Structural Test for Graceful Degradation of NoC Switches

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

Proceedings of the 16th IEEE European Test Symposium (ETS'11) Trondheim, Norway, 23-27 May 2011

doi: http://dx.doi.org/10.1109/ETS.2011.33

Abstract: Networks-on-Chip (NoCs) are implicitly fault tolerant due to their inherent redundancy. They can overcome defective cores, links and switches. As a side effect, yield is increased at the cost of reduced performability. 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 rather than providing only a pass/fail result for the complete switch. The new method combines for the first time the precision of structural testing with information on the functional behavior in the presence of defects to determine the unaffected switch functions and use partially defective NoC switches. According to the experimental results, this improves the performability of NoCs as more than 61% of defects 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 any switch design.

[1] **IEEE COPYRIGHT NOTICE**
©2011 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Structural Test for Graceful Degradation of NoC Switches

Atefe Dalirsani, Stefan Holst, Melanie Elm, Hans-Joachim Wunderlich
Institut für Technische Informatik
Universität Stuttgart
Pfaffenwaldring 47; D-70569 Stuttgart, Germany
email: {dalirsani, holst, elm, wu}@iti.uni-stuttgart.de

***Abstract*—Networks-on-Chip (NoCs) are implicitly fault tolerant due to their inherent redundancy. They can overcome defective cores, links and switches. As a side effect, yield is increased at the cost of reduced performability. 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 rather than providing only a pass/fail result for the complete switch.**

The new method combines for the first time the precision of structural testing with information on the functional behavior in the presence of defects to determine the unaffected switch functions and use partially defective NoC switches. According to the experimental results, this improves the performability of NoCs as more than 61% of defects 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 any switch design.

***Index Terms*—Network-on-Chip, Graceful Degradation, Performability, Logic Diagnosis**

I. 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], [2], [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. In many cases, 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], [6], [7], at power-up or on demand [8]. Switches that fail these structural tests are disabled and isolated [9]. The NoC compensates 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], [10], [11], [12].

By disabling defective switches, the overall performance of the system decreases because cores can get isolated from the network and the diverged traffic can cause congestion. The ability of preserving performance while tolerating a certain number of defects in a design is called *performability*. The performability is increased, if not only complete switches can be disabled but also defective links and ports in a more fine grained way [13].

Graceful degradation in the NoCs has been already discussed in the literature. But, none of the previous studies have used the standard test flow for identifying fault-free switch ports. This is the main contribution of the current work. In this paper, for the first time, a technique based on structural test and diagnosis methods is presented for identifying and retaining the fault-free switch ports for system use and therefore improving performability and yield.

The approach here is a completely new technique based on the standard testing scheme, which can deal with an arbitrary class of switches in order to handle graceful degradation. Unlike the method presented in [8], the method does not impose any additional hardware overhead since we assume the standard flow and architecture of industrial volume test to determine defect locations inside the NoC switches. 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 III, IV, and V describe the identification of the remaining functionality of a switch. Section VI applies this method to a typical switch and presents a thorough performability analysis with NoCs of various configurations.

II. Problem Definition and State of the Art

The overall flow of the approach is depicted in figure 1. 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 1).
2) For each possible function of the switch, it is determined if it may be affected by the identified defect (Functional

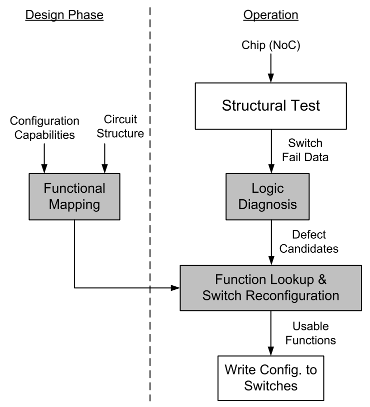

Fig. 1. Overall flow

Mapping block of figure 1).

3) Finally, the ports associated with the affected functions are disabled (Function Lookup & Switch Reconfiguration block in figure 1).

The method can be applied to any NoC architecture with arbitrary switches adhering to the following conditions:

- A requirement for degrading an NoC is that 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], [14], [7], [15] already satisfy these requirements. This condition generally has to be fulfilled for a proper production test.
- 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] through adding a single bit to any input and output port, which 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.
- The NoC must implement a fault tolerant routing scheme like [16], [13], [17], [9] to guide diverged traffic over alternative routes to the correct destinations.

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 [15], [18], [19], [20].

Compared to the previous fault tolerant approaches for identifying switch portions to be deactivated [11], [8], our method does not introduce any hardware overhead and utilizes the available test structures which must be 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.

III. Structural Fault Diagnosis

According to section II, the first step of the proposed method consist of well-established test and diagnosis techniques. The new contribution is the mapping of all possible structural faults f from a fault model F to switch functions, which is described in detail in section IV (step 2). The first step is performed once for every NoC during production test and explained in the following paragraphs.

Structural fault diagnosis [21], [22], [23], [24] 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.

Simulation based cause-effect diagnosis by using the single location at-a-time (SLAT) paradigm [23] is an appropriate technique for finding single defects. This technique uses the simulation results for all single stuck-at faults. As we cannot assume that a punctual defect really behaves like a stuck-at fault, a more general fault model of single fault sites will be used.

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. In the worst case, the victim signal v will *always* generate a faulty value, i.e. the opposite of the good value. We call this model *unconditional line flip* or $v \oplus [1]$.

The so-called conditional line flip model consists of a fault site b and a condition $[cond]$ which is described by a Boolean, temporal or even random expression. The term $b \oplus [\overline{b_{-1}} \wedge b]$, for instance, describes a slow to rise transition fault from time $t = -1$. More details are found in [25]. By using a test set with sufficiently high resolution, the algorithm from [22] 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.

Now, consider 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 the next chapter (step 2) has identified, which switch functions are affected by f. The affected functions and ports are looked up and the according reconfiguration is initiated (step 3).

Figure 2 presents an example of using a semi-faulty switch in the NoC. In this 2D mesh, production test, subsequent diagnosis and functional mapping have evaluated that defects 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 as described in [13], e.g.

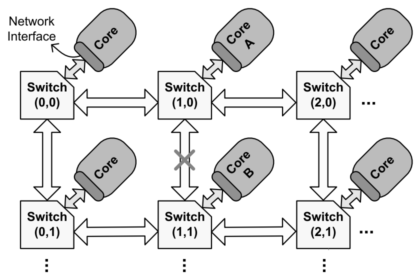

Fig. 2. 2D grid topology

The following chapters explain, how to determine, which functions and ports are affected by a fault f.

IV. 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 3 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 control logic. Besides, the control input assignments for forcing the switch into any of its functions $x \triangleright y$ are known.

A. 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 states, 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.

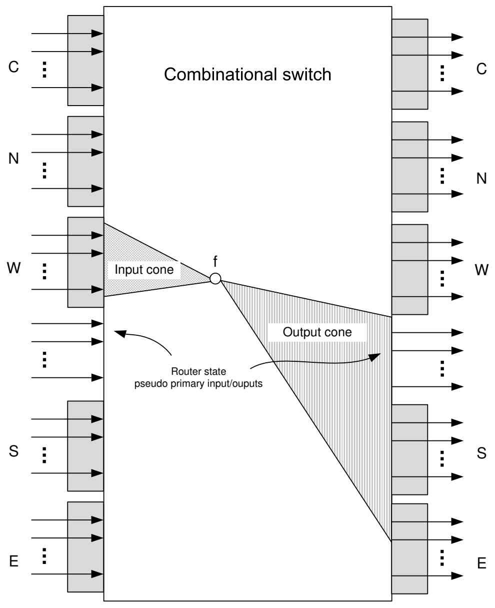

Fig. 3. Combinational logic of the switch with a fault f.

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.

B. 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 recent 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) 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 states.

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) Function conditions.
   For each function $x \triangleright y$ not ruled out by the topological preprocessing and y not disabled in step 2: We mask now 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.

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.

V. 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 2 and 3 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 in step 3. Then, all output ports disabled in step 2 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 in step 3) 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 $P = \{N, S, W, E, C\}$ step 3 indicates that functions $N \triangleright C$, $N \triangleright S$, and $W \triangleright E$ are disabled. Also, step 2 denotes that output port E is disabled. The graph is constructed like figure 4(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 4(b) are the ports that have to be disabled in order to confine the fault effects.

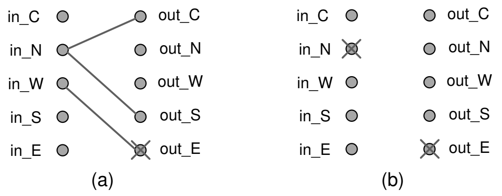

Fig. 4. Minimal vertex cover for a faulty switch

VI. Experimental Results

In this section, we present experimental results on a typical switch designed for NoC mesh architectures. Figure 5 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. Corresponding to [7] and [16], the router implements a fault tolerant wormhole XY routing scheme and processes each input port in a round robin fashion.

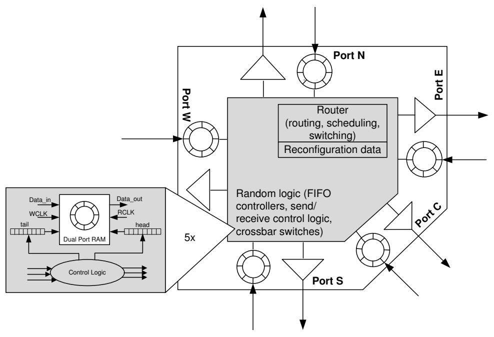
Port N
Port E
Port W
Router (routing, scheduling, switching)
Reconfiguration data
Random logic (FIFO controllers, send/ receive control logic, crossbar switches)
Port C
Port S
5x
Data_in
WCLK
tail
Dual Port RAM
Data_out
RCLK
head
Control Logic

Fig. 5. A typical NoC switch

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.

The router, the crossbar and additional control logic to generate and process the handshake signals of the ports are random logic synthesized from VHDL and mapped to the lsi10k technology library. Since recent commercial ATPG tools do not support the line flip fault model, we conducted our experiments on the conditional stuck-at fault model.

A. Remaining Functionality

The switch has been analyzed using the method presented, and the dictionary was created, which maps each fault to specific input or output ports to be deactivated. Table I shows the statistics of the generated dictionary. We observe that more than 61% of the total random logic of the 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.

TABLE I. Portion of dedicated random logic for each port.

Port	Stuck-at faults	
	count	%
input C	194	5.90%
output C	200	6.05%
input S	195	5.90%
output S	209	6.32%
input W	195	5.90%
output W	215	6.50%
input N	195	5.90%
output N	212	6.41%
input E	195	5.90%
output E	213	6.40%
Sum	2023	61.18%

B. Performability

This section details the amount of performance preserved by our 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. 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.

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.

Table II compares the average number of linked cores (the average clique sizes) in both models for various fault counts in a 20x20 NoC. 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.

TABLE II. Number of linked cores in a 20x20 NoC

#faults	#linked cores (averaged over 100 defect conditions)	
	with degraded switches	removed defective switches
1	399.57	399.00
2	399.25	398.00
3	398.94	393.55
4	398.49	392.52
5	394.24	391.01
7	390.04	371.15
9	393.28	365.34
11	392.03	368.14
13	385.14	347.36
15	387.45	329.74
17	362.06	294.92
20	365.95	273.42

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 $\Delta t_{\min}$ 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 $t' \geq t + \Delta t_{\min}$. With $\Delta t_{\text{avg}} \geq \Delta t_{\min}$ being the average time between two actual packet transmissions, the load per core is defined as

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

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 a switch only accepts new data if its delivery can be guaranteed. 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 6 compares the average drop ratios in presence of various number of faults in 20x20 NoCs with a fixed $\text{load}_{\text{core}}$=14.5%.

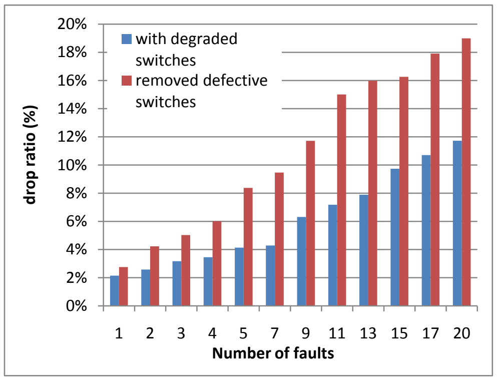

Fig. 6. Average drop ratios in a 20x20 NoC with various fault counts under $\text{load}_{\text{core}}$=14.5%.

Compared to NoCs in which all defective switches are completely disabled, the drop ratio is almost cut in half by using the proposed method if more than 4 faults are present in the network. This shows clearly the value of the additional communication paths provided by the degraded switches especially with high fault densities. Moreover, the overall network load (load_{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 II) is higher in this case and each additional core adds to load_{net}. For instance, with a constant value $\text{load}_{\text{core}} = 14.5\%$ the network loads for NoCs with 9 faults 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.

Similar performance gains can be observed under all network loads. Figure 7 shows a plot of average drop ratios over all network loads (load_{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 7 shows that even the maximum possible load_{net} improves from 66.22% to 71.28% in this case.

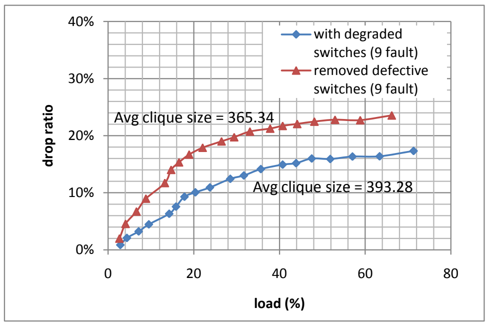

Fig. 7. Average drop ratios under load in a 20x20 NoC with 9 faults.

VII. 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. The experiments show that about 61% of faults in the random logic of a typical switch can be tolerated by deactivating only one switch port. This fine-grained configuration increases the number of pairwise 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] 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.

[15] J. Raik, R. Ubar, and V. Govind, "Test configurations for diagnosing faulty links in NoC switches," in *European Test Symposium, 2007. ETS '07. 12th IEEE*, May 2007, pp. 29–34.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] L. M. Huisman, *Data Mining and Diagnosing IC Fails*. Springer, Sep 2005.

[22] 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.

[23] 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.

[24] 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.

[25] 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.