannot directly work with unconditional line flip, similar conclusions can be made if all faulty patterns are explained by either a stuck-at 0 ($v \oplus [v]$) or a stuck-at 1 ($v \oplus [\overline{v}]$) at $v$.

By using a test set with sufficiently high resolution, the algorithm from [24] is able to identify all faults affecting a single site, and reports, if the test responses can or cannot be explained by a single conditional line flip. Even with a test set of high diagnostic resolution, the diagnosis algorithm might only be able to narrow down the set of simple fault candidates to more than one possibilities due to structural equivalency. This does not influence the overall method, because we observed that in all these cases, the equivalent candidates affect the same functions of the switch. Thus, the functions to disable are precisely identified. How to map defects to affected functions is explained in the next section.

## 4 Mapping Structural Faults to Switch Functionality

This section describes the mapping between structural faults and switch functions, which is done once during the NoC development. The result is stored into a dictionary and looked up after completion of the first step and prior to the third step of our diagnosis approach.

While the diagnostic procedure described above delivers both fault sites and fault conditions, the reconfiguration has to be based on worst-case conditions and assumes unconditional line flip faults. This way, the remaining functionality will not rely on the behavior of the fault, which may change over time. The mapping of every fault $f$ to the set of switch functions, which it may influence, is determined in two steps. The first is a topological preprocessing and the second is functional reasoning. A switch can be represented in a combinational model. Figure 4 provides an example for such a representation. The depicted switch has five ports $P = \{N, S, W, E, C\}$ and additional inputs and outputs from and to its own state elements. Beside that, the control input assignments for forcing the switch into any of its functions $x \triangleright y$ are known.

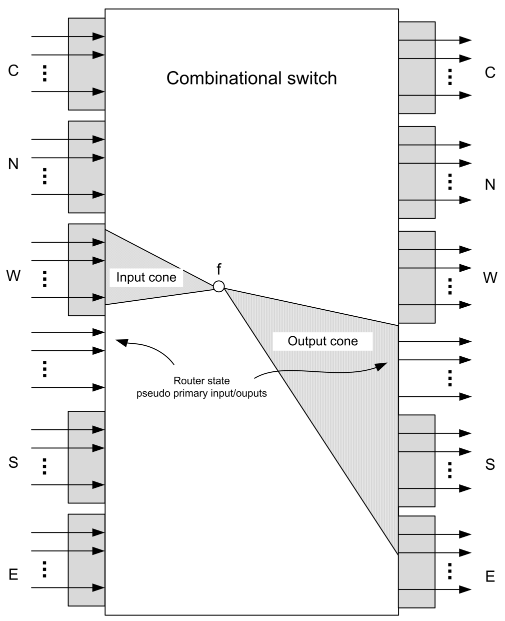


**Fig. 4** Combinational logic of the switch with a fault $f$

## 4.1 Topological Preprocessing

A fault $f$ can disturb the signals within the combinational circuit in arbitrary ways. Let $v$ be the signal associated with $f$. If there is no topological path from the victim signal $v$ neither to the output port $o \in P$ nor to the router state, the fault $f$ does not affect the functions $x \triangleright o$ $(x \in P)$. Only the remaining functions for every $o' \neq o$ have to be analyzed.

This simple observation already provides a good approximation of unaffected functions for all faults close to the output ports of the switch. However, most signals near the input ports have structural paths to many outputs and the router state. Hence, the topological preprocessing is pessimistic and a complementary analysis technique is needed to obtain a better approximation for unaffected functions.

## 4.2 Functional Reasoning

Functional reasoning determines exactly the switch functions that are affected by a fault, and the according ports that have to be disabled. This is achieved by means of constrained ATPG. The fault $f$ is injected and the control inputs are set (constrained) to enable a certain switch function under which the switch needs to be checked. Those output ports which are not of interest for the current check are masked out by ATPG constraints as well (in every check, only one switch output port remains unmasked). Then, ATPG is performed, and if ATPG is able to generate a test pattern activating and propagating the fault under the given constraints, this proves that the fault $f$ in combination with the chosen function may affect the unmasked output port. Depending on the function and the output port, it is either sufficient to disable a certain function or the output port has to be disabled. The distinction between these two cases is done by categorizing the checks into two classes (output port conditions and function conditions). In consequence, the complete analysis for every fault $f$ may have the following implications for reconfiguration:

1. Function conditions.
   For each function $x \triangleright y$ not ruled out by the topological preprocessing: We mask all the outputs $o \neq y$ and assign the control signals to select $x \triangleright y$. If ATPG is able to propagate $f$ to $y$ under this constraint, $x \triangleright y$ has to be avoided.
2. Output port conditions.
   For each output port $o$ and each control signal assignment selecting a function $x \triangleright y \neq o$, we mask all the ports $o' \neq o$ and try to propagate the fault $f$ to output $o$. If this is successful, the port $o$ may generate erroneous traffic even if it is not selected by the control logic. Thus, it has to be disabled.
3. Switch condition.
   The switch is disabled completely in case of:
   (a) Logic diagnosis cannot explain the faulty behavior by pointing to a single fault site.
   (b) ATPG is able to propagate the error signal to the router state.

ATPG should always be able to generate the test or to prove redundancy, since the switch is a rather small circuit. However, if ATPG would fail due to a timeout and abort a fault, the corresponding functions have to be removed as well.

## 5 Switch Reconfiguration

The reconfiguration of a switch affected by a fault $f$ is now determined by finding a minimal vertex cover in a graph constructed with the outcome of conditions 1 and 2 of functional reasoning. The graph contains a vertex for every input and every output port of the switch. In the example with $P = \{N, S, W, E, C\}$, there are in consequence 10 vertices. We add an edge from input port $x$ to output port $y$, if the function $x \triangleright y$ was disabled due to condition 1, i. e. function condition. Then, all output ports disabled due to condition 2 - i. e. output port condition - are marked as part of the vertex cover in the graph. Additional vertices are marked using a simple heuristic until all edges (avoided functions discovered by condition 1) are covered. The marked vertices are the ports to be disabled in order to confine the effects of fault $f$. This information is now stored in a dictionary for fast look-up after test application to the NoCs.

As an example, assume in a switch with $P = \{N, S, W, E, C\}$ function condition indicates that functions $N \triangleright C$, $N \triangleright S$, and $W \triangleright E$ are disabled. Also, condition 2 denotes that output port E is disabled. The graph is constructed like figure 5(a). Since vertex out_E is marked, the function $W \triangleright E$ has been covered already. To avoid functions $N \triangleright C$ and $N \triangleright S$ the efficient decision is to mark input port N. Marked vertices of figure 5(b) are the ports that have to be disabled in order to confine the fault effects.

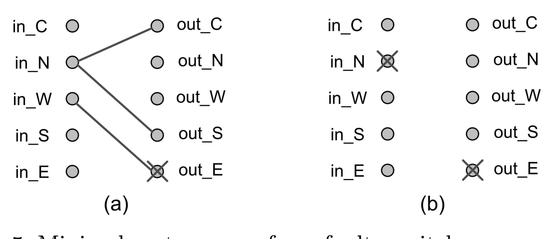


**Fig. 5** Minimal vertex cover for a faulty switch

## 6 Experimental Results

In this section, we present experimental results on a typical switch designed for NoC mesh architectures. Figure 6 shows the internal structure of the NoC switch from [7] used in our experiments. It consists of 5 output ports and 5 input ports. An output port contains a multiplexer to select data from any other input ports. The multiplexers of all output ports form the crossbar switch, which is controlled by the router. An input port contains a FIFO which buffers all received flits until they are processed by the router. The router implements a fault tolerant wormhole XY routing scheme [9] and processes each input port in a round robin fashion [7]. The routing algorithm uses standard XY routing in the fault–free areas and routes the packets over a ring around the faulty switches. By avoiding northwest and east-south turns, this fault-tolerant routing scheme is deadlock-free. The algorithm was originally developed for switch deactivation, but it can be applied for fine-grained port deactivation used here without any modification.

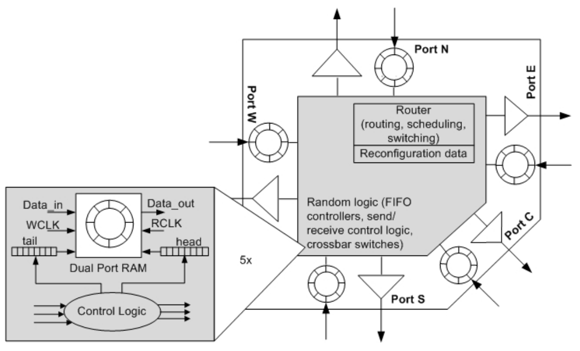


**Fig. 6** A typical NoC switch

The router, the crossbar and additional control logic to generate and process the handshake signals of the ports are random logic (gray area in figure 6) synthesized from VHDL and mapped to the lsi10k technology library. Because the memory elements are usually equipped with advanced BIST infrastructure, there is no need to consider the FIFO storage elements in the diagnosis process. Every port has its own storage space and it is already known which port must be deactivated due to a fault in the corresponding memory elements. Since recent commercial ATPG tools do not support the line flip fault model, we conducted our experiments on the conditional stuck-at fault model.

The experimental results consists of two subsections. In subsection 6.1, utilization of the proposed method for retaining remaining fault free functionalities of a defective switch is demonstrated by applying the method on two typical switches with 12 and 32 bits data width. Then, in subsection 6.2, the benefit of retaining the degraded switches for system use is evaluated by measuring the performability of NoCs under the influence of randomly injected defects.

## 6.1 Remaining Functionality

The experimental result is done for two NoC switches with the same characteristics as described earlier. The first switch is specialized for transmitting packets with 12 bit flits (flow control unit) and the second one supports 32 bit flit sizes as in the Intel Polaris [28]. Table 1 summarizes the synthesis results in terms of area which is occupied by the switches and the routers. Both of the switches have the same number of ports (5 input and 5 output ports), and FIFO sizes. The routers are identical. As shown in table 1, both routers have the same number of flip-flops corresponding to the router states. The small difference between the total cell area of the routers is related to different sizes of header flits.

**Table 1** Area report of 12-bit and 32-bit switch

| | 12-bit switch | 32-bit switch |
|---|---|---|
| number of I/O ports | 323 | 723 |
| number of combinational cells | 2176 | 2842 |
| number of flip-flips | 2683 | 3583 |
| total cell area | 4859 | 6425 |
| Router: | | |
| number of combinational cells | 1065 | 1129 |
| number of flip-flops | 448 | 448 |
| total cell area | 1513 | 1577 |

Both switches have been analyzed using the method presented, and the dictionary was created, which maps each fault site to specific input or output ports to be deactivated. The fault site is an internal signal or gate within the random logic pinpointed by the diagnosis algorithm. 12-bit and 32-bit switches contain 3301 and 4976 fault sites respectively. All the faults in the switches are diagnosable and the diagnosis algorithm is able to determine for each either a single fault site or a set of possible sites all belonging the same switch function. Table 2 shows the statistics of the generated dictionaries. We observe that more than 56% of the fault sites of the 12-bit switch is dedicated to a single input or output port. In other words, a defect within this portion of the switch can be tolerated by disabling one input or output port while the remaining nine input/output ports of the switch are still available to transfer packets. In the 32-bit switch, a much larger part (more than 72%) of the total fault sites is dedicated to a single input or output port. The larger flit size adds logic to the data path of the switch ports. This extra logic increases the probability of having faults in these dedicated parts of the circuit rather than in the shared parts as the router. This is confirmed by the proposed algorithm, which accurately determines the defective ports and demonstrates that a big portion of the faults in the switch influence only the flow of data in a single port and the other non-faulty ports and the router can be used for further operations.

**Table 2** Portion of fault sites in the circuitry dedicated to each port and the router

| | 12-bit switch Stuck-at faults | | 32-bit switch Stuck-at faults | |
|---|---|---|---|---|
| | count | % | count | % |
| input C | 228 | 6.91% | 295 | 5.93% |
| output C | 152 | 4.60% | 445 | 8.94% |
| input S | 221 | 6.69% | 268 | 5.39% |
| output S | 152 | 4.60% | 448 | 9.00% |
| input W | 221 | 6.69% | 268 | 5.39% |
| output W | 152 | 4.60% | 448 | 9.00% |
| input N | 224 | 6.79% | 271 | 5.45% |
| output N | 155 | 4.70% | 448 | 9.00% |
| input E | 221 | 6.69% | 268 | 5.39% |
| output E | 151 | 4.57% | 445 | 8.94% |
| Ports portion sum | 1877 | 56.86% | 3604 | 72.42% |
| Router | 1424 | 43.13% | 1372 | 27.57% |
| Sum | 3301 | 100% | 4976 | 100% |

## 6.2 Performability

This section details the amount of performance preserved by the presented scheme under a certain number of defects in the switches of the NoC. Each experiment was performed with 100 defect conditions, where a defect condition is a subset of all possible stuck-at faults, and the results were averaged. For each defect condition, a defined number of stuck-at faults are randomly injected into the random logic of the switches in each NoC. In this section, all the experiments have been done twice. Once we assume that the NoC is constructed by 12-bit switches, in which about 56% of the faults are dedicated to a single port of the switch. Then, the experiments are repeated with 32-bit switches, in which 72% of the faults dedicated to a single ports of the switch. A structural test and the proposed diagnosis method are applied.

The test results are used to create two cycle accurate SystemC simulation models [13] of the degraded NoC. In the first model, every failing switch is completely isolated by deactivating the ports of all adjacent switches. This represents the state of the art. In the second model, only the ports necessary to tolerate the faults are deactivated according to our proposed diagnosis approach. If a degraded switch looses its direct connection to a core but is still able to forward traffic through neighboring switches, the number of cores decreases while the available bandwidth remains unchanged. If a degraded switch looses connections to some neighboring switches but the core remains connected, the available bandwidth decreases with constant number of cores. Therefore, two performability measurements are considered independently: (1) the connectivity, and (2) the communication performance.

### *6.2.1 Connectivity*

The user expects from the system that all available cores can communicate with each other. Let $a$ and $b$ be two arbitrary nodes visible to the user. To ensure the requirement above, there must be a communication path both from $a$ to $b$ and from $b$ to $a$. If we consider the cores to be nodes in a graph and add edges between all node pairs which are able to communicate bidirectionally, then the available cores are just the ones contained in the largest clique of this graph.

As explained earlier, the performability gain of using degraded switches is achieved irrespective to the network size. However, in order to demonstrate the efficiency of the proposed method a relatively big 20x20 NoC has been chosen for the experiments. Table 3 compares the average number of linked cores (the average size of the largest cliques) in both models for various fault counts in a 20x20 NoC. The first two columns indicate to the result of the networks with 12-bit switches and the result of the networks with 32-bit switches are depicted in second two columns. In both cases, we observe that the number of usable nodes increases significantly by using degraded switches. This is due to two factors. First, cores adjacent to defective switches are often still reachable because only a port towards a neighboring switch is affected. This is of course not possible, if a switch is completely disabled. Second, a completely disabled switch may lead to a partitioning of the network.

In a NoC with 32-bit switches and 20 injected faults 139 cores get separated from the network in average (last row in table 3), if every defective switch is completely turned off. Only a small subset of these inaccessible cores are directly connected to a defective switch (maximum 20 cores), the majority of the cores become inaccessible due to the lack of necessary communication paths within the degraded NoC. With fine-grained port deactivation, the number of inaccessible cores improves drastically to 32 in average. But still, the number of inaccessible cores is larger than the actual number of faults in the network. Furthermore, the defect density causing 20 faults in the NoC switches of course also affects the cores themselves. As the cores usually occupy a larger area on the chip as compared to the area

**Table 3** Number of linked cores (averaged over 100 defect conditions) in a 20x20 NoC

| | 12-bit switch | | 32-bit switch | |
|---|---|---|---|---|
| # faults | with degraded switches | removed defective switches | with degraded switches | removed defective switches |
| 1 | 399.57 | 399.00 | 399.74 | 399.00 |
| 2 | 399.25 | 398.00 | 399.45 | 398.00 |
| 3 | 398.94 | 393.55 | 399.22 | 397.03 |
| 4 | 398.49 | 392.52 | 395.05 | 392.29 |
| 5 | 394.24 | 391.01 | 398.39 | 391.47 |
| 7 | 390.04 | 371.15 | 396.18 | 391.77 |
| 9 | 393.28 | 365.34 | 390.36 | 372.72 |
| 11 | 392.03 | 368.14 | 390.12 | 360.01 |
| 13 | 385.14 | 347.36 | 376.32 | 325.75 |
| 15 | 387.45 | 329.74 | 384.75 | 314.82 |
| 17 | 362.06 | 294.92 | 373.66 | 277.55 |
| 20 | 365.95 | 273.42 | 368.08 | 261.87 |

of the switches, such a defect density will lead also to many defective cores as well. The same argument holds for systems with 12-bit switches and chips with more than 20 faults in the NoC itself are considered useless. Therefore, the experiments are reported only for up to 20 faults.

### *6.2.2 Communication Performance*

The second performability measure is concerned with the communication performance provided to the available cores. The performance is measured by recording the number of packet drops under various network loads.

In both models, each switch is connected to a traffic generator core, which generates uniform traffic with various loads. Each core is sending messages to any other core with equal probability. Let $\varDelta t_{\min}$ be the minimum time interval between two consecutive packet injections of a core. If a core sends a packet at time $t$, it may send the next packet at time $t' \geq t + \varDelta t_{\min}$. With $\varDelta t_{\text{avg}} \geq \varDelta t_{\min}$ being the average time between two actual packet transmissions, the load per core is defined as:

$$\text{load}_{\text{core}} = \frac{\varDelta t_{\min}}{\varDelta t_{\text{avg}}} \cdot 100\%$$

All active cores in the network produce traffic with the same load, $\text{load}_{\text{core}}$. In a degraded network, some cores may be deactivated as described before and the overall load of the network decreases by the ratio of the number of linked cores over the network size:

$$\text{load}_{\text{net}} = \text{load}_{\text{core}} \cdot \frac{\text{linked cores}}{\text{network size}}$$

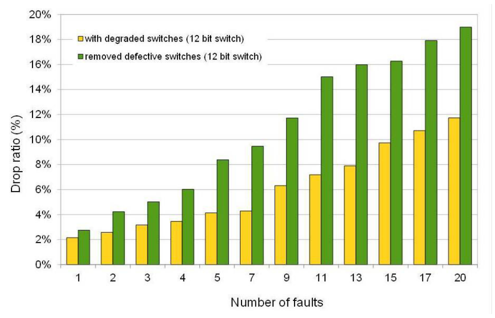
If the network is fault-free and all cores are active, then the network load is the same as $\text{load}_{\text{core}}$.

In a fault-free network, no packets are dropped because as long as $\Delta t_{\text{avg}} \geq \Delta t_{\min}$, there are sufficient network resources to guarantee packet delivery. When we increase the load on a degraded network, packets will be dropped due to overflowing FIFO-buffers. The ratio of dropped packets

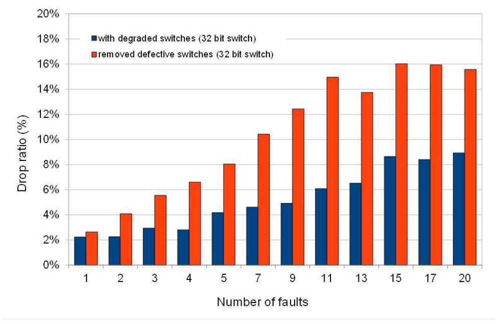
$$\text{drop ratio} = \frac{\text{number of dropped packets}}{\text{number of injected packets}} \cdot 100\%$$

indicates the communication performance retained in the degraded network.

The drop ratio is now measured separately under different loads and different numbers of faults both on the network with completely disabled switches and the network with degraded switches. Figure 7 compares the average drop ratios in presence of various number of faults in 20x20 NoCs with a fixed $\text{load}_{\text{core}}$=14.5%. The experiment has been repeated for both 12-bit switches and 32-bit switches.



(a) 12-bit switch model



(b) 32-bit switch model

**Fig. 7** Average drop ratios in a 20x20 NoC with various fault counts under $\text{load}_{\text{core}}$=14.5%

In both diagrams, we observe that compared to NoCs in which all defective switches are completely disabled, the drop ratio is cut in half by using the proposed method particularly in the range of 4 to 13 faults in the network. This shows clearly the value of the additional communication paths provided by the degraded switches especially with high fault densities. However, it is observed that in presence of 15, 17, and 20 faults the difference between the drop ratio of the two networks is reduced. This is justified with the big difference between the number of linked cores in the networks as reported in table 3. As the number of cores in the network with removed defective switches reduces remarkably, the network is loaded with less traffic (i.e. $\text{load}_{\text{net}}$ decreases). Therefore, the number of injected packets decreases, and there are more network resources available per injected packet, thus leading to a reduced drop ratio.

Similar performance gains can be observed under all network loads. Figure 8 shows a plot of average drop ratios over all network loads ($\text{load}_{\text{net}}$) in 20x20 NoCs with 9 faults injected. The network with degraded switches is able to deliver more packets under every traffic condition, too. Moreover, figure 8 shows that the overall network load ($\text{load}_{\text{net}}$) handled by the NoCs with degraded switches is actually higher than the load handled by the NoCs with disabled switches. This is due to the fact, that the number of linked cores (table 3) is higher in this case and each additional core adds to $\text{load}_{\text{net}}$. For instance, with a constant value $\text{load}_{\text{core}} = 14.5\%$ the network load for NoCs with 12-bit switches are $\text{load}_{\text{net}} = 13.24\%$ in the case of disabled switches and $\text{load}_{\text{net}} = 14.26\%$ in the case of degraded switches.

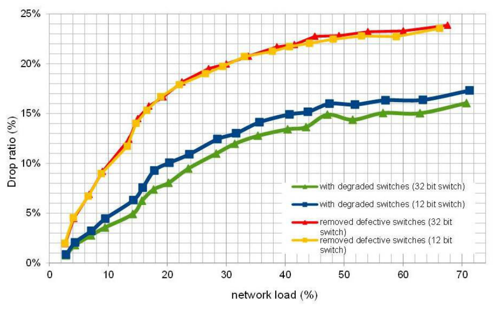


**Fig. 8** Average drop ratios under load in a 20x20 NoC with 9 faults

## 7 Conclusion

We presented a new testing approach to determine the intact functions of defective switches in an NoC. Instead of disabling defective switches completely, these unaffected functions are retained to improve the total NoC performance. Our approach has a bigger advantage especially for the larger switches which support bigger flit sizes. As experiments show about 56% of faults in the random logic of a typical 12-bit switch, and more than 72% of faults of a 32-bit switch can be tolerated by deactivating only one switch port. This fine-grained configuration increases the number of pair-wise connected cores and improves the communication performance compared to an NoC where every faulty switch is disabled completely.

## References

1. A. Agarwal, C. Iskander, and R. Shankar, "Survey of network on chip (NoC) architectures & contributions," *Journal of Engineering, Computing and Architecture*, vol. 3, 2009.
2. L. Benini and G. De Micheli, "Networks on chips: A new SoC paradigm," *Computer*, vol. 35, no. 1, pp. 70–78, Jan 2002.
3. P. Pande, C. Grecu, A. Ivanov, R. Saleh, and G. De Micheli, "Design, aynthesis, and test of networks on chips," *Design & Test of Computers*, vol. 22, no. 5, pp. 404–413, Sep-Oct 2005.
4. A. Agarwal, B. Paul, and K. Roy, "A novel fault tolerant cache to improve yield in nanometer technologies," in *Proc. 10th International On-Line Testing Symposium (IOLTS'04)*, 2004, pp. 149–154.
5. C. Grecu, P. Pande, B. Wang, A. Ivanov, and R. Saleh, "Methodologies and algorithms for testing switch-based noc interconnects," in *Proc. 20th International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'05)*, Oct 2005, pp. 238–246.
6. A. Amory, E. Briao, E. Cota, M. Lubaszewski, and F. Moraes, "A scalable test strategy for network-on-chip routers," in *Proc. IEEE International Test Conference (ITC'05)*, Nov 2005, p. 25.1.
7. M. Hosseinabady, A. Dalirsani, and Z. Navabi, "Using the inter- and intra-switch regularity in NoC switch testing," in *Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE'07)*, Apr 2007, pp. 361–366.
8. D. Fick, A. DeOrio, J. Hu, V. Bertacco, D. Blaauw, and D. Sylvester, "Vicis: A reliable network for unreliable silicon," in *Proc. 46th ACM/IEEE Design Automation Conference (DAC'09)*, Jul 2009.
9. Z. Zhang, A. Greiner, and S. Taktak, "A reconfigurable routing algorithm for a fault-tolerant 2D-mesh network-on-chip," in *Proc. 45th ACM/IEEE Design Automation Conference (DAC'08)*, Jun 2008, pp. 441–446.
10. Y. Fukushima, M. Fukushi, and S. Horiguchi, "Fault-tolerant routing algorithm for network on chip without virtual channels," in *Proc. 24th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'09)*, Oct 2009, pp. 313–321.
11. S.-Y. Lin, W.-C. Shen, C.-C. Hsu, and A.-Y. A. Wu, "Fault-tolerant router with built-in self-test/self-diagnosis and fault-isolation circuits for 2D-mesh based chip multiprocessor systems," *International Journal of Electrical Engineering*, vol. 16, no. 3, pp. 213–222, 2009.
12. M. Palesi, S. Kumar, and V. Catania, "Leveraging partially faulty links usage for enhancing yield and performance in networks-on-chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 3, pp. 426–440, Mar 2010.
13. A. Kohler, G. Schley, and M. Radetzki, "Fault tolerant network on chip switching with graceful performance degradation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 6, pp. 883–896, Jun 2010.
14. Y.-C. Chang, C.-T. Chiu, S.-Y. Lin, and C.-K. Liu, "On the design and analysis of fault tolerant noc architecture using spare routers," in *Design Automation Conference (ASP-DAC), 2011 16th Asia and South Pacific*, jan. 2011, pp. 431 –436.
15. M. Kakoee, V. Bertacco, and L. Benini, "Relinoc: A reliable network for priority-based on-chip communication," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, march 2011, pp. 1 –6.
16. C. Grecu, P. Pande, A. Ivanov, and R. Saleh, "BIST for network-on-chip interconnect infrastructures," in *Proc. 24th IEEE VLSI Test Symposium (VTS'06)*, May 2006, pp. 30–35.
17. J. Raik, R. Ubar, and V. Govind, "Test configurations for diagnosing faulty links in NoC switches," in *Proc. 12th IEEE European Test Symposium (ETS '07)*, May 2007, pp. 29–34.
18. G.-M. Chiu, "The odd-even turn model for adaptive routing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 7, pp. 729–738, Jul 2000.
19. M. Ali, M. Welzl, M. Zwicknagl, and S. Hellebrand, "Considerations for fault-tolerant network on chips," in *Proc. 17th International Conference on Microelectronics (ICM'05)*, Dec 2005, pp. 178–182.
20. A. Alaghi, N. Karimi, M. Sedghi, and Z. Navabi, "Online NoC switch fault detection and diagnosis using a high level fault model," in *Proc. 22nd International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT'07)*, 2007, pp. 21–29.
21. J. Raik, V. Govind, and R. Ubar, "Design-for-testability-based external test and diagnosis of mesh-like network-on-a-chips," *IET Computers Digital Techniques*, vol. 3, no. 5, pp. 476–486, Sep 2009.
22. A. Kohler and M. Radetzki, "Fault-tolerant architecture and deflection routing for degradable noc switches," in *Proc. 3rd ACM/IEEE International Symposium on Networks-on-Chip (NOCS '09)*, 2009, pp. 22–31.
23. L. M. Huisman, *Data Mining and Diagnosing IC Fails.* Springer, Sep 2005.
24. S. Holst and H.-J. Wunderlich, "Adaptive debug and diagnosis without fault dictionaries," *Journal of Electronic Testing – Theory and Applications (JETTA)*, vol. 25, no. 4-5, pp. 259–268, Aug 2009.
25. T. Bartenstein, D. Heaberlin, L. M. Huisman, and D. Sliwinski, "Diagnosing combinational logic designs using the single location at-a-time (SLAT) paradigm," in *Proc. IEEE International Test Conference (ITC'01)*, Oct 2001, pp. 287–296.
26. L. M. Huisman, "Diagnosing arbitrary defects in logic designs using single location at a time (SLAT)," *IEEE Transactions on Cumputer Aided Design of Integrated Circuits and Systems*, vol. 23, no. 1, pp. 91–101, Jan 2004.
27. H.-J. Wunderlich and S. Holst, "Generalized fault modeling for logic diagnosis," in *Models in Hardware Testing*, H.-J. Wunderlich, Ed. Springer, 2009, pp. 159–184.
28. S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and S. Borkar, "An 80-tile sub-100-w teraflops processor in 65-nm cmos," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 29 –41, Jan 2008.