Fault-Tolerant Unicast-Based Multicast for Reliable Network-on-Chip Testing

DONG XIANG, Tsinghua University
KRISHNENDU CHAKRABARTY, Duke University
HIDEO FUJIWARA, Osaka Gakuin University

We present a unified test technique that targets faults in links, routers, and cores of a network-on-chip design based on test sessions. We call an entire procedure, that delivers test packets to the subset of routers/cores, a test session. Test delivery for router/core testing is formulated as two fault-tolerant multicast algorithms. Test packet delivery for routers is implemented as a fault-tolerant unicast-based multicast scheme via the fault-free links and routers that were identified in the previous test sessions to avoid packet corruption. A new fault-tolerant routing algorithm is also proposed for the unicast-based multicast core test delivery in the whole network. Identical cores share the same test set, and they are tested within the same test session. Simulation results highlight the effectiveness of the proposed method in reducing test time.

Categories and Subject Descriptors: B.7.3 [**Integrated Circuits**]: Reliability and Testing—*Testability*

General Terms: Reliability, Design, Algorithms

Additional Key Words and Phrases: Fault-tolerant routing, on-chip networks, router testing, core testing, unicast-based multicast

ACM Reference format:
Dong Xiang, Krishnendu Chakrabarty, and Hideo Fujiwara. 2018. Fault-Tolerant Unicast-Based Multicast for Reliable Network-on-Chip Testing. *ACM Trans. Des. Autom. Electron. Syst.* 23, 6, Article 73 (December 2018), 23 pages.
https://doi.org/10.1145/3243214

1 INTRODUCTION

A network-on-chip (NOC) is a promising communication paradigm for core-based system chips (Benini and DeMicheli 2002; Dally and Towles 2001). Testing complex and embedded NOCs, however, remains a challenging problem. To reduce test cost, reuse of the network for NOC test delivery is attractive (Agrawal and Chakrabarty 2012; Cota 2004, 2006; Froese 2010; Grecu 2007; Liu and

Initial version of this article appeared in "A unified test and fault-tolerant multicast solution for network-on-chip designs," in Proc. of IEEE Int. Test Conf., Oct. 2016. This work is supported in part by the National Science Foundation of China under grants 61373021 and 61774097, and the research foundation from the Ministry of Science and Technology under grant 2016YFB1000200.
Authors' address: D. Xiang, School of Software, Tsinghua University, Qinghua Yuan Street 1#, Beijing 100084, China; email: dxiang@tsinghua.edu.cn; K. Chakrabarty, Dept. of Electrical and Computer Engineering, Duke University, Dongpei Lou, 11-316, Durham, NC 27708; email: krish@ee.duke.edu. H. Fujiwara, Faculty of Informatics, Osaka Gakuin University, 2-36-1 Kishibe-minami, Suita, Osaka 564-8511, Japan; email: fujiwara@ogu.ac.jp.

1084-4309/2018/12-ART73 $15.00
https://doi.org/10.1145/3243214

Iyengar 2006; Tran et al. 2008; Xiang 2011, 2016a, 2013a, 2013b, 2016b; Xiang, Chakrabarty and Fujiwara 2016c). However, unicast-based approaches require sequential delivery of all test packets to all cores, which can make the test cost very high (Cota 2004).

Many components, such as routers and cores, in an NOC are identical in many cases. Routers of the same degree are identical (Xiang 2016b), and they share the same test set. In many cases, even all cores in an NOC can be identical. Identical cores also share the same test set. Therefore, multicast approaches have been presented to reduce test cost (Grecu 2007; Xiang 2011, 2016a, 2013a, 2013b, 2016b; Xiang, Chakrabarty and Fujiwara 2016c).

Grecu (2007) proposed to insert an extra network to support multicast. The methods in Xiang (2016a) and Xiang (2011, 2013b) provided unicast-based multicast techniques for test delivery, which do not need to modify the router architecture to support the multicast operations. The multicast-based router testing approaches (Xiang 2013a, 2016b) duplicate test packets to intermediate nodes of the unicast paths. Therefore, the crossbar switch architecture, and the consumption channel designs in Xiang (2013a, 2016b) are slightly more complicated, because an extra mechanism must be provided to avoid deadlocks at the consumption channels.

Most previous test methods (Agrawal and Chakrabarty 2012; Cota 2004, 2006; Froese 2010; Liu and Iyengar 2006; Xiang 2011, 2016a, 2013a, 2013b, 2016b) disregarded network failures when delivering test packets. Our method targets test delivery in an NOC with multiple link/router failures. We present two new fault-tolerant unicast-based multicast algorithms to deliver test packets for routers and cores, respectively in order for test packets to avoid corrupting by faulty components. This is the reason why we call the proposed method *reliable* NOC testing. A unified methodology is proposed to test links, routers and cores. Interconnects are tested by built-in self-test (BIST) and BIST logic is inserted within each router. However, BIST alone is not sufficient to achieve high fault coverage for routers and complex cores. Therefore, routers and cores are tested using deterministic test data to obtain high fault coverage.

A new fault-tolerant unicast-based multicast procedure is proposed to deliver test packets for router testing. Our method selects an identified fault-free router (IFFR) for each router under test. A test packet is delivered to all IFFRs first. It is then forwarded to the routers under test. After the status of all routers have been identified, our method delivers test packets for the cores.

A test set is generated for different classes of cores/routers separately. Each test packet for cores is delivered to all cores in the corresponding class by using the proposed fault-tolerant unicast-based multicast algorithm. Many cores differ only slightly in terms of the logic implementation; we refer to these as *similar cores*; these cores can still share most test packets. Two similar cores can be merged into the same circuit for the purpose of test generation as in Xiang (2011).

The low-power test application scheme from Xiang (2011, 2013b) is utilized here, and it is combined with the power-aware test scheduling from Xiang (2011). The test responses are compacted by an on-chip X-tolerant multiple-input signature-register (MISR) as in Xiang (2013b). The compacted test responses in the MISR are delivered back to the ATE after all test vectors have been applied. This significantly reduces traffic contention, and reduces test power consumption.

Our method handles the testing of links, routers and cores in a single flow. A test packet is delivered using IFFRs to ensure that faulty routers/links do not corrupt the test packet. We assume that the NOC is designed by a baseline fault-tolerant routing scheme, which utilizes virtual networks to avoid deadlocks. Any other fault-tolerant routing algorithm can also be used for this purpose. Our method considers the most popular NOC interconnects, 2D meshes (Dally and Towles 2001).

There are two significant drawbacks in Xiang (2016b): (i) the router test-scheduling problem is not addressed; (ii) the baseline fault-tolerant routing algorithm is not described (Xiang 2016b); (iii) the availability of multiple ports in an ATE is not exploited, and therefore, the test time remains high. In this work, we present the following:

(1) In Xiang, Chakrabarty and Fujiwara (2016c), we introduced the router test scheduling scheme using only a simple example. The section on router testing in this article introduces new directions. We have presented the basic principles of router test scheduling, problem formulation, and test-scheduling algorithm, and finally a case study. In this way, we have filled a major gap in the conference version.
(2) We utilize the multiple ports of the ATE to reduce test cost for router and core testing. We present the new test scheduling algorithms for router testing and core testing, respectively, using ATEs with multiple ports. These advances lead to a considerable reduction in test time.
(3) The baseline fault-tolerant routing algorithm in Section 5 is also new.
(4) The DFT architectures and control schemes for router and core testing are refined.

The main contributions of the proposed method include: (1) A test delivery scheme for router testing based on a new fault-tolerant unicast-based multicast algorithm. This method first forwards a test packet to a number of IFFRs, therefore, test delivery and test application for router testing can be completed concurrently like the core testing method in Xiang (2013b). The proposed router testing scheme delivers test packets by not using a potentially faulty router/link or a router under test. This important feature, not provided by prior methods (Grecu 2007; Xiang 2013b), can significantly reduce test cost for router testing. (2) A fault-tolerant unicast-based multicast algorithm is proposed for core testing in an NOC that may contain both link and router failures. (3) A fault-tolerant routing algorithm is presented. (4) The multiple ports of the ATE are utilized to reduce test cost.

The rest of the article is organized as follows. We present prior work in Section 2. Router testing is partitioned into multiple test sessions in Section 3. Routers of the same degree are identical, therefore, they share the same test set. The fault-tolerant unicast-based multicast algorithm for core testing is presented in Section 4. The baseline fault-tolerant routing algorithm is described in Section 5. Experimental results are presented in Section 6. A new technique is proposed for router and core testing by using the multiple input/output ports of the ATE in Section 7, and corresponding test scheduling algorithms are presented. The article is concluded in Section 8.

2 RELATED PRIOR WORK

Reuse of the communication platform (Cota 2004, 2006; Dalmasso et al. 2008) is a cost-effective test technique for cores in an NOC-based multicore chip. An algorithm based on the list-scheduling technique was proposed to minimize the test cost. Cota and Liu (Cota 2006) proposed a new test scheduling scheme for NOC testing with multiple-port automatic test equipment (ATE). Delivery of test packets by unicast can be very high (Cota 2004, 2006).

The method in Grecu (2007) exploits the inherent parallelism of the data-transport mechanism to reduce test cost for interconnect testing as well as the test-application time. Test-scheduling algorithms were developed based on a unicast scheme and a multicast presented for sequential and concurrent test data transport. Techniques have also been proposed to improve interconnect reuse for NOC testing (Cota 2006; Dalmasso et al. 2008; Tran et al. 2008). A number of DFT techniques have been proposed for NOC testing (Dalmasso et al. 2008; Peterson and Oberg 2007; Tran et al. 2008).

In Radetzki (2013), Fan (1998, 2002), Lin et al. (2016, 2018), Yang (2005), and Yu (2013), the failure mechanisms, fault models, diagnosis techniques, and fault-tolerance routing algorithms for on-chip networks and regular networks were presented. An effective routing algorithm based on a new flow control scheme was presented in Gorgues et al. (2014) in two-dimensional (2D) mesh-based NOCs. Fault-tolerant routing algorithms were proposed for crossbar based on-chip optical

networks (Xiang et al. 2013) and dragonfly networks (Xiang 2017) by mapping the networks into hypercubes using the original local safety information in faulty hypercubes (Xiang 2001).

An NOC effective testing and configuration strategy was proposed in Ghiribaldi et al. (2013). A fast and scalable built-in self-testing and self-diagnosis procedure has to be carried out concurrently at NOC switches. A distributed functional test mechanism was proposed for NOCs, which scales to large-scale networks with general topologies and routing algorithms (Kakoee et al. 2011, 2014). The careful addition of self-test structures allows ElastiNoC (Seitanidis et al. 2014) to provide fully distributed built-in self-test, where testing unfolds in phases and reaches high fault coverage with small test application time.

The first pin-count-aware optimization approach for test data delivery over an NOC was presented in Richter and Chakrabarty (2014), which addressed optimal utilization of the limited I/O resources provided by ATE to reduce test cost. A unified framework for fault diagnosis and subsequent reconfiguration was proposed in NOCs (Parikh and Bertacco 2016). It provides graceful performance degradation with an increasing number of faults. A new deadlock-free routing algorithm was presented by utilizing all fault-free links in the NOC.

Xiang (2011) proposed a unicast-based multicast algorithm for test delivery to logic cores using the NOC as the interconnect. A test packet can be multicast to the identical cores in an NOC using a unicast-based multicast algorithm (Xiang 2011, 2013b, 2016a) without revising the router architecture. Test responses are collected along the reverse paths of the multicast tree in Xiang (2011). A new unicast-based multicast approach was proposed to deliver test packets for router testing in Xiang (2016b). A test packet can be duplicated to intermediate nodes. A new mechanism was thus proposed to avoid consumption channel deadlocks.

Froese (2010) proposed a new test compression scheme for core testing, where only the seeds are delivered to the cores. A test-delivery optimization algorithm was proposed in Agrawal and Chakrabarty (2012) for NOC-based SOCs with hundreds of cores by using a new dynamic programming model. A comprehensive end-to-end solution was proposed for error correction, data collection, and defect diagnosis and replacement for on-chip networks (Shamshiri et al. 2011; Shamshiri and Cheng 2011).

The unicast test delivery techniques in Cota (2004, 2006), Dalmasso et al. (2008), Froese (2010), and Liu and Iyengar (2006) can incur high test cost. This is a key motivation for the proposed work on a fault-tolerant multicast solution for NOC core and router testing.

3 ROUTER TESTING WITH MULTIPLE TEST SESSIONS

Definitions and notation are presented in Section 3.1. The unicast-based fault-tolerant multicast method is proposed for router testing in Section 3.2. A case study for router testing is presented with the unicast-based fault-tolerant multicast routing algorithm in Section 3.3.

3.1 Definitions and Notation

A *unicast* delivers a packet from a single core, called the source, to a single destination. A *multicast* delivers a packet from a single core called the multicast source to multiple cores called destinations in the NOC (McKinley 1994). A *unicast-based multicast* scheme completes a multicast operation by using multiple unicast steps. Therefore, it is not necessary to modify the unicast router architecture (McKinley 1994; Xiang 2011, 2013b) to implement multicast. The test packet is delivered in each unicast step from the current node (it has received the test packet) to the destination. All destinations of the unicast steps are cores (or routers) under test. All nodes in the multicast tree (except the source, ATE) are the cores/routers under test. The number of unicast steps is $\ln(N + 1)$, here N is the number of destinations for the multicast. This method for core

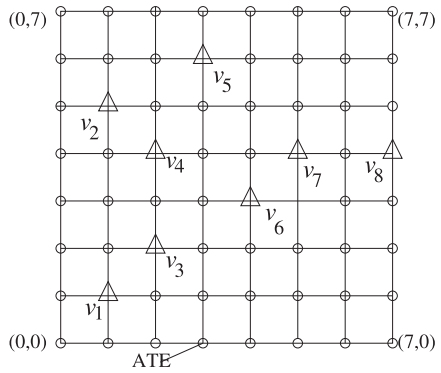


Fig. 1. An illustration of a dimension-ordered chain.

testing is also applicable when the NOC contains completely different cores. In this case, the unicast-based multicast degrades to a unicast.

Let the address of a node x in an n-dimensional mesh (nD) be represented by $(\sigma_1(x), \sigma_2(x), \ldots, \sigma_n(x))$. The binary relation dimension-order, denoted $<_d$, is defined between two nodes x and y as follows: $x <_d y$ if and only if either $x = y$ or there exists an integer j such that $\sigma_j(x) < \sigma_j(y)$ and $\sigma_i(x) = \sigma_i(y)$ for all i, $0 \leq i \leq j-1$. For any set of node addresses, they can be arranged in a unique sequence according to the $<_d$ relation. A sequence of nodes $x_1, x_2, \ldots, x_m$ is a *dimension-ordered chain* (DOC) if and only if (1) $x_i <_d x_{i+1}$ for $1 \leq i < m$ or (2) $x_i <_d x_{i-1}$ for $1 < i \leq m$.

We call this entire procedure, which delivers test packets to a subset of routers, a *test session*. A test session includes the complete procedure associated with the testing of multiple routers. The ATE is the multicast source of a *multicast tree*. An edge (v_1, v_2) is added to the tree if the test packets is delivered from v_1 to v_2 in one unicast step during the test session.

Let $u <_d v <_d w <_d z$. Two packets delivered from u to v, and from w to z (or v to u, and z to w) concurrently with dimension-order routing use no common link (McKinley 1994) in a fault-free mesh. Figure 1 presents a DOC (1,1) (v_1), (1,5) (v_2), (2,2) (v_3), (2,4) (v_4), (3,6) (v_5), (4,3) (v_6), (5,4) (v_7), and (7,4) (v_8) with eight nodes. Two separate fault-tolerant unicast-based multicast algorithms are proposed for router in Section 3 and core testing in Section 4 to reduce potential channel contention. Our fault-tolerant unicast-based multicast algorithm for router testing partitions the set of routers into multiple subsets, where each subset of routers can be accessed by the ATE via one of the IFFRs directly.

3.2 Fault-Tolerant Unicast-Based Multicast

The physical links are tested using a built-in self-test (BIST) scheme as in Xiang (2016b). Assume that the mesh-based NOC contains some link failures, which have been identified by a BIST scheme for physical interconnects. Algorithm 1 presents test scheduling for router testing. The most important differences between the proposed method and the methods in Xiang (2011, 2013b) are as follows: (1) the baseline routing scheme presented in Section 5 avoids link failures and routers/links that have not been identified as being fault-free. The fault-tolerant routing algorithm is cost-effective on the way to avoid deadlocks with a novel virtual network scheme. (2) Test packets are delivered to the selected IFFRs first, and to the routers under test in the last unicast step. (3) The proposed fault-tolerant unicast-based multicast algorithm for router testing is different from Xiang (2016b) and Panda (1999) in that it allows test-data duplication at the intermediate routers along the path in each unicast step. Therefore, the crossbar switch architecture, and the

consumption channel designs in Xiang (2016b) and Panda (1999) require a dedicated mechanism to avoid consumption channel deadlocks.

We cannot broadcast a test packet to all routers/cores, because (1) routers/cores are not all identical, (2) extra mechanism to avoid consumption channel deadlocks is necessary, and (3) no cost-effective deadlock-free fault-tolerant broadcast algorithm for meshes is available.

Routers of the same degree fall into the same class (Xiang 2016b), and they share the same test set. The IFFRs are routers tested in the last test session if possible, and the internal routers that are accessible from the IFFRs can be tested in the current test session. As shown in Algorithm 1, the test process for routers includes the following three parts: (1) internal router testing (degree-4 routers), (2) boundary router testing, and (3) corner router (degree-2) testing.

Our method selects the same number of IFFRs as the number of routers under test in the test session. A test packet is delivered to all selected IFFRs first. The selected IFFRs deliver the received test packets to all routers under test separately in a single unicast step.

It is desirable for the paths in the last unicast step, from all IFFRs to the routers under test, to have no conjoint physical link. This can avoid resource contention. A failure assignment contains some link/router failures. The proposed test scheme partitions the router testing procedure into different test sessions when the NOC contains different failure (links or routers) assignments. In our experiments, link/router failures are randomly injected for the fault-tolerant multicast algorithm.

As shown in Algorithm 1, Q_1 and Q keeps routers under test in the current test session, and fault-free routers tested in the previous test session, respectively. The *while* loop stops when Q becomes empty. Algorithm 1 is different from the previous methods in Grecu (2007) and Xiang (2016b). Test application and test delivery must be completed sequentially in the methods in Grecu (2007) and Xiang (2016b). The test cost can be reduced compared to the previous methods (Grecu 2007; Xiang 2016b). All test packets in the same test session are delivered to routers tested in the test session in tandem. The test vector is applied to the router under test immediately after it has been received. Test delivery and application can be handled concurrently.

Selection of IFFRs for each test session is also important. The details are as follows. First, the routers tested in the last test session are considered as candidates. When the number of IFFRs tested in the last test session is less than the number of routers under test for the current session, other IFFRs can be selected. It is required that the selected IFFRs access the routers under test in the current session along disjoint paths. This process continues until all internal routers have been tested. The boundary routers (degree-3 routers) are tested in the next test session after all internal routers have been handled. The same number of IFFRs are selected to test degree-3 routers. The degree-2 (corner) routers are tested after all all test sessions have been completed.

A test packet is first sent to all selected IFFRs v_i' $(1 \le i \le k)$, where all IFFRs and the source (v) connected to the ATE are arranged as a DOC, and k routers v_1-v_k are tested in a single test session. The test packet is delivered to all selected IFFRs by a recursive procedure as shown in Algorithm 2. The IFFRs are ordered into a DOC (McKinley 1994; Xiang 2011, 2016a). After the test packet has been delivered to all selected IFFRs, it is delivered to all routers under test in the current test session concurrently in the last unicast step. Link contention is also no longer a big problem because of the virtual channel router design and the MISR-based test response collection scheme. The reason why we arrange the destinations of the multicast operation into a dimension-order chain is that we still need to reduce the amount of traffic congestion to reduce test delivery time.

In contrast to McKinley (1994), the proposed method completes a multicast with multiple unicast steps. The ATE is only involved in the first step. However, the multicast source may be involved in all unicast steps for the multicast method in McKinley (1994). This feature can save ATE time, because the second test packet can be sent out of the ATE at the second unicast step. Therefore, test cost for a test session is mainly determined by the number of test packets.

ALGORITHM 1: *test-schedule-router*()

Input:
The NOC with link failures;
Output:
The identified faulty routers;
1: Set the router v connected to the ATE as the IFFR, and the internal router r that is connected to v for the first test session.
2: Deliver back all test responses compacted at the router r to the ATE.
3: The ATE sends the signal to the IFFR v of r for its state. If r is fault-free, put r into Q; otherwise, the ATE must move to another boundary router until a fault-free r has been found.
4: **while** $Q \neq \emptyset$ **do**
5: **for** each $r \in Q$ **do**
6: **if** r_1 is unprocessed internal router and directly connected to r **then**
7: put r_1 into Q_1;
8: remove r from Q if it has no directly connected neighbor that is unprocessed.
9: **end if**
10: **end for**
11: Find an IFFR for each router $r_1 \in Q_1$, assume the IFFRs are put in R. Let v be the router connected to the ATE. For each test packet of routers in Q_1, call *mcast-router* (v, R) to deliver the test packet to all routers in Q_1.
12: **for** each router $r_1 \in Q_1$ **do**
13: The test responses compacted at the MISR are delivered to the ATE, and the ATE sends a flag to its IFFR.
14: **end for**
15: Move the router $r \in R$ to Q if r is identified fault-free.
16: **end while**
17: Let B and B_1 be the sets of boundary routers and IFFRs, respectively. Call *mcast-router*(v, B_1) for each test packet of the boundary routers.
18: **for** each router $r \in B$ **do**
19: The test responses at the MISR are sent back to the ATE, and the ATE sends a flag to its IFFR.
20: **end for**
21: Let C and C_1 be corner routers and the IFFRs, respectively. Call *mcast-router*(v, C_1).
22: **for** each router $r \in C$ **do**
23: The test responses at the MISR are sent back to the ATE, and the ATE sends a flag to its IFFR.
24: **end for**

Test packets are delivered from the ATE to all routers tested in the same test session along the given multicast tree. However, the paths to deliver test packets in each unicast step can be different, because the baseline routing algorithm is adaptive. Internal router testing may include multiple test sessions. The number of failures, and distribution of failures can have an impact on the number of test sessions.

As presented in Algorithm 2, the multicast tree is kept in the header flits at the ATE as the unicast-based multicast algorithm, which is implemented by just forwarding the DOC to the root of the multicast tree first. Each router, which receives the test packet in the previous unicast step, determines its successors. A subset of destinations ordered into a DOC is forwarded to the corresponding successors. The process continues until all destinations have received the test packet.

The method in Xiang (2016b) applies tests inside a router and delivers tests sequentially. The tests are applied to the router under test when its consumption channel has been filled. Test delivery is started again until all tests have been delivered to the router as presented in Xiang

ALGORITHM 2: *mcast-router*(v,D)

Input:
The DOC with all IFFRs and the source v;
Output:
Receipt confirmation of the packet from all routers;
if $|D| = 2$ **then**
  deliver the packet to the remaining node d by calling *fault-tolerant-router*(v, d);
end if
Else, divide D into subsets D_1 and D_2 with $0 \leq |D_1| - |D_2| \leq 1$.
if c is in the lower half D_1 **then**
  deliver the test packet from v to the first node v_1 in the upper half D_2 with *fault-tolerant-router*(v, v_1);
  call *mcast-router*(v_1, D_2) at v_1, call *mcast-router*(v, D_1) at v.
end if
if v is in the upper half D_2 **then**
  deliver the packet from v to the last node v_2 in the lower half D_1 with *fault-tolerant-router*(v, v_2);
  call *mcast-router*(v_2, D_1) at v_2, call *mcast-router*(v, D_2) at v.
end if
Concurrent delivery of the test packet from all IFFRs in D to the routers under test.

(2016b). Assume that N identical routers are tested concurrently in the test session. The proposed method requires $\log_2 N + 2$ unicast steps to complete the multicast operation for each test packet.

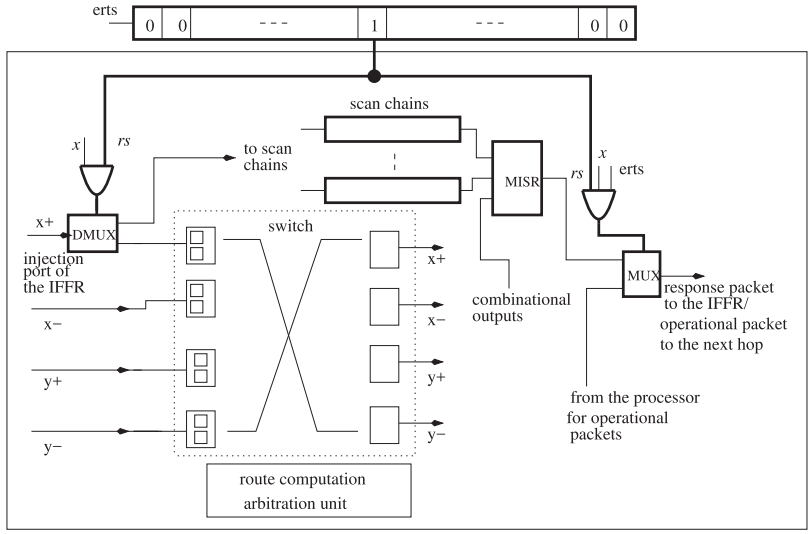
Our method does not use any of the routers under test as an intermediate node to avoid corrupting the test packet. Therefore, this may increase the number of unicast steps for test delivery, because a fault-tolerant unicast-based multicast algorithm must be used. However, this requirement provides concurrent test delivery and test application for router testing, which is a very attractive feature. A pipeline can be established to deliver test packets for router testing. The test cost is only determined by the number of test packets, and the number of unicast steps required to deliver each packet does not have an impact on the test cost. This can significantly reduce test cost compared to the method in Xiang (2016b).

The fault-tolerant routing algorithm provided by *fault-tolerant-router*(v_1, v_2) delivers a test packet from v_1 to v_2, which uses only the IFFRs and fault-free links identified by the previous test sessions instead of the whole network.

3.3 DFT Architecture to Implement Router Testing

Figure 2 presents the design for testability (DFT) architecture for router testing. The bold-faced components are inserted for router testing. A demultiplexer (DMUX) is inserted into one of the four input ports. The selection signal of the demultiplexer is the output of the AND gate with inputs x and rs. The NOC is under test when $x = 1$, otherwise, $x = 0$. When the test session for the router comes, $rs = 1$; otherwise, $rs = 0$. All router test session control signals are connected to an extra register. The multiplexer connected to the output of the MISR is selected by $sel1$, the output of the 3-input AND gate. The signal $erts$ (end of router test session) is set to 1 when a router test session is finished. In all other cases, $erts = 0$. The extra register is connected to all routers.

One of the two outputs of a DMUX is connected to the input port. The other output of the DMUX is connected to the multiplexer (MUX), whose output is connected to scan chains. The output of the MUX can be up to the width (d) of the physical channels, which drives up to d scan chains. The scan-out signals are connected to the MISR for test response compaction. The outputs from other combinational logic are also connected to the MISR. The output of the MISR is connected to another MUX, whose output is the injection port of the IFFR for the router under test; the



erts: end of a router test session; rs: router test session selection signal

Fig. 2. DFT architecture to implement test delivery and test application for router testing.

other input of the MUX is the channel from the local processor for operational packets, which are delivered to the next hop.

A multiple-input signature register (MISR) is inserted into each router to compact test responses. Synchronization for each unicast step is not necessary unlike the methods in Cota (2004) and Xiang (2011), because our method only delivers the final compacted responses back to the ATE. Test responses in the MISRs for routers tested in the same session are sent back to the ATE separately after a router test session has finished. Therefore, we assume that router failures for the routers under test can be identified after the end of a test session.

3.4 A Case Study

Let us consider the 8×8 mesh with five link failures as shown in Figure 3. We assume that the router c that is connected to the ATE is fault-free. The ATE delivers tests for the degree-4 router at node (3,1) in the first test session as shown in Figure 3(a), which is identified as being fault-free.

Two degree-4 routers at node (2,1) and (4,1) are tested in the second test session as shown in Figure 3(b). There is only a single IFFR (3,1) as shown in Figure 3(b) in the second test session, and the router c is another IFFR. A test packet is delivered to the router (3,1) in the first unicast step, which is delivered to the router (2,1) in the second unicast step from the router (3,1). The test packet is concurrently delivered to the router (4,1) from the router c in the second unicast step.

As shown in Figure 3(c), the new method selects (2,1) (v_1'), (3,1) (v_2'), (4,1) (v_3'), and c as IFFRs. Two unicast steps is required to deliver a test packet to the IFFRs, which is forwarded from v_1' to v_1, v_2' to v_2, v_3' to v_3 , and c to v_4 concurrently in the third unicast step.

Figure 3(d) presents the DOC to deliver test packets five for all six degree-4 routers at nodes (1,2) (v_1), (2,3) (v_2), (3,2) (v_3), (4,3) (v_4), (5,2) (v_5), and (6,1) (v_6). Six IFFRs at nodes (1,1) (v_1'), (2,2) (v_2'), (3,1) (v_3'), (4,1) (v_4'), (4,2) (v_5'), and (5,1) (v_6') are selected to deliver test packets. The selected six IFFRs are tested in the previous test sessions. Figure 3(e)–(h) presents the DOCs for test sessions 5–10 for internal router testing.

Figure 3(i) presents test delivery for the four corner (degree-2) routers. Four IFFRs, (1,0), (1,7), (6,0), and (6,7), are selected. A test packet at the ATE is delivered to v_2' in the first unicast step,

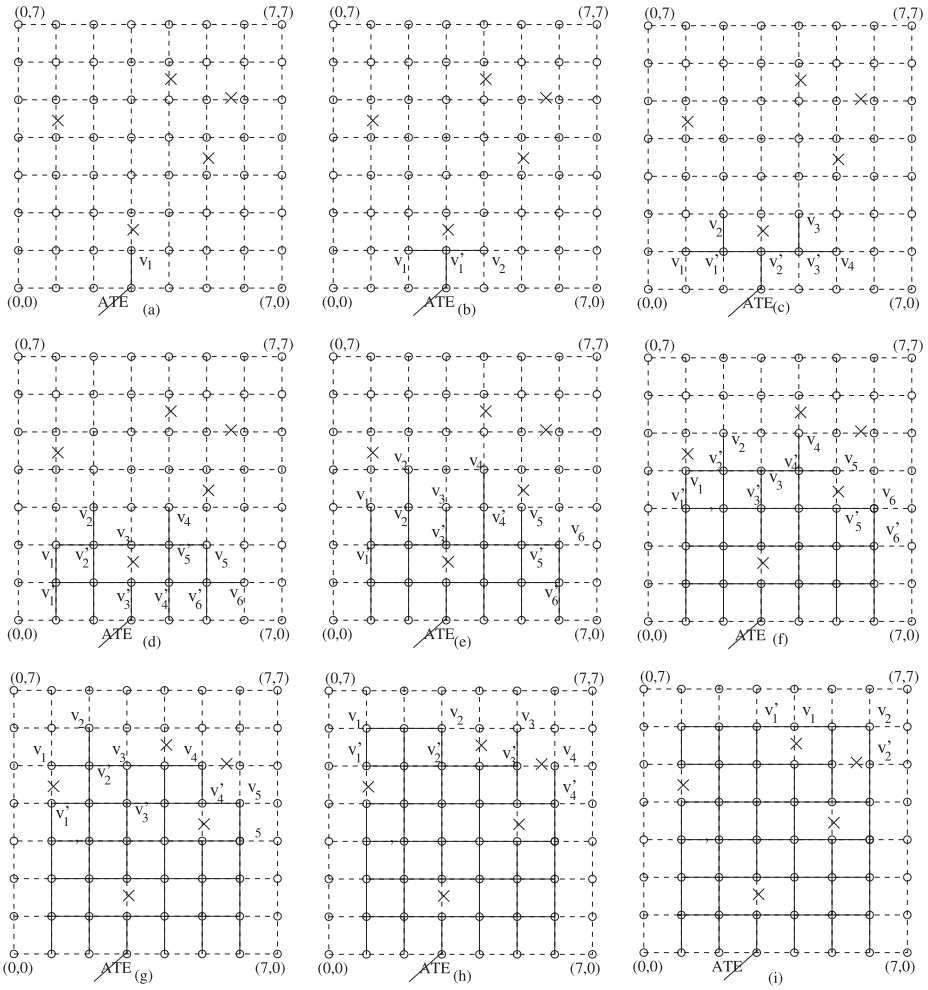


Fig. 3. The unicast-based fault-tolerant multicast scheme: ((a)–(i)) the first to ninth test sessions for internal router testing.

which is forwarded to v_3' from the ATE and from v_2' to v_1' in the second unicast step. The test packets at v_3' is forwarded to v_4' in the third unicast step. The test packets at v_1'-v_4' are delivered to nodes v_1-v_4, respectively, in the fourth unicast step.

Figure 4(a) presents the 10th test session. All boundary (degree-3) routers are tested concurrently in the test session. Totally, 23 degree-3 routers v_1-v_{23} are tested in the test session. Our method selects twenty-three IFFRs v_1'-v_{23}' to deliver test packets for the routers under test in this test session with seven unicast steps as shown in Figure 4(b). The test cost for the 10th test session is determined by the number of test packet instead of the number of unicast steps (Xiang 2011, 2016a). Figure 4(c) presents the general test scheduling for router testing.

Figure 5(a) presents the multicast tree of the 5th test session for internal router testing, and Figure 5(b) presents the multicast tree in the 10th test session for boundary router testing. As mentioned earlier, the number of unicast steps for a test session does not have great impact on the test cost. The routers under test in the 10th test session are omitted in Figure 5(b). Router testing cost is determined by the number of test sessions for internal router testing.

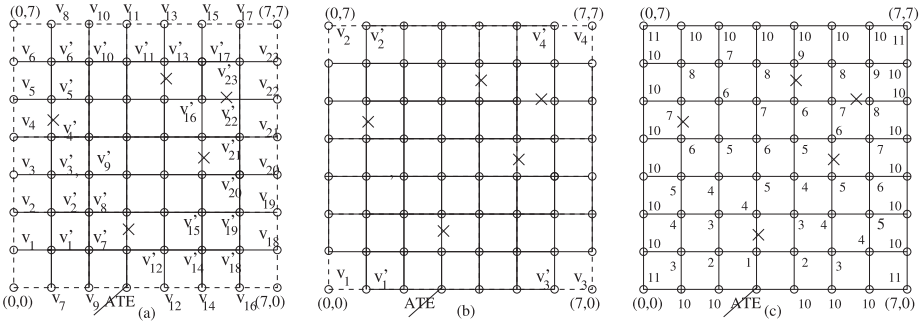


Fig. 4. The unicast-based fault-tolerant multicast scheme: (a) the 10th test session for boundary routers, (b) the 11th test session for corner router testing, and (c) the whole test scheduling for router testing.

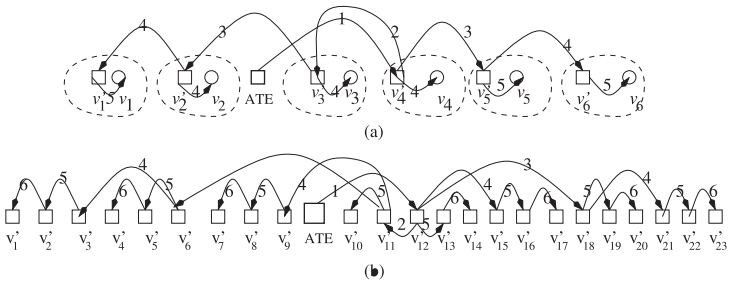


Fig. 5. Multicast trees: (a) the 5th test session, (b) the 10th test session.

Figure 6 presents the details for fault-tolerant multicast test delivery in the sixth test session. Four unicast steps are required to deliver a test packet to all six routers under test. As shown in Figure 6(a), a test packet is delivered from the ATE to v_4' in the first unicast step, which is forwarded from v_4' to v_3' in the second unicast step as presented in Figure 6(b). In the third unicast step, the test packet is delivered from v_2' to v_1', v_3' to v_3, v_4' to v_4, and v_5' to v_6' concurrently as shown in Figure 6(c). As shown in Figure 6(d), the test packet is forwarded from v_1' to v_1, v_2' to v_2, v_5' to v_5, and v_6' to v_6 concurrently in the fourth unicast step.

As mentioned earlier, test delivery for router test packets can only use bandwidths of the identified fault-free links and IFFRs (solid-link-connected-components). As shown in Figure 6(b) in the third unicast step of the sixth test session, the test packet is sent from v_3' to v_2', or v_4' to v_5'. The packets cannot be directly delivered from v_2' to v_3, or v_4' to v_5, because links v_2'-v_3 and v_4'-v_5 are not identified fault-free links to avoid corrupting the test packet. Similar situations occur in the fourth unicast step as shown in Figure 6(c) when the test packet is delivered from v_2' to v_1', and v_5' to v_6'.

4 FAULT-TOLERANT UNICAST-BASED MULTICAST FOR CORE TESTING

All identical cores share the same test set as in the router-testing method presented in Section 3. The core testing problem is formulated as a fault-tolerant unicast-based multicast problem for the NOC with some router and link failures after the router testing process. The main difference between the new method and the one proposed in Xiang (2011, 2016a) is the baseline routing scheme. The method in Xiang (2011, 2016a) used a dimension-order routing scheme, which did not consider the impact of faulty components. The faulty routers/links can corrupt the test data when delivering a packet across the faulty components.

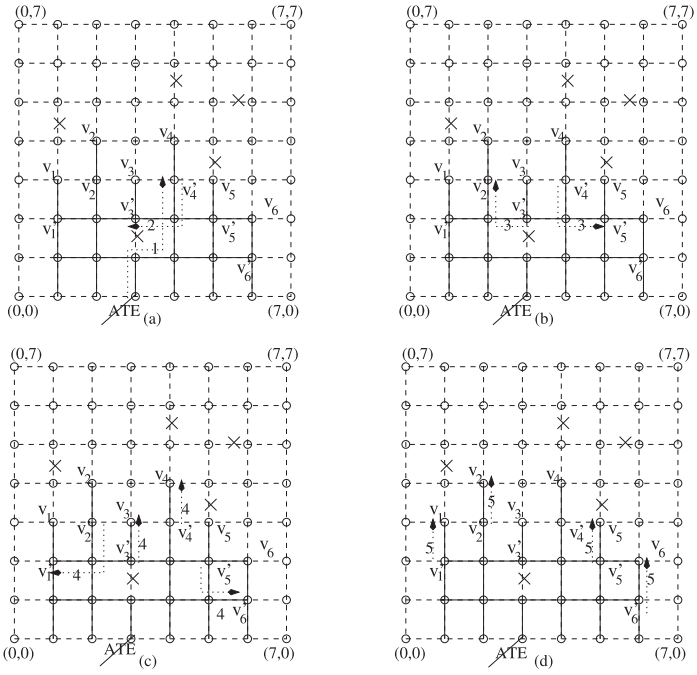


Fig. 6. The unicast-based fault-tolerant multicast test delivery in the fifth test session for router testing: (a) the first and second unicast steps, (b) the third unicast step, (c) the fourth unicast step, and (d) the fifth unicast step.

ALGORITHM 3: *core-testing*()

Input:
The NOC with injected link and router failures;
Output:
States of all cores;
1: Partition cores into classes $C = \{C_1, C_2, \ldots, C_k\}$, where cores in the same class are identical.
2: **while** $C \neq \emptyset$ **do**
3: Select a core class $C_i \in C$, $C \leftarrow C - C_i$; sort the cores detected by v in the NOC into a DOC D.
4: Call *deliver*(v', D) to deliver each of the test packets from the router v' connected to the ATE to all cores in the DOC D.
5: Deliver test responses at the MISR in each core in C_i back to the ATE.
6: **end while**

The whole test process is still partitioned into multiple test sessions, where all test packets of the core class are delivered to all cores in the class and applied to the cores. A novel fault-tolerant routing algorithm is provided with a new virtual network scheme.

The proposed fault-tolerant multicast algorithm is different from the one for router testing in Section 3. The fault-tolerant multicast algorithm unlike the fault-tolerant multicast algorithm in Section 3, which also does not constrain test delivery inside the sub-network with tested components. The new fault-tolerant unicast-based multicast algorithm for core testing utilize the full bandwidths of the network to deliver test packets.

Test delivery in the faulty network and test application inside cores under test can proceed concurrently unlike the router testing in Xiang (2016b). Test responses are compacted by an

ALGORITHM 4: *deliver*(v',D)

Input:
The NOC with injected failures, and the source v'.
Output:
1: **if** $|D| = 2$ **then**
2: deliver the packet to the remaining node d by calling *fault-tolerant-route*(v', d); exit.
3: **end if**
4: Divide D into two equal subsets D_1 and D_2.
5: **if** v' is in the lower half D_1 **then**
5: deliver the test packet from v' to the first node v_2 in the upper half D_2 with *fault-tolerant-route*(v', v_2);
5: call *deliver*(v_2, D_2) at v_2, call *deliver*(v', D_1) at v'.
6: **end if**
7: **if** v' is in the upper half D_2 **then**
8: deliver the packet from v' to the last node v_1 in the lower half D_1 with *fault-tolerant-route*(v', v_1);
9: call *deliver*(v_1, D_1) at v_1, call *deliver*(v', D_2) at v'.
10: **end if**

unknown-tolerant MISR (Xiang 2016a, 2013b). The final compacted test responses are delivered back to the ATE by a single packet after all the tests have been applied.

Unlike the method for router testing described in Section 3, and in Figure 2, test packets for core testing are delivered to the consumption channel as shown in Figure 8. Unlike in the router testing method, it is not necessary to select an IFFR for each core under test unlike the router testing method. Therefore, test packet delivery and test application can be completed concurrently as in Xiang (2016a, 2011).

Consider a multicast operation in a 2D mesh. As shown in Algorithm 3, all cores fall into multiple classes $C_1, C_2, \ldots, C_k$. Our method tests cores class by class until all cores have been handled. All identical cores are ordered into a DOC $c_1, c_2, \ldots, c_l$. A recursive fault-tolerant multicast scheme in Algorithm 3 is proposed for test delivery. All test packets for the same class of cores are delivered consecutively in a single test session.

The fault-tolerant unicast-based multicast algorithm is presented in Algorithms 3 and 4. A test packet is delivered to a node v' from the node connected to the ATE. The cores in the same class are arranged into a DOC D. The core sequence is divided into two equal parts D_1 and D_2. Let v' be in the lower part D_1; it delivers the test packet to the first node c_2 in the upper half D_2, where c_2 will be responsible for the test packet delivery to all other cores in D_2 using the same procedure recursively, and v' manages test packet delivery of the first part. If the core v' is in the upper part D_2, then it sends the packet to the last node c_1 in the first part D_1. The core c_1 manages test packet delivery for D_1, and the core v' handles test packet delivery of the second part recursively. Our method needs $\log_2 N + 1$ unicast steps to deliver a test packet to all destinations, where N is the number of destinations related to the multicast operation.

In contrast to McKinley (1994), the fault-tolerant unicast-based multicast can be completed with multiple unicast steps. Therefore, the ATE can deliver the next test packet in the second unicast step. The procedure *fault-tolerant-route*(v', d) delivers a packet from v' to d using the baseline fault-tolerant routing algorithm.

The multicast tree is kept in the header flits, and it is implemented by simply forwarding the DOC to the root of the multicast tree. Each core, which receives the test packet in the previous unicast step, determines its successors by running the procedure presented in Algorithms 3 and 4. A subset of destinations ordered into a DOC is forwarded to the corresponding successors. The process continues until all destinations have received the test packet.

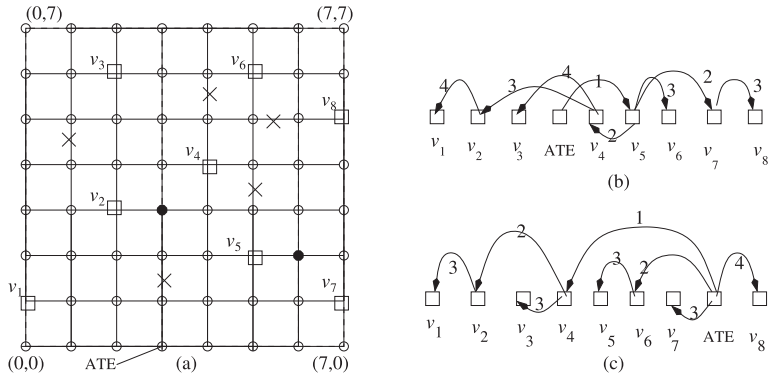


Fig. 7. Fault-tolerant unicast-based multicast to deliver test data for cores: (a) cores under test, (b) the multicast tree with the ATE at (3,0), and (c) the multicast tree with the ATE at (7,3).

Let us consider the 8×8 mesh shown in Figure 7(a), which contains five link and two router failures. The address sequence (0,1) (v_1), (2,3) (v_2), (2,6) (v_3), (4,4) (v_4), (5,2) (v_5), (5,6) (v_7), (7,1) (v_6), and (7,5) (v_8) is a DOC. Figure 7(b) presents the multicast tree used to deliver a test packet from the ATE to all destinations in four unicast steps. The numbers at the arrowed lines present the unicast steps. Unlike the method in Xiang (2011), we assume that the NOC is designed based on the baseline fault-tolerant routing algorithm. Therefore, a test packet is delivered to the destinations based on the new fault-tolerant routing scheme instead of dimension-order routing. The baseline fault-tolerant routing algorithm can be replaced by any other fault-tolerant routing algorithms.

The node that receives the test packet from the ATE must forward the test packet to multiple nodes in the following consecutive unicast steps. That node must keep the test packet in the consumption buffer in all the unicast steps. As shown in Figure 7(b), a bottleneck can occur at node v_5, because the node v_1 as shown in Figure 7(b) must forward the test packet to other destinations. The later packets may have to wait until the consumption buffer at v_5 has been released. Therefore, test packet delivery is delayed. Figure 7(c) presents the multicast tree when connecting the ATE to router (7,3).

The number of extra pins must be kept small. Each core requires a number of scan-in and scan-out pins, which can make the total number of extra pins large when the number of cores in the NOC is large. The number of scan enable pins of the scan flip-flops for all cores can also be unacceptable only if each core uses a separate scan enable pin. Cores of the same class can be tested concurrently. We show below why the proposed method is efficient in terms of pin overhead.

All the scan-in pins of a core are driven by the consumption buffer, therefore, no extra scan-in pins are necessary. All the scan-out pins are connected to the MISR, where the output of the MISR is connected to the injection buffer. The injection buffer is the interface between the local processor and the network, which injects packets from the local processor to the network for the next unicast step or delivers the final response packet to the ATE. In our method, all cores share the same test selection pin as shown in Figure 8. Therefore, the number of extra pins to control scan testing in an NOC is just one.

Our method needs a total of k extra pin as shown in Figure 8, where each extra pin drives a class of cores and k is the number of different classes of cores (k is set to be no more than 4 for all experimental results in this article. It can be any number). If control signal $c_i = 1$ ($i \in \{0, 1, \ldots, k-1\}$), then the ith class of cores is under test. The number of extra pins can be reduced by using an extra register.

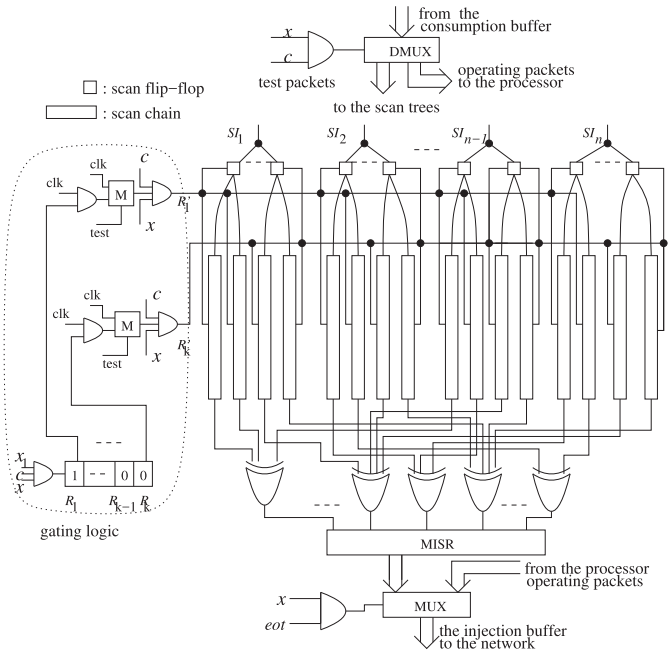


Fig. 8. DFT architecture for core testing.

An extra pin x is set to logic value one in test mode and is set to zero during the operational mode. It is the same signal for router testing as presented in Figure 2. The signal "test" has the same value as the select-pin of the multiplexers that are inserted into the scan flip-flops. It is set to one in scan-shift cycles, and to zero for functional cycles. The scan chains are selected by the content of the extra register while shifting-in a test vector. All scan flip-flops are driven by the normal clock signal *clk* during functional cycles. The register is shifted by one bit after d shift cycles, where d is the maximum depth of the scan chains. The signal x_1 is set to zero during the shift cycles and to one when shifting the register to the next state. The test vector is then shifted into the next subset of scan trees.

Routers are under test when $x = 1$, $x_1 = 0$, $c_i = 0$ for $i \in \{0, 1, \ldots, k-1\}$. Test responses of routers are delivered from the MISRs to the ATE after each test session. The compacted test responses of all MISRs at cores are delivered back to the ATE after all test sessions have been completed. The NOC is set to the operational model when $x = 0$, $x_1 = 0$, and $c_i = 0$ for $i \in \{0, 1, \ldots, k-1\}$.

In total, our method requires five extra pins: x to indicate the testing process, x_1 for low-power core testing, an extra pin for all core class selection, *erts* to indicate the end of a router test session, and a *eot* to show the end of core and router testing. It is required to distinguish between router and core testing, and control the gating logic for low-power core testing using the extra pins.

5 BASELINE FAULT-TOLERANT ROUTING ALGORITHM

In this section, we present details of an adaptive deadlock-free fault-tolerant routing algorithm for 2D-mesh-based NOCs. The proposed algorithm is based on virtual cut-through (VCT) switched NOCs, hence it is used for fault-free NOCs. Each input port is assigned two buffers, where each buffer is assigned to a separate virtual channel (VC). Each VC is large enough to keep the whole packet.

A 2D mesh is partitioned into eight virtual networks: $x{+}y{+}e{+}$ (1), $x{+}y{+}e{-}$ (2), $x{+}y{-}e{+}$ (3), $x{+}y{-}e{-}$ (4), $x{-}y{+}e{+}$ (5), $x{-}y{+}e{-}$ (6), $x{-}y{-}e{+}$ (7), and $x{-}y{-}e{-}$ (8). Here e is an extra dimension, which does not really exist.

The extra dimension is not meaningful for the deadlock-free adaptive fault-tolerant routing algorithm in 2D meshes. We can merge the eight virtual networks as follows: (1) $x^*y{-}$ (4,8), (2) $x^*y{+}$ (2,6), (3) $x{-}y^*$ (5,7), (4) $x{+}y^*$ (1,3). We have the following virtual channel assignment, $x^*y{-}$ ($c_{x,1}$, $c_{y,1}{-}$), $x^*y{+}$ ($c_{x,1}$, $c_{y,1}{+}$), $x{-}y^*$ ($c_{x,2}$, $c_{y,2}$), and $x + y^*$ ($c_{x,2}{+}$, $c_{y,2}$) for x and y channels in the four merged virtual networks.

Four virtual networks are grouped into two separate subsets. The first two virtual networks fall into the first class, and the last two fall into the second class. Packets can be delivered in the same subset of virtual networks to avoid faulty nodes or links.

Our method sets two prevented turns. One prevented is assigned in the first virtual network subset, and the other is assigned to the second subset.

Our method also arranges two special turns. Any packet with the special turn enters into the other subset of virtual networks, which turns back after traversing the special turn channels along the same direction.

It is also important to ensure fault tolerance in a 2D mesh if the shortest paths are blocked. Packets in the first subset of virtual networks can be delivered from one x channel to the other virtual network in the same subset. Any packet in the second subset of virtual networks can be delivered from a y channel in one virtual network to the original one. Any packet misrouted to the other virtual network can return back to the other virtual network in the same subset.

6 EXPERIMENTAL RESULTS

We have implemented the proposed method, and the methods in Xiang (2011) and Grecu (2007). Single stuck-at faults are considered in routers and cores in this article. Testing of the NOC for transition faults can be carried out in a similar manner. We assume that test application is started immediately after a test packet has been received at each core. As in previous work, at most three packets can be kept at the injection buffer and the consumption buffer at each router. The consumption buffer is the interface from the network to the processor, and the injection buffer is the interface from the processor to the network.

The start-up and receipt latency are set to 10 clock cycles in all the simulations (Xiang 2011). Two adjacent routers transfer a single flit data in each clock cycle, where a flit contains 32 bit data. The IWLS2005 circuits ethernet, vga_lcd, and netcard and the largest ITC99 circuit b19 are randomly assigned to the NOC as cores. Table 1 shows the statistics of the cores. Circuits b19, netcard, ethernet and vga are randomly assigned as cores when the NOC contains four classes of cores. Circuits netcard, b19 and vga are randomly assigned to the NOC when it contains three different classes of cores; vga and netcard are randomly assigned to the cores when the NOC contains two classes of cores; the circuit netcard is assigned to all cores when the NOC contains a single class of cores.

Column "AO" in Table 1 presents the area overhead (percentage) of the DFT architecture in a core compared to area overheads of Xiang (2016a) in all classes of cores. The results show that the new method requires slightly less area overheads than previous methods. The DFT architecture to tolerate unknown responses and the decompressor for selective encoding can contribute to this. Table 1 presents the fault coverage (FC), test pattern count (vec) and CPU time (in seconds) to generate tests for single stuck-at faults and transition faults. Our method also obtains complete test coverage for the synthesized routers. The proposed new core testing scheme is superior to the previous one-to-one unicast method in Cota (2004). Extensive performance comparison with the one-to-one unicast core testing technique was presented in Xiang (2011).

Table 1. Statistics of the Cores and ATPG Data

	statistics for cores				stuck-at tests			transition tests		
circuits	FFs	gates	AO	AO (Xiang 2016a)	vec	FC	CPU(s)	vec	FC	CPU(s)
b19	6,642	225,800	1.87	1:2.40	1560	98.80	4880	9820	84.47	11246
ethernet	13,715	105,371	2.45	-	910	99.20	1580	3710	99.39	7616
vga_lcd	17,079	153,664	1.96	3:4.26	980	99.40	3240	8142	99.64	14110
netcard	97,796	568,986	2.53	4:4.24	2260	99.10	4680	27810	97.52	35220

Table 2. Statistics of the Synthesized Routers

	router statistics				
routers	FFs	area	gates	AO	AO (Xiang 2016b)
2-ports	1324	187,738	18,229	6.64	-
3-ports	1927	226,041	22,651	6.55	10.39
4-ports	2458	275,413	27,651	6.51	9.53

Table 3. Fault Injection

network	Case 1	Case 2
6x6	1 router, 2 links	2 routers, 4 links
8x8	1 router, 3 links	2 routers, 4 links
16x16	1 router, 4 links	2 routers, 6 links

Table 4. Router Test Cost in Separate NOCs Measured in Clock Cycles

NOCs	(Grecu 2007)	no fault	Case 1	Case 2	(Xiang 2016b)
6×6	2,966,529	439,314	478,473	554,301	2,644,510
8×8	4,009,340	547,591	560,191	591,175	2,644,510
16×16	8,208,375	900,458	913,825	946,073	2,644,510

Table 2 presents the statistics for the synthesized routers with different numbers of input ports. Column AO in Table 2 presents the area overhead of the proposed fault-tolerant routing scheme and the DFT logic at the routers compared with Xiang (2016b). It is found that area overheads for routers (Xiang 2016b) with the same number of ports is higher than that of the proposed method. This result can be attributed to the deadlock avoidance mechanism for consumption channels (Xiang 2016b). Table 3 presents the fault injection in different networks.

In the Table 4, column "no fault" represents the case that the NOC under test is fault-free. A given number of link and router faults are randomly injected into the NOC as shown in Table 2 for two cases (Case 1 and Case 2). For example, one router and four link faults in Case one, two router and four link faults in Case 2, are injected into the 6×6 mesh.

Table 4 presents router test cost comparison with Grecu (2007) and Xiang (2016b) in different NOCs. Router test cost for the methods in Grecu (2007) and the proposed new method increases significantly when the size of the NOC increases. The size of the NOC does not have significant impact on the router test cost for the method in Xiang (2016b), therefore, the router test cost of $6 \times 6 \times 6$ is presented in Table 3 for this method. The reason why router test cost increases significantly when the size of the NOC increases is that the number of internal router test sessions

Table 5. Core Test Cost in a 6×6 NOC Measured in Clock Cycles

core class	(Xiang 2011)	Case 1	Case 2	(Xiang 2016a)
1	667,903	692,732	703,245	221,119
2	752,358	764,975	782,351	-
3	926,386	950,234	971,028	293,446
4	1,004,728	1,038,392	1,072,849	450,456

Table 6. Core Test in an 8×8 NOC Measured in Clock Cycles

class	(Xiang 2011)	Case 1	Case 2	(Xiang 2016a)
1	680,912	693,857	714,778	221,119
2	767,167	780,532	798,172	-
3	945,139	957,448	975,317	293,446
4	1,028,387	1,067,358	1,088,421	450,456

Table 7. Core Test cost in a 16×16 NOC Maesured in Clock Cycles

class	(Xiang 2011)	Case 1	Case 2	(Xiang 2016a)
1	736,149	764,720	785,080	221,119
2	809,824	853,280	878,453	-
3	989,854	1,011,452	1,026,031	293,446
4	1,074,258	1,092,421	1,148,236	450,456

increases significantly. The faulty components do not lead to a noticeable increase in the router test cost for all sizes of NOCs.

The router test cost for the proposed method does not increase significantly when router and link failures are injected. This is mainly because the number of test sessions does not significantly increase. For example, the number of test sessions for internal router testing is still 9 in the fault-free 8×8 NOC, which is the same as that in the faulty NOC as shown in Figure 3. Therefore, the difference between test delivery cost in fault-free and faulty NOC reflects the latency difference introduced by faulty components.

Tables 5, 6, and 7 present core test cost comparison in 6×6, 8×8, and 16×16 NOCs, respectively. Table 5 presents test time for core testing in 6×6 meshes. The column "core test time" presents a comparison of the proposed method with Xiang (2011).

As shown in Table 5, the test time for core testing in both Case 1 and Case 2 is a little higher than that in a fault-free NOC based on the method in Xiang (2011) because of the new fault-tolerant routing scheme. The test time increases slightly when the number of link and router failures increases. That is, router and and link failures do not have a significant impact on the test-delivery time for core testing.

Router test time for the new method is much less than that in Grecu (2007), even though (Grecu 2007) provided hardware support for multicast. The most important reasons are as follows: (1) the new method provides a scheme in which test delivery and test application can proceed concurrently, but the method in Grecu (2007) handles test delivery and test application sequentially; (2) the scan-tree architecture is used in our work, while the results in Grecu (2007) are obtained using scan chains with the same number of scan-in pins.

Table 6 presents experimental results for faulty 8×8 meshes when three link and one router failures are randomly injected in Case 1, and four link and two router failures are randomly

injected in Case 2. The proposed router testing method can reduce test cost significantly compared to Grecu (2007). Increase in the network size leads to significant router test-cost increase the method in Grecu (2007) and the new method. However, the core testing time based on the new method remains almost the same.

Table 7 presents results for 16×16 meshes. The router testing time for the previous method (Grecu 2007) is almost doubled compared to that for the 8×8 network. The core testing time for the proposed method increases by small amount. In Tables 5, 6, and 7, we show test cost for routers and core compared with Xiang (2016a, 2016b). The method in Xiang (2016a) was proposed for 3D stacked core testing, which did not consider the impact of link/router failures. Scan forest was combined with selective encoding (Wang and Chakrabarty 2008) to compress test data of cores, but the new article uses only scan forest to compress test data inside cores. Therefore, due to the use of different DFT architectures, test data and the test delivery time for the new method are more than that in Xiang (2016a).

The cores assigned to the 3D stacked NOCs are different from the ones in this article. In the NOCs with four core classes, three core classes, and a single core class, circuits (b19, des, ethernet, vga), (b19, des, vga), and (b19) are randomly assigned, respectively. In this article, des is replaced by netcard for the first two cases, (vga, netcard), (netcard) are randomly assigned for two core classes and a single core class. The above two reasons contribute to the difference in the core test cost between the new method and the one in Xiang (2016a).

The fault-tolerant multicast approach for core testing is not affected much by the number of link/router failures. However, the router testing approach is very sensitive to these failures, which may increases the number of internal router test sessions. As the size of an NOC increases, the new router testing approach becomes less sensitive to the number of failures. The router test cost for the new method is much less than that in Xiang (2016b), because concurrent test delivery and test application for router testing are utilized.

7 UTILIZING THE MULTIPLE PORTS OF ATES TO REDUCE TEST COST

The test cost for core and router testing can still be very high as presented in Sections 3 and 4. We extend the proposed router and core testing methods by utilizing the multiple ports of the ATE to increase the bandwidths available for test delivery. The number of test sesssions for internal router testing determines the router test cost. Separate ports can deliver test packets for different internal test sessions. The number of internal router test sessions can significantly be reduced using the multiple ports of the ATE.

The core testing scheme is scalable. We show that the core test cost does not increase significantly when the size of the network increases. Multiple ATE ports can also significantly reduce test cost for core testing. Our method considers ATEs with up to four input/output ports. The core testing cost can still be effectively reduced by using the multiple input/output ATE ports when the number of core classes is large enough.

Each pair of input/output port of the ATE is connected to each of the four boundaries in the NOC. Therefore, test packets can be delivered into the network in parallel to reduce the number of internal router test sessions. Algorithm 5 presents the test scheduling scheme for router testing with multiple-port ATE. All ports deliver the same set of test packets to the NOC concurrently. The algorithm *mport-test-schedule-router*() is applicable only for internal router test sessions. The boundary router and corner router test sessions must still be handled sequentially after all internal router test sessions. The procedure *mcast-router*() is the same as that in Algorithm 1 for router test with a single input/output port.

Our method assumes that at least one tested router in an internal router test session is fault-free. We also have this implicit assumption for the single input/output port router testing approach as

ALGORITHM 5: *mport-test-schedule-router*()

Input:
The NOC with link failures;
Output:
The identified faulty routers;

Let a router c_i be connected to the ith port for the ATE as the initial IFFRs, and the router r_i connected to c_i for the first test session of each port.
Deliver all test packets to r_i, and send back the compacted test responses at r_i to the ATE. The ATE sends the signal to the IFFRs to indicate the state of the router.
If any of the router r_i for the ith port is fault-free, put r_i into Q_i; if one router r_i is faulty, move the port to another boundary router until a fault-free c_i has been found, put c_i to Q_i.
while one of the queues $Q_i \neq \emptyset$ **do**
  for $i = 1$ to k **do**
    for each $r_i \in Q_i$ **do**
      if $r_{i,1}$ is directly connected to r_i and $r_{i,1}$ is an unprocessed internal router **then**
        put $r_{i,1}$ into $Q_{i,1}$, remove r_i from Q_i.
      end if
    end for
    Find an IFFR for each router $r_{i,1} \in Q_{i,1}$, assume the IFFRs are put in R_i.
    Let c_i be the router connected to the ith port of the ATE. For each test packet of routers in $Q_{i,1}$, call *mcast-router* (c_i, R_i) to deliver the test packet to all routers in $Q_{i,1}$.
    for each router in $r_i \in Q_{i,1}$ **do**
      The test responses at the MISR are delivered back to the ATE, and the ATE sends a flag to its IFFR for the state.
    end for
    Put all identified fault-free routers in $Q_{i,1}$ to Q_i.
  end for
end while
Let B be the set of boundary routers. Find IFFRs B_1 for the boundary routers in B. Call mcast-router(c,B_1) for each of the test packets of the boundary routers.
for each router $r \in B$ **do**
  The test responses at the MISR are sent back to the ATE, and the ATE sends a flag to the IFFRs.
end for
Let C and C_1 be the corner routers and IFFRs, respectively. Call *mcast-router*(c, C_1) for each of the test packets.
for each router $r \in C$ **do**
  The test responses at the MISR are sent back to the ATE, and the ATE sends a flag to the IFFRs.
end for

presented in Section 3. This implies that each input/output port can continually deliver test data into the network. Separate ports deliver the same set of test packets for the data flits, but different header flits. The reason is that different ports may be involved in different sets of destinations for the same test packet. The tested internal router regions increase from up to four ports until all regions have been connected.

Figure 9(a) illustrates test scheduling with a single port. Figure 9(b) presents test scheduling for an ATE with two ports. The number of internal router test sessions is reduced from 9 to 6 as presented in Figure 9(b). The number of internal router test sessions is reduced from 6 to 4 as shown in Figure 9(c) for ATEs with four ports.

The router test cost can apparently be reduced using multiple port ATEs. Multiple ports of an ATE can improve the performance of the core testing process. The main idea is to deliver

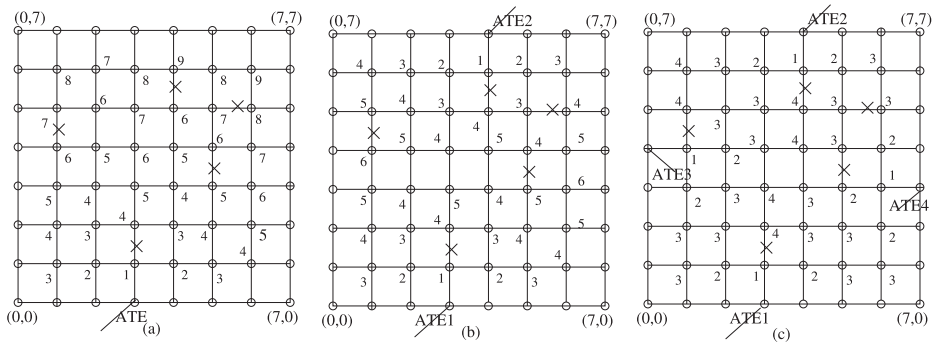


Fig. 9. Multiple ports of the ATE to reduce test cost: (a) single port, (b) two ports, (c) four ports.

test packets for different classes of cores from separate ports of the ATE. The difference from the internal router test scheduling algorithm is that the delivery time for different classes of cores is different. Any port is assigned to a new class of cores as soon as test delivery for the previous class has been completed and the input/output port has been released. Separate ports deliver different test packets into the network in any case. That is, different test packets are delivered to the NOC for different classes of cores via different ports at the same time.

All core classes are ordered by the test data volumes to enhance the parallelism by scheduling bigger cores test earlier with multiple-port ATEs. Each core class is assigned a port and all cores can be tested concurrently if the number of core classes is no more than the number of ATE ports.

All cores must be tested concurrently when all cores are identical. This introduces low-power in the NOC because of the gating technique based low-power DFT architecture as presented in Figure 8, where only 10% logic is activated for each core during test application. Most power consumption of an NOC is introduced by test application for core testing, therefore, test power can be well-controlled.

The core classes of the most test data volumes are assigned to the ATE ports first when the number of core classes is greater than the number of ports of the ATE. The ATE port is assigned to the class of cores that have the most test data volume among the remaining core classes immediately after the ATE port has been released. This process continues until all cores have been handled. The test responses captured at the MISRs are delivered back to the ATE after all cores in the same class have been tested. Test responses at the MISR of each core are established into a single packet. The test response packet is delivered back to the ATE using the baseline fault-tolerant routing algorithm.

Our method can still work when the ATE cannot keep test data of all cores. In that case, our method enables fine granularity parallelism using multiple ports of the ATE. That is, different test packets of the same class of cores are concurrently delivered via multiple ports and different paths.

No path contention when different ports of the ATE deliver different test packets of the same class of cores, because different ATE ports deliver test packets along different multicast trees although the destinations are the same. Figure 7(b) and (c) shows the multicast trees when the ATE ports are placed at (3,0) and (7,3), respectively. After test packet delivery of the same class of cores has been completed, all ports of the ATE begin to deliver tests of another class of cores.

8 CONCLUSIONS

We have presented a unified approach to test a network-on-chip in entirety. The proposed scheme detects link failures, router faults, and faults in cores in a single flow. Routers with the same degree (i.e., the same number of routers that is connected to it) are deemed to be identical from a test

perspective. Test packets for the routers tested in the same test session are delivered in the faulty network via the IFFRs, whereby identical routers can share the same test packets.

The test packets of cores are delivered along fault-free routers and links after all faulty routers have been identified by using a new fault-tolerant multicast algorithm. The proposed fault-tolerant multicast algorithm does not require any changes to the router architecture. Hence, unlike in prior work, this method does not constrain test-packet delivery within the tested sub-network. New test scheduling algorithms have also been presented using ATEs with multiple ports, which further reduce test cost for router and core testing. Experimental results highlight the effectiveness of the proposed method in reducing test time. Small delay defect or transition fault testing for router/core of NOCs are further work of this article.

REFERENCES

M. Agrawal, M. Richter, and K. Chakrabarty. 2012. A dynamic programming solution for optimizing test delivery in multicore SOCs. In *Proceedings of the International Test Conference*, paper 1.3.

L. Benini, and G. DeMicheli. 2002. Networks on chips: A new SOC paradigm. *IEEE Comput.* 35, 1 (2002).

E. Cota, L. Carro, and M. Lubaszewski. 2004. Reusing an on-chip network for the test of core-based systems. *ACM Trans. Des. Autom. Electr. Syst.* 9, 4 (2004), 471–499.

E. Cota, and C. Liu. 2006. Constraint-driven test scheduling for NOC-based systems. *IEEE Trans. Comput.-Aid. Des.* 25, 11 (2006), 2465–2478.

E. Cota, F. L. Kastensmidt, M. Cassel, M. Herve, P. Almeida, P. Meirelles, A. Amory, and M. Lubaszewski. 2008. A high-fault-coverage approach for the test of data, control, and handshake interconnects in mesh networks-on-chip. *IEEE Trans. Comput.* 57, 9 (2008), 1202–1215.

A. Dalirsani, S. Holst, M. Elm, and H. J. Wunderlich. 2011. Structural test for graceful degradation of NOC switches. In *Proceedings of the IEEE European Test Symposium.* 183–188.

W. J. Dally and B. Towles. 2001. Route packets not wires: On-chip interconnection networks. In *Proceedings of the IEEE/ACM Design Automation Conference.*

J. Dalmasso, E. Cota, M. L. Flottes, and B. Rouzeyre. 2008. Improving the test of NOC-based SOCs with help of compression schemes. In *Proceedings of the Annual Symposium on VLSI.* 139–144.

J. Fan. 1998. Diagnosability of the mobius cubes. *IEEE Trans. Parallel Distrib. Syst.* 9, 9 (1998), 923–928.

J. Fan. 2002. Diagnosability of crossed cubes under the comparison diagnosis model. *IEEE Trans. Parallel Distrib. Syst.* 13, 10 (2002), 1099–1104.

V. Froese, R. Ibers, and S. Hellebrand. 2008. Reusing NOC-infrastructure for test data compression. in *Proceedings of the IEEE 28th VLSI Test Symposium.* 227–231.

A. Ghiribaldi, D. Ludovici, F. Trivino, A. Strano, J. Flich, J. L. Anchez, F. Alfaro, M. Favalli, and D. Bertozzi. 2013. A complete self-testing and self-configuring NOC infrastructure for cost-effective MPSoCs. *ACM Trans. Embed. Comput. Syst.* 12, 4 (2013), Article 106.

A. Ghofrani, R. Parikh, A. Shamshiri, A. DeOrio, K. T. Cheng, and V. Bertacco. 2012. Comprehensive online defect diagnosis in on-chip networks. In *Proceedings of the IEEE 30th VLSI Test Symposium.* 44–49.

M. Gorgues, D. Xiang, Z. Yu, J. Flich, and J. Duato. 2014. Achieving balanced buffer utilization with a proper co-design of flow control and routing algorithm. In *Proceedings of the IEEE/ACM International Symposium on Networks-on-Chip.* 25–32.

C. Grecu, A. Ivanov, R. Saleh, and P. P. Pande. 2007. Testing network-on-chip communication fabrics. *IEEE Trans. Comput.-Aid. Des.* 26, 12 (2007), 2201–2214.

M. R. Kakoee, V. Bertacco, and L. Benini. 2011. A distributed and topology-agnostic approach for on-line NOC testing. In *Proceedings of the IEEE/ACM International Symposium on Networks-on-Chip.* 113–120.

M. R. Kakoee, V. Bertacco, and L. Benini. 2014. At-speed distributed functional testing to detect logic and delay faults in NOCs. *IEEE Trans. Comput.* 63, 3 (2014), 703–717.

L. M. Lin, L. Xu, S. Zhou, and S.-Y. Hsieh. 2016. The t/k-diagnosability for regular networks. *IEEE Trans. Comput.* 65, 10 (2016), 3157–3170.

L. M. Lin, L. Xu, S. Zhou, and D. Wang. 2018. The g-good-neighbor conditional diagnosability of arrangement graphs. *IEEE Trans. Depend. Secure Comput.* 15, 3 (2018), 542–548.

D. H. Linder and J. C. Harden. 1991. An adaptive and fault-tolerant wormhole routing strategy for k-ary n-cube. *IEEE Trans. Comput.* 40, 1 (1991), 2–12.

D. Kohler and M. Radetzki. 2006. Test scheduling with thermal optimization for network-on-chip systems using variable rate on-chip clocking. In *Proceedings of the Design, Automation and Test in Europe Conference (DATE'06).*

P. K. McKinley, H. Xu, A. H. Hossein, and L. M. Ni. 1994. Unicast-based multicast communication in wormhole-routed networks. *IEEE Trans. Parallel Distrib. Syst.* 5, 12 (1994), 1252–1265.

D. K. Panda, S. Singal, and R. Kesavan. 1999. Multidestination message passing in wormhole k-ary n-cube networks with base routing conformed paths. *IEEE Trans. Parallel Distrib. Syst.* 10, 1 (1999), 76–96.

R. Parikh and V. Bertacco. 2016. Resource conscious diagnosis and reconfiguration for NOC permanent faults. *IEEE Trans. Comput.* 65, 7 (2016), 2241–2256.

K. Peterson and K. Oberg. 2007. Toward a scalable test methodology for 2D-mesh networks-on-chip. In *Proceedings of the IEEE/ACM Design Automation, and Test in Europe Conference.*

M. Radetzki, C. Feng, X. Zhao, and A. Jansch. 2016. Methods for fault tolerance in networks-on-chip. *ACM Comput. Surv.* 46, 1 (2016), Article 8.

M. Richter and K. Chakrabarty. 2014. Optimization of test pin-count, test scheduling, and test access for NOC-based multicore SOCs. *IEEE Trans. Comput.* 63, 3 (2014), 691–702.

I. Seitanidis, A. Psarras, E. Kalligeros, C. Nicopoulos, and G. Dimitrakopoulos. 2014. ElastiNoC: A self-testable distributed VC-based network-on-chip architecture. In *Proceedings of the IEEE/ACM International Symposium on Networks-on-Chip.* 135–142.

S. Shamshiri, A. Ghofrani, and K. T. Cheng. 2014. End-to-end error correction and online diagnosis for on-chip networks. In *Proceedings of the IEEE International Test Conference*, Paper 10.3.

S. Shamshiri and K. T. Cheng. 2008. Modeling yield, cost, and quality of a spare-enhanced multicore chip. *IEEE Trans. Comput.* 60, 9 (2008), 1246–1258.

X. T. Tran, Y. Thonart, J. Durupt, V. Beroulle, and C. Robach. 2008. A design-for-test implementation of an asynchronous network-on-chip architecture and its associated test pattern generation and application. in *Proceedings of the IEEE/ACM International Symposium On Networks-on-Chip*, 149-158.

Z. Wang and K. Chakrabarty. 2008. Test Data Compression Using Selective Encoding of Scan Slices. *IEEE Trans. VLSI Syst.* 16, 11 (2008), 1429–1440.

D. Xiang and Y. Zhang. 2011. Cost-effective power-aware core testing in NOCs based on a new unicast-based multicast scheme. *IEEE Trans. Comput.-Aid. Des.* 30, 1 (2011), 135–147.

D. Xiang, K. Chakrabarty, and H. Fujiwara. 2016a. Multicast-based testing and thermal-aware test scheduling for 3D ICs with a stacked network-on-chip. *IEEE Trans. Comput.* 65, 9 (2016), 2767–2779.

D. Xiang. 2013a. A cost-effective scheme for network-on-chip router and interconnect testing. In *Proceedings of the IEEE Asian Test Symposium.* 207–212.

D. Xiang, G. Liu, K. Chakrabarty, and H. Fujiwara. 2013b. Thermal-aware test scheduling for NOC-based 3D integrated circuits. In *Proceedings of the 21th IFIP/IEEE International Conference on VLSI-SOC.* 99–104.

D. Xiang and K. Shen. 2016b. A new unicast-based multicast scheme for network-on-chip router and interconnect testing. *ACM Trans. Des. Autom. Electr. Syst.* 21, 2 (2016), Article 24.

D. Xiang, K. Chakrabarty, and H. Fujiwara. 2016c. A unified test and fault-tolerant multicast solution for network-on-chip designs. In *Proceedings of the IEEE International Test Conference.* DOI: 10.1109/TEST.2016.7805827

D. Xiang. 2001. Fault-tolerant routing in hypercube multicomputers using local safety information. *IEEE Trans. Parallel Distrib. Syst.* 12, 9 (2001), 942–951.

D. Xiang, B. Li, and Y. Fu. 2017. Fault-tolerant routing in dragonfly networks. (accepted to appear) DOI: 10.1109/TDSC.2017.2693372.

D. Xiang, Y. Zhang, and Y. Xu. 2013. A fault-tolerant routing algorithm design for on-chip optical networks. In *Proceedings of the 32th IEEE International Symposium on Reliable Distributed Systems.*

X. Yang, D. J. Evans, and G. M. 2005 The locally twisted cubes. *Int. J. Comput. Math.* 82, 4 (2005), 401–413.

W. Yu, H. Zhuang, C. Zhang, G. Hu, and Z. Liu. 2001. RWCap: A floating random walk solver for 3-D capacitance extraction of VLSI interconnects. *IEEE Trans. Comput.-Aid. Des.* 32, 3 (2001), 353–366.

Received January 2018; revised June 2018; accepted July 2018