On Covering Structural Defects in NoCs by Functional Tests

Atefe Dalirsani, Nadereh Hatami, Michael E. Imhof, Marcus Eggenberger, Gert Schley,
Martin Radetzki, Hans-Joachim Wunderlich
Institute of Computer Architecture and Computer Engineering, University of Stuttgart, Germany
{dalirsani, hatami, imhof, eggenbms, schley, radetzki, wu}@informatik.uni-stuttgart.de

***Abstract*—Structural tests provide high defect coverage by considering the low-level circuit details. Functional test provides a faster test with reduced test patterns and does not imply additional hardware overhead. However, it lacks a quantitative measure of structural fault coverage. This paper fills this gap by presenting a satisfiability based method to generate functional test patterns while considering structural faults. The method targets NoC switches and links, and it is independent of the switch structure and the network topology. It can be applied for any structural fault type as it relies on a generalized structural fault model.**

***Index Terms*—Network-on-Chip (NoC), Functional Test, Functional Failure Modeling, Fault Classification, Boolean Satisfiability (SAT)**

I. Introduction

Network-on-Chips (NoCs) constitute the interconnection infrastructure of today's massively parallel many-core architectures. As silicon technologies continuously shrink, NoCs like other hardware devices become increasingly vulnerable to process and runtime variations [1]. Variability results in complex structural faults which can lead to unpredictable errors. A single fault in the NoC may cause packets to be dropped or become corrupted, resulting in incoherent and erroneous traffic, ultimately causing the entire chip to fail [2].

While structural testing targets certain structural fault models and tries to prove the absence of these faults, functional testing validates some specified functionalities. Due to complexity reasons, this is inherently incomplete, and fault simulation shows usually a rather limited fault coverage obtained by functional test strategies [3, 4]. Yet functional testing has still some benefits over a structural test as it does not need a separate test mode, it can be applied in system speed and it may even detect faults not covered otherwise [5–7].

The goal of the paper at hand is to combine the benefits of structural and functional test of NoCs. It presents an automated approach to generate functional test patterns with high structural fault coverage. Functional test patterns can be applied during the normal operation of a switch interleaving the normal traffic. The conditional line flip (CLF), as introduced in [8], is used to specify any type of structural faults. A formal satisfiability-based (SAT) approach classifies structural faults into functional failure classes. Fault classification is specially useful to extend the functional failure classes, so that the structural fault coverage of the corresponding functional test increases. It determines which structural faults cause a certain functional failure. Besides, it provides a weighted functional failure classification with respect to the number of structural faults in each class. The method includes four tasks:

1) Definition of functionalities of an NoC switch, and formalization of the corresponding failure modes.
2) Mapping the failure modes to the switch structure in the form of clauses. This allows test generation by modern satisfiability solvers (SAT).
3) Modeling structural faults (not just stuck-at faults) by clauses and adding these clauses to the failure mode description.
4) Solving the SAT problem allows now to generate data input for the functional test and to quantify the structural faults covered by each of the functional failure modes.

The outcome of this method is functional data packets for the switches and links which can be applied in system mode and form highly effective test sequences. The experimental results show that functional tests generated this way achieve a significantly higher fault coverage than the ones obtained by commercial sequential ATPG tools.

The paper is organized as follows: The next section starts with a brief outlook of state of the art in functional NoC testing. It then explains how functional failures are modeled and gives an overview of the classification and test generation method. Section III discusses the structural fault injection procedure while section IV provides a formal modeling for functional failure modes. Section V describes the fault classification approach which is used in section VI to describe the functional test pattern generation method. The experimental results are discussed in section VII.

II. Basics of Functional NoC Testing

A. State of the Art

So far, functional NoC test approaches [9–13] do not explicitly consider structural faults, which results in a lower fault coverage compared to structural tests. Aisopos et al. [14] propose a modeling of variation-induced faults in NoCs to study the impact of delay faults at the system level. In [15], a software-based self-test approach generates test patterns targeting structural faults where the switch under test still must go to the test mode.

The correlation of structural faults to high level faults of an NoC has a key role in the success of a functional test methodology [16]. For this reason, functional failure modes must be carefully defined. The next subsection describes the failure modes analyzed in this paper.

B. Functional Failure Modes for NoC Switches

The specification of an NoC switch implies the following functionalities:

- The received data is routed via the correct output port.
- The data is left intact.
- No data is lost.
- No new data is generated.

Accordingly, the functional failure modes of an NoC switch are defined as:

- Misrouting: The received packet is routed to the wrong output port. This fault may cause deadlock in the network.
- Data corruption: The data is corrupted for at least one flit in the packet.
- Packet/flit loss: At least one flit of the received packet is never delivered to the output port of the switch.
- Garbage packet/flit: A new packet/flit is generated and routed to the output port. This includes routing a received packet to more than one output port, or generating spurious flits among the flits of a packet.

C. Method Overview

In order to classify and weight the failure modes and additionally generate the corresponding functional test, models have to be generated which include the fault free switch, the faulty instance and the functional failure modes (Fig. 1).

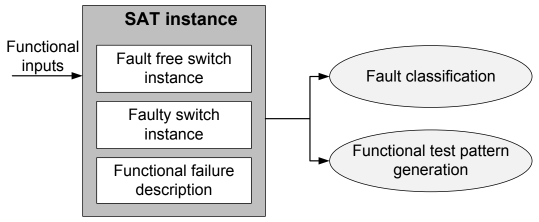
SAT instance
Functional inputs
Fault free switch instance
Faulty switch instance
Functional failure description
Fault classification
Functional test pattern generation

Fig. 1: Overview of the fault classification and test generation method

Functional failures evolve over multiple cycles, and a sequential approach like bounded model checking [17] is a promising option. However, for complexity reasons, it is more attractive to unroll the switch for the number of cycles, namely T, in which the functional failure may affect its behavior. The maximum number of cycles is equal to the sequential depth of the circuit [18]. The faulty instance is a copy of the switch including the literals for the injected structural fault. Modeling the circuit and its faulty instance is the principle of many SAT-based ATPG approaches [19, 20]. According to the target functional failure, appropriate clauses for checking the functional mismatch between the good and faulty copy are added to the model. After obtaining the model, the classification algorithm iteratively searches for a primary input assignment such that the injected structural fault causes the target functional failure. The satisfying assignment is used as a functional test pattern as described in section VI. For the unclassified faults, new functional failure classes may be required.

Before discussing the functional failure models in detail, the structural fault injection is summarized in the next section.

III. Structural Fault Injection Mechanism

As the switch is a sequential circuit, the fault has to be propagated through the internal states before it can eventually be observed at primary outputs. Hence, the corresponding test response to a functional test pattern must be considered in multiple consecutive cycles.

Toward this, we apply the standard technique of time-frame expansion [3] for switches. This technique transports the circuit's sequential behavior from the time to the space domain. The combinational core of the structural switch is extracted by removing the flipflops. Input and output signals of the flipflops are then replaced by pseudo primary output (PPO) and input (PPI) ports. The combinational core of the switch, Φ_c, is transformed to a conjunctive normal form (CNF) using the Tseitin transformation [21]. The sequential behavior of the switch, Φ_s^T, is modeled by time-frame expansion of Φ_c, as in [15]. The literals of the PPIs of each copy are connected to the literals of the PPOs of the previous copy in the SAT instance, as shown in Fig. 2.

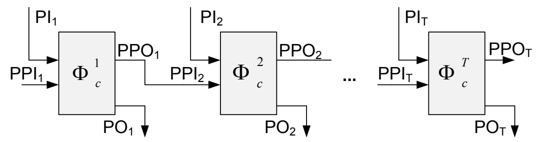

Fig. 2: The time-frame expansion model of an NoC switch [15]

As we do not want to restrict the methodology to a specific structural fault model, the general calculus of conditional line flips (CLF) is used as introduced in [8]. Accordingly, to inject the fault at fault location l_t, the faulty line, l_t^f, is defined by a flip of signal l_t in the good copy, if the condition holds:

$$l_t^f := l_t \oplus d_t$$

where d_t is the condition literal at cycle t. It has been shown that the CLF calculus can describe any complex digital fault model like delay faults, crosstalk and bridges [8]. The condition is defined as a function over time to describe the arbitrary nature of defects. In our model, d_t is a variable that can be freely assigned by the SAT-solver. Since the structural fault exists in all cycles of the unrolled circuit, the CLF clauses are defined for all the copies. With this modeling, the SAT solver searches for all possible assignments for d_t over time (i.e. all possible structural defects) that cause a certain functional failure:

$$\bigwedge_{t=1}^{T}(l_t^f \leftarrow l_t \oplus d_t)$$

However, it is sufficient to detect a CLF in a single time-frame. In addition, d_t can be restricted with respect to a target structural fault. For example, stuck-at-1 on literal l can be modeled in CLF as $l^f := l \oplus \bar{l}$, which is enforced by setting $d_t := \bar{l}_t$.

The faulty instance ($\Phi_{SF}^{f,T}$) for the structural fault f is built from the output cone (also known as downstream logic) of the faulty literals ($\Phi_s^{f,T}$) and the CLF clauses for all copies:

$$\Phi_{SF}^{f,T} = \Phi_s^{f,T} \wedge \bigwedge_{t=1}^{T}(l_t^f \leftarrow l_t \oplus d_t)$$

As the effect of a fault may be latent through several cycles before appearing at the primary outputs, $\Phi_{SF}^{f,T}$ should be able to model fault propagation at different cycles. Therefore, the fault appearance at PPOs (PPOs are in the output cone of the faulty literal) is also considered. If the fault appears at PPOs of the faulty output cone at cycle t, the equivalent input cone at cycle $t+1$ is also added to $\Phi_{SF}^{f,T}$.

IV. Modeling Functional Failure Modes

A. Modeling Approach

In this subsection, the functional failures introduced in section II are defined in a more formal way. We set

- I^n: Set of possible input vectors of n bits,
- $C(i)$: Functional circuit response for input vector i,
- $C^f(i)$: Functional circuit response under fault f for input vector i,
- $f_{in}(i)$: Boolean formula that defines functional input constraints,
- $f_{out}(C(i), C^f(i))$: Boolean formula that defines the functional mismatch.

A fault f is *functionally testable* if and only if:

$$\exists i \in I^n : f_{in}(i) \wedge f_{out}(C(i), C^f(i)).$$

In general, a functional failure is defined by an input characteristic function (f_{in}), an output characteristic function (f_{out}), and T, the number of cycles in which the functional failure is active. The corresponding SAT instance, Φ_{FF}^T, is a conjunction of the clauses of the input and output characteristic functions, represented in conjunctive normal form (CNF):

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

Each NoC switch consists of a number of switch ports (input/output) through which the switch communicates to its neighboring switches or a network interface. The functional failure can be individually defined for each switch port. In the following, the input and output characteristic functions are discussed in detail. Fig. 3 illustrates the underlying switch structure and the involved signals.

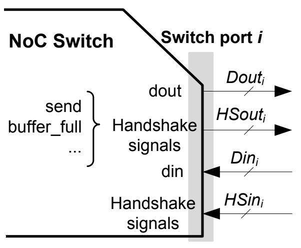

Fig. 3: Switch interfaces

A functional test pattern must fulfill the circuit input specification in the functional mode, for instance the input must be a well-formatted packet. The input characteristic function, f_{in}, is a function of the primary inputs of the switch and specifies the input constraints for the duration of T. Input signals of the switch ports include data lines and handshake controlling signals. Hence, f_{in} is defined over all switch ports as:

$$f_{in} = \bigwedge_{j \in Ports} f(Din_{j,t}, HSin_{j,t})$$

where $Din_{j,t}$ and $HSin_{j,t}$ are the data inputs and handshake signals of the port j at cycle t. In the operational mode, an NoC switch receives just inputs in the form of an NoC packet. According to the packet specification, f_{in} is defined as a Boolean function over the data inputs to guarantee the valid packet format. For example, if a packet starts with a head flit and ends with a tail and the number of flits per packet is given by parameter fpp, f_{in} is defined as:

$$\bigwedge_{j \in Ports} \bigwedge_{t=1}^{T-fpp+1} (Din_{j,t} = \text{head}) \Leftrightarrow (Din_{j,t+fpp-1} = \text{tail}).$$

If the packet is protected with error detection/correction codes, this constraint should also be defined here. For example, if the first bit of the input data, $Din_{j,t}[0]$, is the parity bit for the $Din_{j,t}[1..(k-1)]$, where k is the number of data bits, the following Boolean formula should also be added to f_{in}:

$$Din_{j,t}[0] = Din_{j,t}[1] \oplus Din_{j,t}[2] \oplus \ldots \oplus Din_{j,t}[k-1].$$

Output signals of the switch port include data lines and handshake signals (Fig. 3). The output characteristic function, f_{out}, describes a functional mismatch between the outputs in the good and faulty switch. For the switch ports $j \in Ports$:

$$f_{out} = \bigvee_{j \in Ports} f(Dout_{j,t}, Dout_{j,t}^f, HSout_{j,t}, HSout_{j,t}^f)$$

For $t \in T$, the data lines are denoted by $Dout_{j,t}$ and $HSout_{j,t}$ standing for the handshake signals of the switch port j in the good copy of the switch. Throughout the paper, for any set of signals X of the good copy, X^f denotes the corresponding signals in the faulty copy.

For example, for a simple switch with a send signal for handshake, considering data corruption as the target functional failure, the output characteristic function is defined as:

$$f_{out} = \bigvee_{j \in \mathit{Ports}} \bigvee_{t=1}^{T} (Dout_{j,t} \neq Dout^{f}_{j,t}) \wedge (send_{j,t} \wedge send^{f}_{j,t})$$

where $send_{j,t}$ is the send signal of the port j.

B. Example: Misrouting

Assume a simple switch with *send* as the handshake signal. The *send* signal is set, whenever a valid flit is sent out from the switch port. To model misrouting as the functional failure, the output characteristic function is defined as:

1) The send signal in the good copy is not from the same port as the send signal in the faulty copy. In other words, the packet is sent via a wrong port:

$$\bigvee_{\substack{e,i \in \mathit{Ports}, \\ e \neq i}} \bigwedge_{t=c_h}^{T} (\overline{send}_{e,t} \wedge send^{f}_{e,t}) \wedge (send_{i,t} \wedge \overline{send}^{f}_{i,t}) \quad (1)$$

where $c_h = T - fpp + 1$ enforces the condition to be hold for the length of the packet.

2) The data is intact:

$$\bigvee_{\substack{e,i \in \mathit{Ports}, \\ e \neq i}} \bigwedge_{t=c_h}^{T} (data_{i,t} = data^{f}_{e,t}) \quad (2)$$

where $data_{i,t}$ is the data sent by the good copy of the switch.

The output characteristic function for misrouting will be:

$$f_{out} = (1) \wedge (2)$$

C. Functional Failure Injection

Throughout this paper, we use the six functional failure classes introduced in section II. For T being the number of unrolled copies in the SAT instance, the output characteristic functions (f_{out}) are defined by the equations presented in Table I. In the equations, $send_{i,t}$ and $data_{i,t}$ respectively refer to the send signal and data outputs corresponding to the port i at the cycle t in the good copy of the switch.

Functional Failure	f_{out}
Misrouting	As described in section IV-B
Data corruption	$\bigvee_{i \in \mathit{Ports}} \bigvee_{t=1}^{T} (data_{i,t} \neq data^{f}_{i,t}) \wedge (send_{i,t} \wedge send^{f}_{i,t})$
Flit loss	$\bigvee_{i \in \mathit{Ports}} \bigvee_{t=1}^{T} (send_{i,t} \wedge \overline{send^{f}_{i,t}})$
Packet loss	Flit loss holds for the packet length
Garbage flit	$\bigvee_{i \in \mathit{Ports}} \bigvee_{t=1}^{T} (\overline{send_{i,t}} \wedge send^{f}_{i,t})$
Garbage packet	Garbage flit holds for the packet length

TABLE I: Output characteristics of the Functional Failure Modes

So far, we have generated the CNFs of the sequential switch Φ^{T}_{s}, the faulty instance $\Phi^{f,T}_{SF}$ describing the structural fault and Φ^{T}_{FF} which describes the target functional failure corresponding to Table I. The SAT-instance explaining the relation between the target functional failure and the structural faults, Φ_R, is built using the definition of the functional failure and the good and faulty copy of the switch:

$$\Phi_R = \Phi^{T}_{FF} \wedge \Phi^{T}_{s} \wedge \Phi^{f,T}_{SF}. \quad (3)$$

V. Fault Classification

The SAT instance in Eq. (3) is used for fault classification. By means of classification, the relation between structural faults and the defined functional failure classes is extracted.

The fault classification algorithm is summarized in Alg. V.1. Firstly, the *CreateSAT* function extracts the SAT instance for the target functional failure mode, F. In addition, it builds Φ^{T}_{s}, the good copy of the switch.

Algorithm V.1: Classification($F, \{fl\}, T$)

```
$F$ : functional failure mode
$fl$ : structural fault list
$T$ : cycles for the functional failure activation
$\Phi^{T}_{FF}$ = CreateSAT($F$);
$\Phi^{T}_{s}$ = CreateSAT($\Phi^{T}_{c}$);
while ($\{fl\} <> null$)
  pick $f_i$ from $\{fl\}$
  $\{fl\} = \{fl\} - f_i$
  $\Phi^{f,T}_{SF}$ = CreateSAT($f_i$);
  $\Phi_R = \Phi^{T}_{s} \wedge \Phi^{T}_{FF} \wedge \Phi^{f,T}_{SF}$;
  Boolean $S$ = Solve($\Phi_R$);
  if ($S$)
  $solution = solution \cup \{f_i\}$;
```

The algorithm iteratively picks a fault location from the fault list. The faulty circuit ($\Phi^{f,T}_{SF}$) is then constructed according to the selected fault location as explained in section III. The complete SAT model, Φ_R, is built as in Eq. (3).

The *Solve* routine searches for an assignment to the input variables, so that Φ_R is satisfiable. In other words, SAT searches for a satisfying input pattern and any structural fault $f_i \in fl$, so that the functional failure occurs. It returns true when a solution is found. In this case, the structural fault, f_i, is added to the set of *solutions*. Upon termination of the algorithm, the solution includes a subset of structural faults which cause F.

The classification results can be used to find the appropriate fault tolerant technique for the NoC switch. For more probable functional failure modes, a faster fault tolerant technique is preferred, e.g., retransmission is commonly used to deal with transient faults in NoCs. A check for detecting a functional failure can be done either switch-to-switch or end-to-end. Detecting a functional failure in a switch-to-switch manner requires additional hardware and increases the component's latency. Nevertheless, an end-to-end retransmission introduces a higher performance penalty in case of an error.

The classification does not only quantify the structural faults in the functional failure classes, but also determines which structural fault locations cause certain functional failures. This information can be used to make a cost-aware fault tolerant decision at multiple abstraction levels. For the functional failures that are correctable at higher abstraction levels, no fault tolerant feature is required to protect the corresponding structural fault location at lower levels. For example, data corruption can be detected and even corrected by adding an error detection/correction code in the packet payload. But misrouting is not easily detectable. The classification can identify the structural fault locations which cause misrouting. These structural fault locations should be protected by means of a fault tolerant technique at low level.

VI. Functional Test Pattern Generation

The classification algorithm can also be used to generate functional test patterns. Two conditions must be satisfied:

- The structural fault is activated: The CLF condition is true in at least one time frame and leads to different values of the faulty literals in the good and faulty circuit.
- The functional failure is activated: The functional failure instance is satisfiable.

Fig. 4 shows the functional test generation flow. Primarily, Φ_{FF} is constructed from the *functional failure library* which includes the functional failure modes. The library can be extended in case of insufficient fault coverage.

The functional test patterns are produced using Alg. V.1. Here, the solution includes not only the fault locations, but also the satisfying input assignments which will be served as the functional test pattern.

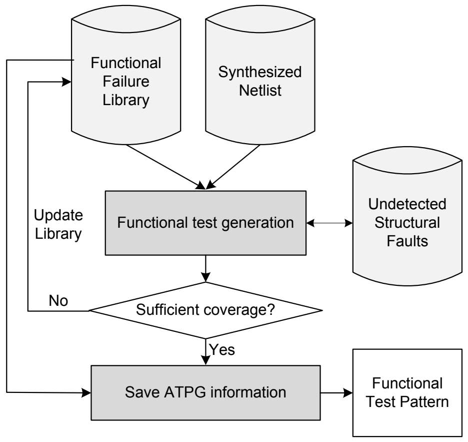

Fig. 4: Functional test generation

Since the functional failure library usually includes more than one failure, a list of functional failures is given to the classification algorithm to construct Φ^T_{FF}. For $FF = \{FF_1, FF_2, ..., FF_m\}$ being the set of m functional failures, Φ^T_{FF} is defined as:

$$\Phi^T_{FF} = \bigvee_{i=1}^{m} \Phi^T_{FF_i} \qquad (4)$$

in which $\Phi^T_{FF_i}$ is the SAT instance which describes the functional failure FF_i. The structural fault must be detectable by at least one functional failure, therefore, a disjunction over $\Phi^T_{FF_i}$ is required. In fact, for test pattern generation, it is sufficient that only one of the functional failures is activated.

If the structural fault coverage is not sufficient, the functional failure library can be updated to improve the coverage, which is beyond the scope of this paper. The classification algorithm can identify the set of structural faults which does not lead to any functional failure in the library.

VII. Experimental Results

The efficiency of the presented fault classification approach is evaluated using a typical switch of the mesh topology. It consists of five input and output ports, crossbar multiplexers, a router, and an additional control logic for handshake signals. Handshake is performed by the *send* and *receive* signals which indicate a valid output or input flit. The *buffer_full* signal shows the state of input buffers as seen in Fig. 3. The switch implements the wormhole XY routing and processes the input channels in a round-robin fashion. The VHDL switch has been synthesized with a commercial synthesis tool in the lsi10k library, which is constrained to one- and two-input gate primitives. The total cell area of the synthesized switch is 14940, where the cell area of a two input nand gate is 1 area unit. The memory elements are equipped with advanced memory BIST, and hence are not considered in the fault classification process. Although the experiments are performed on a mesh switch, the methodology is design independent and can be applied to other architectures as well.

The functional failure library consists of the six functional failure classes of table I. For the experiments, the general CLF fault modeling was restricted to stuck-at faults, but the method can be easily extended to more complex structural faults. The good and faulty switches and the output characteristic functions of the functional failures were modeled as a SAT instance. The sequential depth of the switch determines the number of time frames, T, and the size of the SAT problem. The defined input characteristic function guarantees functional test patterns in the form of NoC packets.

A. Classification

The presented approach enables functional failure classification considering structural faults. Fig. 5 presents the distribution of the structural faults among the functional failure classes. According to the definition, any packet loss can be categorized as flit loss as well. However in the experiments, the faults which cause packet loss have not been counted for flit loss. It is observed that packet loss, data corruption, and garbage packet classes have the highest number of structural faults. This implies that these faults are the most probable functional failures that might be observed due to a structural fault in the switch. Therefore, these functional failures should be detected by a switch-to-switch fault tolerance technique. The other three failure modes are less realistic and may be detected end-to-end.

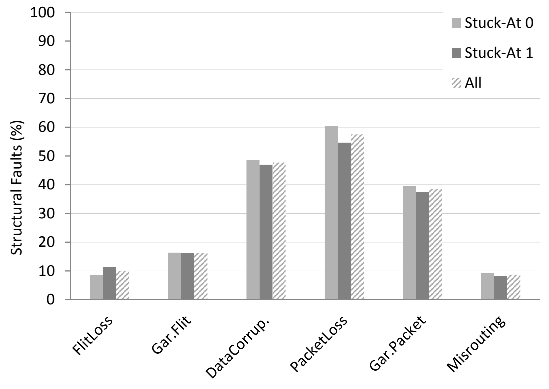

Fig. 5: Distribution of structural faults in functional failure classes

B. Structural Fault Coverage

The presented technique shows that among 27690 stuck-at faults in the switch, 22101 faults are functionally testable; i.e., there exists an input assignment such that the fault causes one of the specified functional failures. The fault coverage of the functional test pattern generation is defined as:

$$\text{Fault coverage}(\%) = \frac{\text{Functionally testable faults}}{\text{Total number of faults}} \times 100 \quad (5)$$

Table II summarizes the coverage results for the proposed functional test pattern generation approach. The proposed scheme achieves 79.82% structural fault coverage. Structural tests like full-scan deliver full fault coverage. However, it must be noted that some faults are functionally redundant and therefore never become testable by a sequential test. As the proposed method is sequential, the results are compared with a sequential ATPG. Toward this, a commercial sequential ATPG tool without any constraints is applied. In the same sequential depth, it reports a fault coverage of 57.53%, while 2.79% of the faults are untestable. The result reveals that there exists a good correspondence between the introduced functional failure modes and the structural faults in the switch. In addition, we have achieved higher fault coverage compared to structural sequential ATPG.

	Fault coverage (%)	Untestable faults (%)
Proposed	79.82	20.18
Sequential ATPG	57.53	2.79

TABLE II: Functional test pattern generation results vs. ATPG

VIII. Conclusion

This paper presented an automated approach for functional test pattern generation for NoC switches with high structural fault coverage. The formal definition of functionalities of the switch and the corresponding failure modes allow mapping of failure modes to the switch structure in the form of clauses. Thus, fault classification and test generation can be conducted by modern SAT solvers.

In addition to the quantification of structural faults covered by each functional failure mode, the experiments show an effective classification resulting in high fault coverage of the generated functional test. Moreover, the classification can be used to select the appropriate fault tolerant technique while trading off the area and timing constraints.

Acknowledgment

This work has been supported by the German Research Foundation (DFG) under grant WU 245/12-1 (ROCK) and Ra 1889/4-1 (ROCK). The authors would like to thank Dr. Stefan Holst for helpful discussions.

References

[1] *International Technology Roadmap*, 2013.
[2] M. Radetzki, C. Feng, X. Zhao, and A. Jantsch, "Methods for Fault Tolerance in Networks-on-Chip," *ACM Computing Surveys*, vol. 46, no. 1, pp. 8:1–8:38, 2013.
[3] M. Bushnell and V. Agrawal, *Essentials of Electronic Testing for Digital, Memory, and Mixed-signal VLSI Circuits*. Kluwer Academic, 2002.
[4] L. Wang, C. Wu, and X. Wen, *VLSI Test Principles and Architectures: Design for Testability*, ser. Systems on Silicon. Elsevier Science, 2006.
[5] P. Maxwell, I. Hartanto, and L. Bentz, "Comparing Functional and Structural Tests," in *Proc. Intl. Test Conf. (ITC)*, 2000, pp. 400–407.
[6] J. Zeng, M. Abadir, G. Vandling, L. Wang, A. Kolhatkar, and J. Abraham, "On Correlating Structural Tests with Functional Tests for Speed Binning of High Performance Design," in *Intl. Test Conf. (ITC)*, 2004, pp. 31–37.
[7] H. Fang, K. Chakrabarty, and H. Fujiwara, "RTL DFT Techniques to Enhance Defect Coverage for Functional Test Sequences," *Journal of Electronic Testing*, vol. 26, no. 2, pp. 151–164, 2010.
[8] H.-J. Wunderlich and S. Holst, "Generalized Fault Modeling for Logic Diagnosis," in *Models in Hardware Testing*. Springer Netherlands, 2010, pp. 133–155.
[9] N. Karimi, A. Alaghi, M. Sedghi, and Z. Navabi, "Online Network-on-Chip Switch Fault Detection and Diagnosis Using Functional Switch Faults," *Journal of Universal Computer Science*, vol. 14, no. 22, pp. 3716–3736, 2008.
[10] 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. VLSI Test Symp. (VTS)*, 2012, pp. 44–49.
[11] A. Frantz, F. Kastensmidt, L. Carro, and E. Cota, "Dependable Network-on-Chip Router Able to Simultaneously Tolerate Soft Errors and Crosstalk," in *Proc. Intl. Test Conf. (ITC)*, 2006, pp. 1–9.
[12] J. Raik, V. Govind, and R. Ubar, "An External Test Approach for Network-on-a-Chip Switches," in *Proc. Asian Test Symp. (ATS)*, 2006, pp. 437–442.
[13] M. Kakoee, V. Bertacco, and L. Benini, "A Distributed and Topology-Agnostic Approach for On-line NoC Testing," in *Proc. intl. Symp. on Networks on Chip (NoCS)*, 2011, pp. 113–120.
[14] K. Aisopos, C.-H. Chen, and P. Li-Shiuan, "Enabling System-Level Modeling of Variation-Induced Faults in Networks-on-Chips," *Proc. Design Automation Conference (DAC)*, pp. 930–935, 2011.
[15] A. Dalirsani, M. E. Imhof, and H.-J. Wunderlich, "Structural Software-Based Self-Test of Network-on-Chip," in *Proc. VLSI Test Symposium (VTS)*, 2014.
[16] T. Bengtsson, S. Kumar, and Z. Peng, "Application Area Specific System Level Fault Models: A case study with a simple NoC Switch," *Proc. Intl. Design and Test Workshop (IDT)*, 2006.
[17] E. Clarke, A. Biere, R. Raimi, and Y. Zhu, "Bounded Model Checking Using Satisfiability Solving," *Formal Methods in System Design*, vol. 19, no. 1, pp. 7–34, 2001.
[18] A. Kunzmann and H.-J. Wunderlich, "An Analytical Approach to the Partial Scan Problem," *Journal of Electronic Testing: Theory and Applications (JETTA)*, vol. 1, no. 2, pp. 163–174, 1990.
[19] H. Konuk and T. Larrabee, "Explorations of Sequential ATPG using Boolean Satisfiability," in *IEEE VLSI Test Symposium*, 1993, pp. 85–90.
[20] P. Stephan, R. Brayton, and A. Sangiovanni-Vincentelli, "Combinational Test Generation using Satisfiability," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 9, pp. 1167–1176, 1996.
[21] G. Tseitin, "On the Complexity of Derivation in Propositional Calculus," in *Automation of Reasoning*. Springer Berlin Heidelberg, 1983, pp. 466–483.