Non-Blocking Testing for Network-on-Chip

Letian Huang, *Member, IEEE*, Junshi Wang, *Student Member, IEEE*, Masoumeh Ebrahimi, *Member, IEEE*, Masoud Daneshtalab, *Member, IEEE*, Xiaofan Zhang, Guangjun Li, and Axel Jantsch, *Member, IEEE*

Abstract—To achieve high reliability in on-chip networks, it is necessary to test the network as frequently as possible to detect physical failures before they lead to system-level failures. A main obstacle is that the circuit under test has to be isolated, resulting in network cuts and packet blockage which limit the testing frequency. To address this issue, we propose a comprehensive network-level approach which could test multiple routers simultaneously at high speed without blocking or dropping packets. We first introduce a reconfigurable router architecture allowing the cores to keep their connections with the network while the routers are under test. A deadlock-free and highly adaptive routing algorithm is proposed to support reconfigurations for testing. In addition, a testing sequence is defined to allow testing multiple routers to avoid dropping of packets. A procedure is proposed to control the behavior of the affected packets during the transition of a router from the normal to the testing mode and vice versa. This approach neither interrupts the execution of applications nor has a significant impact on the execution time. Experiments with the PARSEC benchmarks on an 8$\times$8 NoC-based chip multiprocessors show only 3 percent execution time increase with four routers simultaneously under test.

Index Terms—Reconfigurable router architecture, built-in self-test, on-chip interconnect, single-chip multiprocessors

1 Introduction

New sources of faults in transistors and interconnects are growing concern due to aggressive CMOS scaling [1]. Hence, researchers have turned their attention to methods to develop reliable systems from unreliable components, managing both design complexity and process uncertainty [2][3]. Many approaches have been proposed to tolerate various types of faults in different components of NoCs [4], [5], [6], [7], [8]. However, to take measures actively, detecting and diagnosing faults is a precondition in activate tolerance mechanisms.

Physical phenomena such as wear-out and process variation may cause circuit-level faults in different components of the network. Most of these faults have serious impacts on the network behavior and performance. For example, a physical failure may lead to an illegal turn which may form a cycle leading to a deadlock. Such situations are difficult to detect and there are few attempts in literature to address deadlock detection [9], [10]. In addition, the entire network may crash already before the deadlock is detected. Faults in the control path of a router introduce complex patterns which are beyond the capability of error-correcting codes (ECC). Common solutions such as End-to-End retransmission significantly increase the packet latency and the execution time of applications [11] in addition to the need of a supporting infrastructure such as an Ack/Nack protocol.

- *L. Huang, G. Li, J. Wang, and X. Zhang are with the University of Electronic Science and Technology of China, Chengdu, Sichuan, 611731, P.R. China.*
  E-mail: {huanglt, gjli}@uestc.edu.cn, {wangjsh, zhangxf}@std.uestc.edu.cn.
- *M. Ebrahimi and M. Daneshtalab are with the Royal Institute of Technology, Sweden. E-mail: {mebr, masdan}@kth.se.*
- *A. Jantsch is with the Institute of Computer Technology, Vienna University of Technology, Vienna, Austria. E-mail: axel.jantsch@ict.tuwien.ac.at.*

The "Incubation period" is defined as the period between the occurrence of a physical failure and the observation of a fault on the network level. If physical failures can be detected within the incubation period, the fault-tolerant mechanisms such as a proper routing algorithm or a redundancy technique can be applied to prevent network-level faults. Thereby, suitable mechanisms are demanded to detect and diagnose physical failures before they lead to serious effects on the network.

Among previous works in the realm of fault tolerance, some rely on off-line testing mechanisms [12], [13], [14]. These approaches are not scalable and they need additional resources for the testing purposes. Moreover, faults caused by environmental conditions such as thermals stress, IR-drop, and wear-out can be hardly detected by off-line testing due to the difficulty of creating similar working conditions in a dynamic environment and designing a proper testing mechanism for them. Other traditional error detection mechanisms deal with faults in the data path (e.g., buffers, crossbar unit, and link transmission) [15], [16] or control path (e.g., virtual channel allocation, switch allocation, and routing computation units) [17], [18] of a router. However, these approaches alert long after circuit faults or network failures have already occurred.

On top of the above mentioned mechanisms, built-in self-test (BIST) is another common mechanism in digital circuits to detect faults at run-time. The circuits under test are driven by specific input sets in order to cover as many potential faults as possible. The faults in the testing units can be diagnosed based on the responses of the input patterns. Multiplexers are used to isolate the circuit under test from the remaining part during the testing period. On-line BIST can be applied on both the data and control paths of a router. The fault coverage can be improved by carefully choosing the test patterns. More importantly, on-line BIST is able to raise the alarm at real time, rendering enough time to apply fault-tolerance techniques. Despite the advantages of BIST, the unit under test should be disconnected from the network using some wrappers during the testing procedure. Such isolations will affect the network performance heavily. In more details, the traditional methods mainly suffer from three problems. First, by disabling a router, the network connectivity and integrity will be disturbed which demands an advanced routing algorithm to support such irregularity. Second, the processing core will be disabled along with the routers under test (RUT), blocking all packets sending to/from this core or passing through the router. Third, the underlying application will be halted because of cache coherence protocols or data dependencies between the cores. Task migration may be a possible solution but it is a costly approach since a testing procedure usually does not last long.

In summary, to enhance the reliability of NoCs at real time, on-line BIST mechanisms should be applied as frequently as possible. This helps to detect and diagnose faults early enough and subsequently allows taking countermeasures. In this paper, we propose a network-level approach enabling BIST to be applied at real-time. The main contributions are as follows:

1) A reconfigurable router architecture is proposed with bypassing channels to keep all processing cores accessible from every other core in the network despite the isolation of RUT. In other words, this architecture guarantees the integrity between computational units and memory blocks in many-core systems.
2) An adaptive routing algorithm is proposed which is able to limit the performance loss imposed by disabled routers. This algorithm has the capability to support multiple RUT without any packet drop except some specific patterns of RUT. To prevent any packet drop in the network, we introduce a testing sequence where the unsupported patterns are never taken.
3) A network-level testing approach is proposed to control the affected packets in the transition phase of the routers from normal to testing mode and vice versa. Unlike traditional methods, the proposed approach does not block any packet during the testing process. A finite-state machine (FSM) controls the behavior of packets and allows them to be routed in the network.

Combining these techniques a comprehensive approach is obtained called TARRA where "T" stands for the network-level Testing strategy, "AR" refers to Adaptive Routing and "RA" represents the Reconfigurable Architecture.

The reminder of this paper is organized as follows. In Section 2, related work on fault detection and reconfigurable router architecture is reviewed. In Section 3, the proposed reconfigurable router architecture is described. Section 4 introduces the adaptive routing algorithm on the proposed router architecture. In Section 5, the network-level testing approach is presented. Experimental results on synthetic traffic profiles and PARSEC benchmarks are reported in Section 6 while the summary and conclusion are given in the last section.

2 Related Work

Fault detection and testing mechanisms are two main aspects in the domain of design for test. The fault detection aspect has been studied and investigated in several works [15], [16], [17], [18]. Among them, error detection codes (EDCs) are widely used in NoCs because they are light-weight and flexible [15], [16]. Hamming codes and parity check codes are popular though their capabilities are limited. NoCAlert which could be defined as assertion-based methods is proposed in [17], employing some checkers to detect any of the listed in variances. The checkers or monitors are light-weight and significantly smaller than the units they check. However, both EDCs and monitors could not pre-detect the faults and can only detect physical failures which have been triggered.

A typical BIST consists of different components in order to generate test vectors and analyze the outputs of the testing unit. BIST can be used both off-line and on-line. The inputs of the units are controlled by multiplexers and they are switched between normal packets or test packets generated by the BIST circuit. A wrapper is formed by the multiplexers used in each input and output along with the control logic. A wrapper may isolate one or several components inside a router [4] or it may cover an entire router in addition to some components from the neighboring routers [19], [20], [21], [22], [23]. A wrapper for an entire router based on IEEE 1500 is presented in [19]. The presented wrappers in [20], [21] and [22] cover the transmission link, the output buffers, and the input buffers of the neighboring routers. The presented wrapper in [23] includes the router along with the output buffers of upstream routers. If the wrapper only contains a small piece of the circuit within one router, the test vectors can drive the input of this circuit directly like the one in [4]. In most cases, test vectors are applied through packets so that the test vectors are stored in the payload flits. For detecting faults in the data path, the correctness of test vectors should be examined. To detect faults in the control path, the correct delivery of packets should be checked.

Different testing approaches have been proposed to organize the testing procedures more efficiently in terms of time, power consumption and hardware overhead. For those detection and diagnoses methods where test vectors are distributed through the network packets, it is vital to design the testing packets efficiently in order to increase the fault coverage.

In some works these packets are distributed all over the network. For example, in [24], 16 packets are used to cover the maximal aggressor fault (MAF) on all links, including network interfaces within a 4×4 mesh NoC. The proposed idea in [12] is based on sending test packets along straight-paths, turn-paths and resource-paths. The suggested paths in [25] are made up of straight-paths, turn-paths and U/Z-shape double-turn paths. Some other choices are presented in [26] and [27] where the packets containing the test vectors are transmitted within a 2×2 sub-network to detect the short circuit faults in both data and control paths. The contention between data packets and test packets in the network is the main shortcoming of the aforementioned works. Another disadvantage is that they cannot locate faults. Therefore, some of the presented approaches are based on sending test-vector packets between neighbors [23]. In [23], the test procedure is divided into nine phases with different selected paths. A multicast test-vector transport based testing approach is discussed in [28], which exploits the

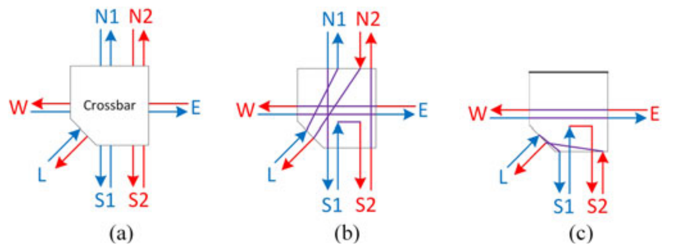


Fig. 1. The reconfigurable router with three different modes: (a) A router using crossbar unit, (b) default bypassing connections, (c) default bypassing connections of top borderline routers.

TABLE 1
Default Bypassing Connections Except Top Borderline Routers

Input channels	L	E	W	N1	N2	S1	S2
Output channels	N1	W	E	S1	L	S2	N2
Output channels of top border line	S1	W	E	-	-	S2	L

inherent parallelism of the data transport mechanism to reduce the test time and the test cost. The source and destination of test packets are chosen from the neighboring routers or the network interface of RUT. The work in [29] utilizes three different kinds of packets between RUT and its neighbors to detect faults on communication channels, output port arbiters and the routing units.

In traditional methods, to test a router, both the router and the connected processing element are disconnected from the network using a wrapper. The disabled part interrupts the normal traffic flow in the network which causes heavy traffic congestion. The testing approach in both [30] and [23] is based on BIST with wrappers to disconnect the testing parts. A token-based approach is used to determine the testing sequence and only one router is chosen at a time. In [30], there is no interval between testing two routers because it is an off-line testing approach. In [23], however, an interval of 10,000 flits is considered between each two tests to retrieve the network from congestion before testing another router.

The design of the reconfigurable router which could be used for keeping connectivity of the whole system while one or more routers are under test is the basis of non-blocking online testing for network-on-chip. Unfortunately, most works do not provide a proper solution. In [4], a bus is shared by all input and output ports and is used to replace the crossbar when it fails. Using this approach, cores can access the network through the bus but the performance would be largely affected. NoRD is proposed in [31] where a separate virtual channel is used to connect all cores to each other through a ring. The main shortcoming of NoRD is that the ring will be very long and the performance will significantly loss if the network scale is large.

In summary, even though a wide variety of testing approaches has been proposed, in all of them the execution of applications is interrupted. So, these approaches are not suitable where frequent on-line testing is demanded. The shortage is mainly caused by the incomplete connectivity of the network which forces blocking or dropping packets.

3 Reconfigurable Router Architecture

We have utilized the routing algorithm presented in [32] which is able to keep the core in the network despite connecting to a disabled router (either because of faults or testing the router). When a router is disabled, it is reconfigured and acted as a bypassing router, maintaining the connectivity between the routers in horizontal and vertical directions. In this paper, we improve the design of the reconfigurable router and the routing algorithm based on the idea of [32]. We first describe the reconfigurable router architecture. In the next section, a proper routing algorithm for this architecture is introduced.

3.1 Default Router Architecture

In the TARRA configuration, one and two physical channels are used along the X and Y dimension, respectively. This is the minimum amount of channels to provide fully adaptiveness in 2D mesh networks. As shown in Fig. 1a, each router has seven pairs of channels, i.e., Local(L), East(E), West(W), North1(N1), North2(N2), South1(S1) and South2 (S2). By default, the input and output channels are connected through a crossbar unit (Fig. 1a).

When a router is under test, the input channels are directly connected to the output channels through bypassing channels (Fig. 1b). In this situation, the local core delivers its packets through the north neighboring router by utilizing the N1 channel (i.e., L-to-N1). The core receives packets from the north neighboring router by using the N2 channel (i.e., N2-to-L). As shown in Table 1, the other static connections are as: N1-to-S1, S2-to-N2, S1-to-S2, E-to-W and W-to-E.

Bypassing connections are valid for all the routers except the top borderline routers as they do not have any north neighboring router to make a connection with. In this case, the local core sends and receives packets through its south neighboring router. Fig. 1c shows the default connections of the top borderline routers, also listed in Table 1. The bypassing connections are not only beneficial to keep the integrity of the network when the router is under test but also useful to reduce the latency. This is due to the fact that the bypassing connections directly connect the neighboring routers to each other without processing the packets in RUT. The north or south neighbor of RUT, which helps the packets to reach the core connected to RUT, is implicitly called “ladder router”.

Fig. 2 shows an example where three routers (i.e., at locations 4, 7, and 14) are under test and the bypassing connections are employed. The router 12 performs as the ladder router to send and deliver packets to/from the core 7 while the router 9 is the ladder router of both the cores 4 and 14.

3.2 General Rules Guranteeing Deadlock and Livelock Freedom

In the TARRA routing algorithm, the network can be partitioned into two disjoint subnetworks covering disjoint sets of channels as: ((X+)(Y1+)(Y1-) and (X-)(Y2+)(Y2-), called as subnetwork A (Fig. 3a) and subnetwork B (Fig. 3b), respectively.

A cycle cannot be formed within each subnetwork. This is due to the fact that to form a cycle at least four directions are needed (X+, X-, Y+ and Y-). However, each of the

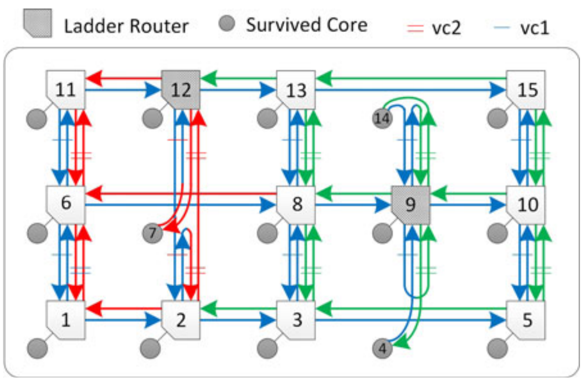


Fig. 2. An example of three testing routers and use of bypassing connections.

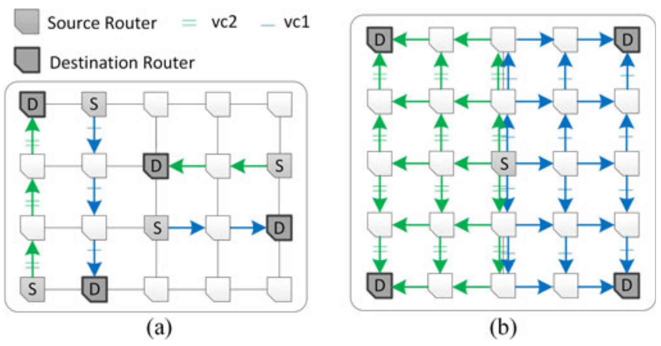


Fig. 4. The default path choices of the routing algorithm (the blue and green lines indicate the channels of the subnetwork A and the subnetwork B, respectively).

subnetwork A and B cover three directions as (X+, Y1+ and Y1-) and (X-, Y2+ and Y2-), respectively. Therefore, there is no possibility of forming a complete cycle within each subnetwork. It is also necessary to prove that subnetworks are disjoint from each other, so a cycle cannot be formed between subnetworks. As shown in Figs. 3a and 3b, two subnetworks are completely disjoint from each other and they do not share any channel with each other. As a result, the network is deadlock free if packets are solely belonging to either the subnetwork A or the subnetwork B. To improve the routing flexibility, packets can also switch between subnetworks, but only once. This will not lead to a deadlock because no complete cycle can be formed. As shown in Fig. 3c, in the TARRA algorithm, packets in the subnetwork A can safely switch to the subnetwork B but after the transition, packets are not able to take any channel of the subnetwork A anymore.

The TARRA algorithm is livelock free as in the worst case packets belonging to the subnetwork A reach the easternmost column and continue routing by switching to the subnetwork B. Since there is no chance of switching to the subnetwork A again, the movement is limited and the algorithm is livelock free.

4 Fully Adaptive Routing Algorithm

It is common that routing algorithms reorganize the traffic on routers in the case of faults which results in a significant performance loss. We propose a new routing algorithm for TARRA based on the algorithm in reference [32] to improve the performance. In this section, we first describe the propagation of the router status, i.e., disabled or working, in a 3×3 area. Each router should be informed about the status of eight direct/indirect neighboring routers. In the routing algorithm, the status of indirect neighboring routers (Northeast, Northwest, Southeast and Southwest) are used only when the destination router is under test and packets should be routed to the ladder router.

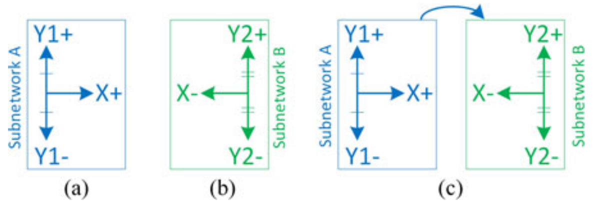


Fig. 3. Two disjoint subnetworks formed by dividing the channels.

In this section, the basic form of the TARRA algorithm without any RUT is introduced first. After that, we investigate all situations where an intermediate router, a source router or a destination router is under test. It will be shown that, all situations are covered by the TARRA algorithm and no packet is dropped in the network. The entire algorithm is given in Pseudo code. Finally, the capability of the routing algorithm to tolerate RUT is investigated.

4.1 Default Path Choices of the Routing Algorithm

In the TARRA routing algorithm, the east, south, northeast and southeast-bound packets are routed in the subnetwork A in which they can freely use any channels belonging to this subnetwork including the E, N1 and S1 channels (Fig. 3a). The west, north, northwest and southwest-bound packets are routed in the subnetwork B and they can use any of the W, N2 and S2 channels (Fig. 3b) without any limitation. In addition, all packets in the subnetwork A can switch to the subnetwork B but not vice versa. As shown in Fig. 4a, the east, west, north and south-bound packets route normally, respecting the channel assignment. For example, the south-bound packets use the S1 channels belonging to the subnetwork A while the north-bound packets use the N2 channels belonging to the subnetwork B.

As long as a packet has not reached the 3×3 area, a fully adaptive routing algorithm is applied to deliver the packet to the destination. The northeast, southeast, northeast and southwest-bound packets also choose their routes from two possible directions as X-axis or Y-axis according to the congestion of two neighboring routers. For example, as shown in Fig. 4b, a northwest-bound packet can choose its path from north or west freely. All packets use these default paths as long as they are outside of a 3×3 region with a router under test in the center.

4.2 Tolerating a Intermediate Router under Test

In this section, we investigate the cases that RUT is located in an intermediate router rather than the source or destination one. Fig. 5a shows examples where RUT is simply bypassed by the east, west, north and south-bound packets. Consistent with the router architecture of the bypassing router, shown in Fig. 1b, north-bound packets are able to bypass RUT using the N2 channel while south-bound packets use S1 in order to bypass RUT. Fig. 5b shows an example where a northeast packet is delivered from the source core 1

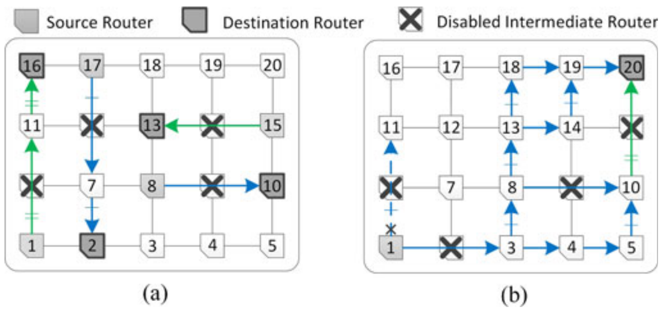


Fig. 5. Different locations of an intermediate router under test.

toward the destination core 20. Along the path, at each intermediate router, an available neighboring router is chosen for the packet (e.g., at the router 4 in Fig. 5b, the east neighbor is selected, and at the router 8, the north neighbor is chosen) unless both routers in minimal directions are under test (e.g., at the router 1).

In this case, the packet should be sent to the east direction to bypass RUT. The reason for this choice is that, if the north direction is selected, the packet has to be routed through N2, meaning that the packet switches to the subnetwork B. However, the subnetwork B does not cover the east direction and thus the packet cannot reach the destination from the router 11. Using the bypassing connections, the packet should be sent to the east direction at the router 14. However, since the router 15 is under test, the packet is sent to the north direction. Similar examples can be used for the southwest, northwest and southwest packets. In short, RUT can be ignored by selecting a functioning neighbor along the path. If both neighbors in a minimal path are under test, then the packet is routed through the bypassing connections. This example shows that all packets are able to reach from any source core to any destination core regardless of an intermediate router under test.

4.3 Tolerating a Destination Router under Test

Fig. 6 shows all situations where a packet is sent to a core, i.e., connected to a destination router under test, from any locations in the network. Fig. 6a shows the cases where a destination is located to the east and west of source routers. The survived core is accessible through its south neighboring router (using the N2 channel) when standing in the top borderline, otherwise through the north neighboring router (using the S2 channel). Westward packets belong to the subnetwork B where the W, N2, and S2 channels can be taken without any limitation. Eastward packets belong to the subnetwork A where the E, N1, and S1 channels can be freely taken. Packets must switch to the subnetwork B to take N2 or S2 and access the destination core. As it has been already discussed, switching from the subnetwork A to the subnetwork B is allowable.

Fig. 6b shows examples of northward and southward packets. Southward packets, by default, are routed through the S1 channel and they can bypass RUT using the same channel. However, if the destination router is under test, packets legitimately switch to the S2 channel which is directly connected to the destination core. For northward packets, by default, the N2 channel is used. Thereby if RUT is in the top borderline, the packet can directly reach the destination core. Otherwise, at first RUT is bypassed to reach the ladder router and then the packet is forwarded to the destination core.

As can be seen in Figs. 6c and 6d, packets are routed using a fully adaptive routing algorithm until they reach the 3×3 area surrounding RUT. Upon reaching this area, different routes are selected based on the position of the packet regarding the faulty router.

If the packet reaches north, south, west and east neighbors of the faulty destination, they can choose a path to the ladder router as shown in Figs. 6a and 6b. If the packets reach the southwest and southeast neighbors of the faulty destination (router 7 and router 9 in Fig. 6c), they can only choose the path to the south neighbor of RUT and then reach the ladder router through the bypassing channel. If the packets arrive at the northwest or northeast neighbors of the faulty destination (router 12 and router 14 in Fig. 6d) they can only choose the path to the ladder router.

This example shows that all packets are able to reach from any source core to any destination core regardless of the destination router under test and without violating any rule.

The TARRA algorithm respects the general rules of channel assignments in all conditions without exception. This guarantees that the algorithm is deadlock and livelock free. In the other words, in all cases, packets are either belonging to the subnetwork A or the subnetwork B. In addition, packets belonging to the subnetwork A can switch to the subnetwork B but not vice versa.

4.4 Tolerating a Source Router under Test

Fig. 7 shows all cases to prove that a packet can reach to any core in the network if delivered from a source core

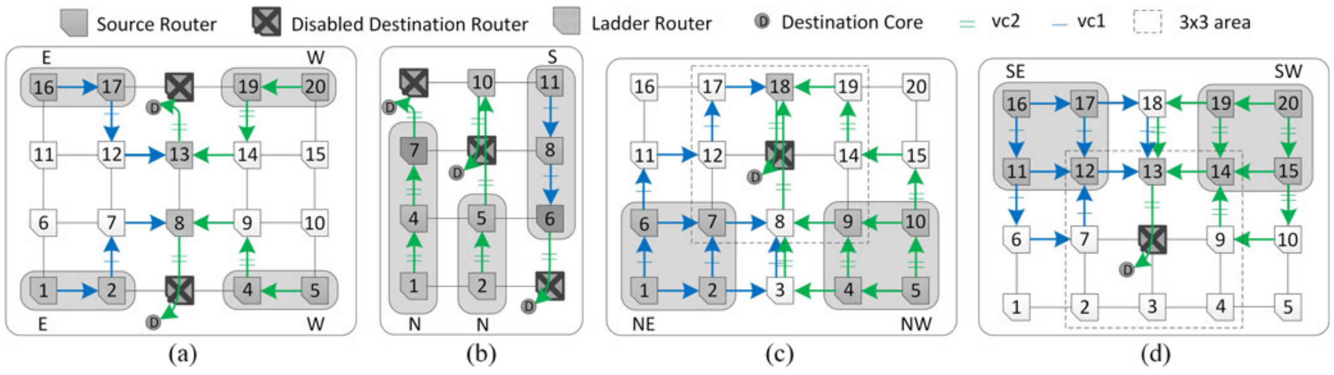


Fig. 6. Different locations of a destination router under test.

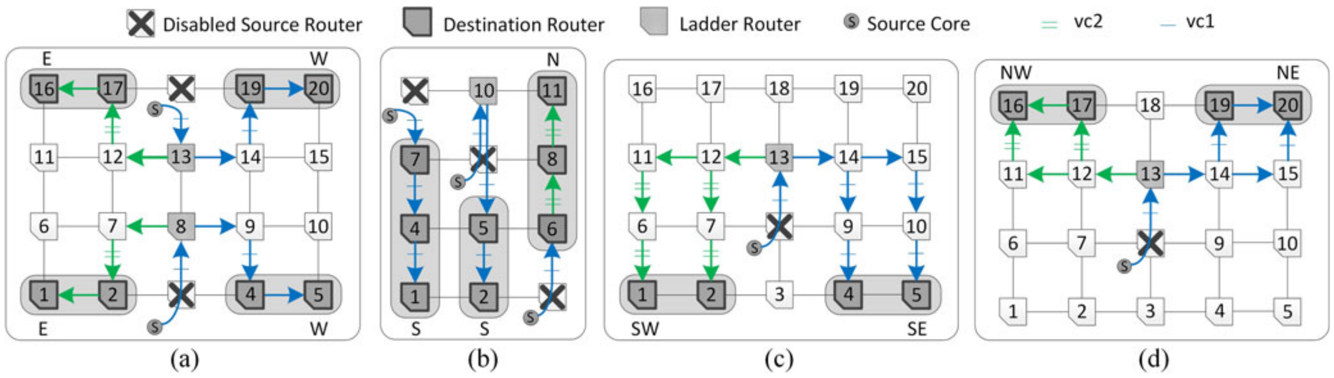


Fig. 7. Different locations of a source router under test.

connected to RUT. In all cases, the packet is delivered to the ladder router. For this transmission, first the N1 or S1 channel is used which belongs to the subnetwork A. The east, south, northeast, and southeast-bound packets continue in the subnetwork A to reach the destination router while the west, north, northwest and southwest-bound packets are routed in the subnetwork B. Thereby, all packets are able to reach from any source core to any destination core regardless whether a source router is under test. The complete Pseudo-code of the TARRA routing algorithm is shown in Fig. 8.

4.5 Reliability Analysis of Multiple Routers under Test

4.5.1 One Router under Test

We have already proved that all locations of a single router under test are supported in the network without any packet drop and by allowing all cores to function normally.

4.5.2 Two Router under Test

As it is shown in Fig. 9, only four patterns are not supported when two routers are under test at the same time. In traditional methods, routers are tested one by one and these patterns are never happened. However, when multiple routers are tested simultaneously, there is a possibility that these unsupported patterns occur. In this paper, a new testing sequence is defined in Section 5 and these unsupported patterns would never be selected according to this testing sequence.

4.5.3 Three or More Routers under Test

The unsupported patterns with three or more routers under test consist of the unsupported patterns with two routers under test and any single router under test. These patterns should be also avoided when multiple testing is applied, covered by the defined testing sequence strategy.

5 Network-Level Testing Strategy

5.1 Testing Sequence of Routers

We define *TT* as a testing cycle for single router and *TIT* (Test Interval Time) as a complete testing cycle within which all routers are tested, backing to the first router again (Fig. 10). The time interval between the start of testing two adjacent routers (e.g., router 0 and router 1 in Fig. 10) can be

```
Require: Current: ($X_C$,$Y_C$); Source: ($X_S$,$Y_S$); Destination:
    ($X_D$,$Y_D$); $\Delta_X$: $|X_D - X_C|$; $\Delta_Y$: $|Y_D - Y_C|$;
Ensure: Selected port: $Sel$
1: if $X_C = X_D$ and $Y_C = Y_D$ then $Sel \leftarrow$Local;
2: else if $X_C < X_D$ and $Y_C = Y_D$ then
3:     if Neighbor(E)=functioning then $Sel \leftarrow$E;
4:     else if Neighbor(N)=available then $Sel \leftarrow$N1;
5:     else$Sel \leftarrow$S1;
6:     end if
7: else if $X_C > X_D$ and $Y_C = Y_D$ then
8:     if Neighbor(W)=functioning then $Sel \leftarrow$W;
9:     else if Neighbor(N)=available then $Sel \leftarrow$N2;
10:    else$Sel \leftarrow$S2;
11:    end if
12: else if $X_C = X_D$ and $Y_C < Y_D$ then
13:    if Neighbor(N)=under-test then $Sel \leftarrow$N2;
14:    else if $X_D > X_S$ then $Sel \leftarrow$N1;
15:    else$Sel \leftarrow$N2;
16:    end if
17: else if $X_C = X_D$ and $Y_C > Y_D$ then
18:    if Neighbor(S)=under-test then
19:        if Neighbor(S)=D then $Sel \leftarrow$S2;
20:        else$Sel \leftarrow$S1;
21:        end if
22:    else if $X_S > X_D$ then $Sel \leftarrow$S2;
23:    else$Sel \leftarrow$S1;
24:    end if
25: else if $X_C < X_D$ and $Y_C < Y_D$ then
26:    if $\Delta_X = 1$ and $\Delta_Y = 1$ and destination router is under
   test then $Sel \leftarrow$E;
27:    else$Sel \leftarrow$N1 or E according to congestion;
28:    end if
29: else if $X_C > X_D$ and $Y_C < Y_D$ then
30:    if $\Delta_X = 1$ and $\Delta_Y = 1$ and destination router is under
   test then $Sel \leftarrow$W;
31:    else$Sel \leftarrow$N2 or W according to congestion;
32:    end if
33: else if $X_C < X_D$ and $Y_C > Y_D$ then
34:    if $\Delta_X = 1$ and $\Delta_Y = 1$ and destination router is under
   test then $Sel \leftarrow$E;
35:    else$Sel \leftarrow$S1 or E according to congestion;
36:    end if
37: else$X_C > X_D$ and $Y_C > Y_D$
38:    if $\Delta_X = 1$ and $\Delta_Y = 1$ and destination router is under
   test then $Sel \leftarrow$W;
39:    else$Sel \leftarrow$S2 or W according to congestion;
40:    end if
41: end if;
```

Fig. 8. The pseudo-code of the TARRA routing algorithm.

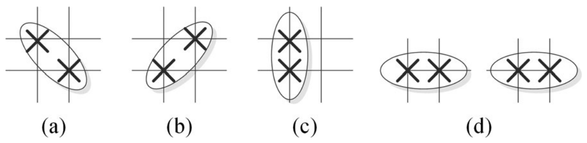


Fig. 9. Unsupport patterns of two routers are under test at the same time (a) diagonal-A, (b) diagonal-B, (c) columnar, (d) row-at-edge.

obtained by dividing TIT to the number of routers in the network. If testing is applied more frequently, TIT might become shorter than the overall testing cycle of all routers, and thus the test of multiple routers is overlapped. The number of overlapped RUT at one time can be calculated as:

$$\#\text{overlapped RUT} = \left\lceil \frac{TT \times \#\text{router}}{TIT} \right\rceil \tag{1}$$

The range of TIT for k overlapped RUT is

$$\frac{TT \times \#\text{router}}{k} \leq TIT < \frac{TT \times \#\text{router}}{k-1} \tag{2}$$

We propose the idea of testing multiple routers which is supported by the proposed reconfigurable architecture and its supportive routing algorithm.

In multiple RUT, it is essentially important that disabling several routers does not lead to any packet drop which has been guaranteed in the defined testing sequence strategy.

Each router contains a timer driven by the clock signal, called "testing interval timer". The timer may have different initial values (TIV) for different routers and will be reset when it overflows (Fig. 10). The sequence of routers going to the test mode is controlled by this initial value. The initial value can be measured by:

$$TIV(A) = \frac{Seq(A) \times TIT}{\#\text{router}} \tag{3}$$

in which, $TIV(A)$ is the initial value of router A and $Seq(A)$ is the sequence number of router A in test sequence.

The test sequence determines the order of routers in the testing cycle. Two common types of sequences are as the natural sequence and ring sequence. In the natural sequence, ordering is simply obtained by reducing/increasing the router number by 1 as shown in Fig. 11a.

Under a high testing frequency, multiple routers should be tested together respecting the testing sequence strategy. Thereby, some testing patterns such as testing two adjacent routers (e.g., the router 0 and 1) may fall into the

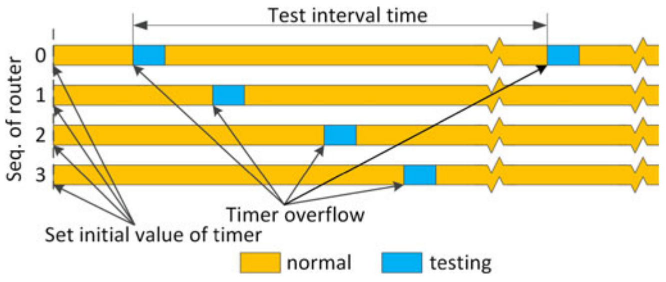


Fig. 10. Testing control.

unsupported patterns similar to Fig. 9d which in turn may lead to dropping packets. In the ring sequence technique [23], the testing order is shown in Fig. 11b. Based on this testing sequence, some unsupported patterns similar to Figs. 9c and 9d may be formed.

Considering the unsupported patterns in Fig. 9, we propose an odd-even sequence which allows multiple testing without any packet drop. In the odd-even sequence, first the routers with odd numbers are tested and then those with even numbers. The proposed odd-even sequence is shown in Fig. 11c. This testing sequence does not lead to any unsupported patterns as long as the maximum number of routers under test does not exceed $\frac{a}{2}$ in the $a \times b$ network.

5.2 Control over the Transition Phases of Routers

In the normal functioning mode of a router, the crossbar unit is used to deliver packets from an input port to an output port while in the testing mode, default bypassing channels are utilized. In the testing stage, the crossbar unit should be disabled and the default bypassing connections should be established. In our proposed method, the testing mode involves three stages as Emptying, Testing, and Recovering, shown in Fig. 13 and described as follows.

In the emptying stage, the crossbar unit functions normally. This stage includes two sub-steps as:

1) *Routing completion of packets on fly*: First, packets on fly should be routed before any phase transition. Let us assume that the router A is going to be tested and thus it needs to be switched from the normal mode to the bypassing mode. For this purpose, it sends an empty-request signal to the neighboring routers. Upon receiving this request, the neighboring routers temporarily stop sending any new packets to this router (Fig. 12a) but proceed to deliver all packets which have already been partially delivered (Fig. 12b). By transferring all flits of incomplete

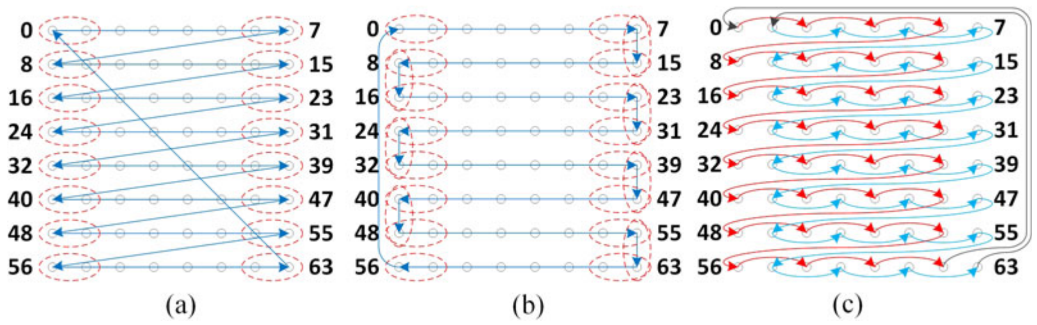


Fig. 11. Three different test sequence: (a) natural sequence, (b) ring sequence, (c) odd-even sequence.

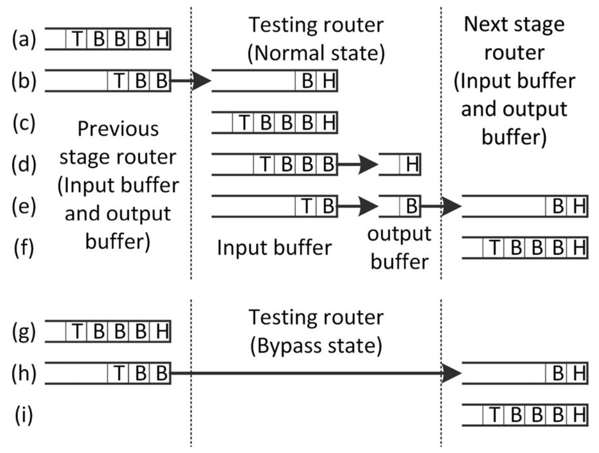

(a)
(b)
(c)
(d)
(e)
(f)
Previous stage router (Input buffer and output buffer)
Testing router (Normal state)
Input buffer
output buffer
Next stage router (Input buffer and output buffer)
(g)
(h)
(i)
Testing router (Bypass state)
T B B B H

Fig. 12. Different types of packets during emptying and recovery. (a)-(f) are packets during emptying phase; (g)-(i) are packets during recovery phase; (H: head flit; B: body flit; T: Tail flit).

packets, the neighboring router sends an empty-acknowledge signal to the router A.

2) *Delivering all packets already inside the router*: Second, packets which are partially/completely stored in the buffers should be routed completely. The router A requires to empty all its input buffers following the selected routes (Figs. 12c, 12d, and 12e). In the situation of Fig. 12f, packets are already transferred to the next router and thus they are not affected by the transition phase. If the packets could not be ejected to the neighboring router, they will be blocked in the emptying router and wait temporarily. As soon as the router A is emptied and all empty-acknowledge signals are received from the neighboring routers, the crossbar unit is disabled and the bypassing links are activated.

When the router goes to the testing mode, it works as a bypassing router where packets can be delivered in horizontal and vertical directions, keeping the connectivity with the processing unit as described in Section 3. Right after establishing the bypassing links, the neighboring routers start sending packets again, implying that the period of blocking packets is very short and not affecting the network performance.

To perform the actual test on the router in the testing stage, a proper approach such as BIST should be applied. The detail of such testing approaches is beyond the scope of this paper.

In the recovering stage, when the router goes from the testing mode to the normal mode, the default bypassing connections should keep activated. For this purpose, all incomplete packets in the corresponding ports of the neighboring routers are completely delivered.

The router A sends empty-request signals to the neighboring routers. The neighboring routers deliver all incomplete packets using the bypassing channels and stop sending any new packets to RUT. When this action is performed, the neighboring routers send empty-acknowledge signal to the router A. By receiving all empty-acknowledge signals at the router A, it enters the normal mode where the crossbar unit should be enabled and the bypassing connections should be disabled.

Different types of packets during the recovering mode are shown in Figs. 12g, 12h, 12i. In the situation of Fig. 12g packets should be temporarily blocked in the neighboring routers until the transition is completed and the router backs to its normal functioning mode. In Fig. 12h packets should be completely delivered using the bypassing channels. The packets of Fig. 12i are already delivered to the neighboring router and thus they are not affected by this transition phase.

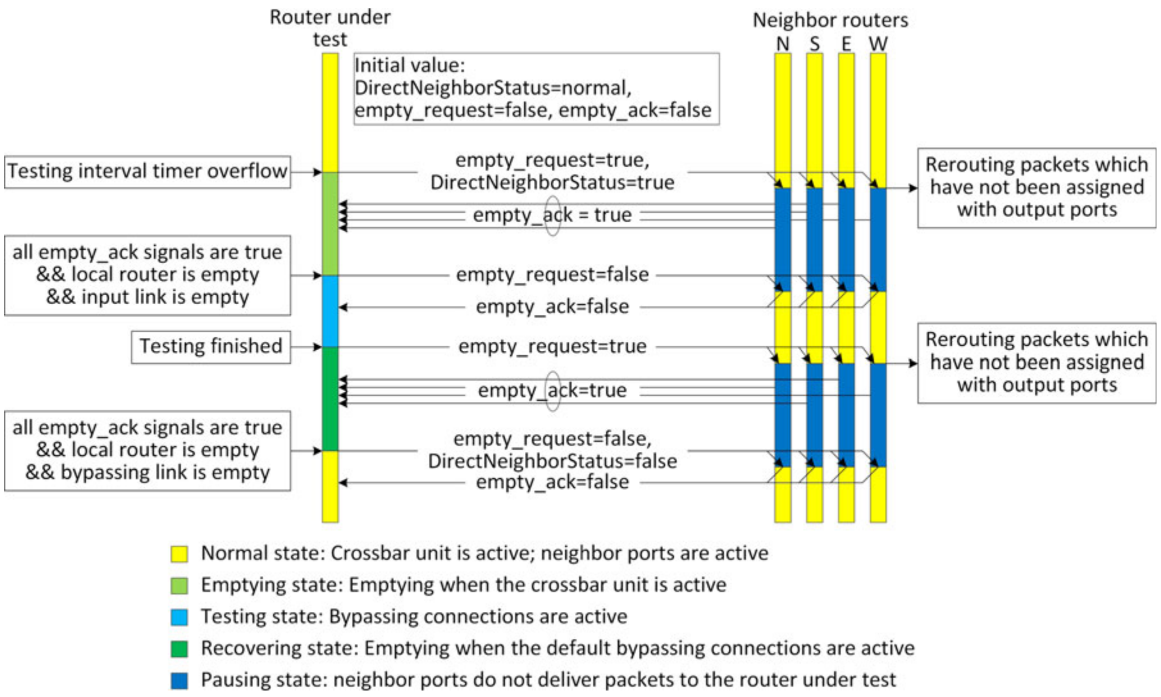

Router under test
Neighbor routers
N S E W
Initial value:
DirectNeighborStatus=normal,
empty_request=false, empty_ack=false
Testing interval timer overflow
empty_request=true,
DirectNeighborStatus=true
empty_ack = true
Rerouting packets which have not been assigned with output ports
all empty_ack signals are true && local router is empty && input link is empty
empty_request=false
empty_ack=false
Testing finished
empty_request=true
empty_ack=true
Rerouting packets which have not been assigned with output ports
all empty_ack signals are true && local router is empty && bypassing link is empty
empty_request=false,
DirectNeighborStatus=false
empty_ack=false
Normal state: Crossbar unit is active; neighbor ports are active
Emptying state: Emptying when the crossbar unit is active
Testing state: Bypassing connections are active
Recovering state: Emptying when the default bypassing connections are active
Pausing state: neighbor ports do not deliver packets to the router under test

Fig. 13. Neighboring ports test state machine and control over the transition phases. The FSM of the router under test and the neighboring routers are illustrated using blocks in the left and right side of the figure, respectively. The handshake signals between the router under test and the neighboring ports are shown in the center.

TABLE 2
Relationship between TIT and the Number of Overlapped RUT

overlap router(s) \ test cycle	500-cycle	1000-cycle
1	$TIT \geq 32{,}000$	$TIT \geq 64{,}000$
2	$16{,}000 \geq TIT > 32{,}000$	$32{,}000 \geq TIT > 64{,}000$
3	$10{,}889 \geq TIT > 16{,}000$	$21{,}333 \geq TIT > 32{,}000$
4	$8{,}000 \geq TIT > 10{,}889$	$16{,}000 \geq TIT > 21{,}333$

The whole procedure is shown in Fig. 13. Testing a router starts when the testing interval timer overflows and thereby the router enters the emptying stage. Two handshake signals, empty-request and empty-acknowledge are used to synchronize between RUT and its neighboring routers.

5.3 Testing of Additional Circuits

In TARRA, the additional components of testing such as bypassing channels, testing interval timer, and finite-states machine are also prone to fault. As these circuits are not used during the normal functioning mode of a router, they can be easily tested by BIST.

6 Experiment Results and Discussion

6.1 Experiment Setup

To evaluate the efficiency and performance of TARRA, we performed system simulations. We use SimpleScalar augmented with POPNET simulating an 8×8 NoC-based CMP. Sixty-four CPU cores are configured based on Alpha 21264. Each processor node has a private 32 KB L1 cache and 128 KB L2 cache (8 MB shared distributed L2 for the entire system). Each router in this network has seven physical ports (one for each of the local, east and west ports and two for each of the north and south ports) with a 12-flit buffer at each physical port. Each packet has 5 flits. No virtual channel is employed.

In [4], it takes 1,000 cycles to test the data path of a router and 1,50,000 cycles to test the control path with 25,000 patterns. In [23], the duration of testing procedure is about 1,200 cycles for a different testing model. Thus, we ran simulations based on two values for the testing period of a single router:

1) 500 cycles: Based on these assumptions, an entire testing period takes 32,000 clock cycles (=500×64 routers) without considering any timing interval between two routers. If the frequency of testing increases ($TIT <$ 32,000 clock cycles), multiple routers have to be tested in parallel. On the other hand, by decreasing the testing frequency ($TIT >$ 32,000 clock cycles), the testing interval time between two router increases. To cover different cases, we chose TIT range of 10,000 to 1 million for the entire testing period where we consider both multiple RUT (from 2 to 4 routers simultaneously) and one RUT with various timing intervals between each two routers. The relationship between TIT and the number of overlapped RUT is shown in Table 2.
2) 1000 cycles: In this case, an entire testing period takes 64,000 clock cycles (= 1000×64 routers). Thereby, the range of 10,000 to 1 million clock cycles is a reasonable assumption, covering all different cases of multiple and single router(s) under test.

TABLE 3
Traffic Profiles Used in Simulation

Uniform	To each router chosen randomly.
Transpose1	(x,y) → (7-y,7-x).
Transpose2	(x,y) → (y,x).
BitReversal	Destination id is the bit reversal of source id.
Suffle	Destination id is the right loop shift of source id.
Butterfly	Get destination id by swaping the LSB and MSB of source id.

TARRA is compared with the baseline approach which has been described in [23]. In the baseline method [23], packets have to be blocked in the input buffers until the testing period is ended. In the other words, packets are blocked not only during the emptying and recovering phase but also during the testing phase. This freezing period will quickly result in blocking packets in the network, which will be resolved when the router backs to its normal functioning mode. All the other conditions are the same for the proposed and the baseline method such as the applied testing sequence.

The different synthetic traffic profiles which are used for evaluation the latency are shown in Table 3. The latency is obtained based on two different packet injection rates (PIR) 0.005 packets/cycle/router and 0.020 packets/cycle/router.

To evaluate the efficiency of TARRA and the baseline method [23] on real applications, we measure the execution time on PARSEC benchmarks [33]. Applications are executed for 100 million instructions. The execution time is measured under the similar condition (i.e., the testing period of one router and the range of total testing cycle) as in the synthetic traffic profiles.

6.2 Hardware Overhead

To assess the area overhead and power consumption, TARRA and [23] router architectures are synthesized using Synopsys Design Compiler. For synthesizing, we use the TSMC45nm technology at the operating frequency of 1 GHz and supply voltage of 0.9 V. The overhead of the BIST circuit is not included as the test circuit design is outside the scope of this paper. The power consumption and area overhead of TARRA and the baseline methods are comparable as reported in Table 4. The small differences (3 percent leakage power, 6 percent dynamic power, and 4 percent area overhead) are because of employing the bypassing connections in the TARRA architecture which have not been used in the baseline method.

TABLE 4
Power and Area Analysis

	TARRA	Baseline [23]
Leakage Power	749.14μW	720.47μW
Dynamic Power	483.08μW	452.71μW
Area	50.74μm^2	48.60μm^2

**The area and power of BIST circuits are not considered in table.*

TABLE 5
Number of Cycles in the Emptying and Recovering States under Different Traffic Profiles

Traffic Profiles	Empty Cycle (cycle) Min	Empty Cycle (cycle) Aver	Empty Cycle (cycle) Max	Recover Cycle (cycle) Min	Recover Cycle (cycle) Aver	Recover Cycle (cycle) Max
PIR = 0.005 packets/cycle/router						
Uniform	1.00	1.40	1.47	1.00	1.24	1.67
Transpose1	1.16	1.44	2.00	1.00	1.20	1.50
Transpose2	1.13	1.59	2.67	1.00	1.15	1.33
BitReversal	1.33	1.54	2.17	1.00	1.20	1.50
Shuffle	1.00	1.32	1.47	1.00	1.16	1.42
Butterfly	1.00	1.27	1.45	1.00	1.07	1.13
PIR = 0.020 packets/cycle/router						
Uniform	2.39	2.68	3.17	1.33	1.78	2.33
Transpose1	1.83	2.61	3.33	1.17	1.65	2.83
Transpose2	2.63	2.99	3.67	1.39	1.57	1.67
BitReversal	1.83	2.70	3.29	1.41	1.74	2.33
Shuffle	2.23	2.55	3.58	1.33	1.59	2.00
Butterfly	1.17	2.01	3.10	1.00	1.31	1.74

6.3 Timing Penalty during Emptying and Recovering Phases

As already described, packets affected by RUT are only blocked in the emptying and recovering phases and are routed in the other phases (normal and testing modes). We argued that the emptying and recovering states are too short to affect the network performance. In Table 5 and Table 6, we reported the average number of cycle spent in these periods under synthetic and PARSEC benchmarks, respectively. We give out the maximum, average, and minimum value among different TIT. The maximum number of average cycles is 3.67 and 2.67 under the Transpose2 traffic profile when the packet injection rate is 0.020 and 0.005 packet/cycle/router. On average less than 2 cycles is spent on the emptying and recovering periods.

Either emptying or recovery periods are much shorter than TIT. So, the network would not be blocked and packets are not dropped if the packet injection rate is not very high.

TABLE 6
Number of Cycles in the Emptying and Recovering States under PARSEC Benchmarks

Benchmark	Empty Cycle (cycle) Min	Empty Cycle (cycle) Aver	Empty Cycle (cycle) Max	Recover Cycle (cycle) Min	Recover Cycle (cycle) Aver	Recover Cycle (cycle) Max
blackscholes	1.00	1.22	1.32	1.03	1.10	1.19
caneal	1.46	1.57	1.69	1.13	1.22	1.27
dedup	1.06	1.09	1.12	1.00	1.03	1.06
fluidanimate	1.75	1.85	1.97	1.33	1.39	1.49
freqmine	1.00	1.01	1.01	1.00	1.00	1.00
raytrace	1.23	1.30	1.43	1.05	1.11	1.13
streamcluster	1.13	1.15	1.23	1.05	1.07	1.11
swaptions	1.00	1.01	1.01	1.00	1.00	1.01
vips	1.02	1.03	1.05	1.00	1.01	1.02

Comparing the emptying and recovery periods, it is obvious that the emptying mode takes longer than the recovery mode. The reason is that the emptying phase includes both delivering the incomplete packets in the local router and in the neighboring routers. The recovering phase, on the other hand, only deals with the incomplete packets in the neighboring routers as the local router is under test and bypassed.

Regarding the packet injection rate (Table 5), a higher PIR leads to a longer emptying and recovering phases. This behavior can be explained by the queueing theory. A higher PIR leads to a more number of flits stored in buffers which have to be completely delivered during emptying and recovering phases.

6.4 Latency Measurement under Different Traffic Profiles

Figs. 14 and 15 illustrate the average latency under different traffic profiles with the packet injection rate of 0.005 and 0.020 packet/cycle/router, respectively. The most important point in these figures is that on-line testing does not affect performance when TARRA is applied. This includes both single router under test with different timing intervals and multiple routers under test up to four simultaneous

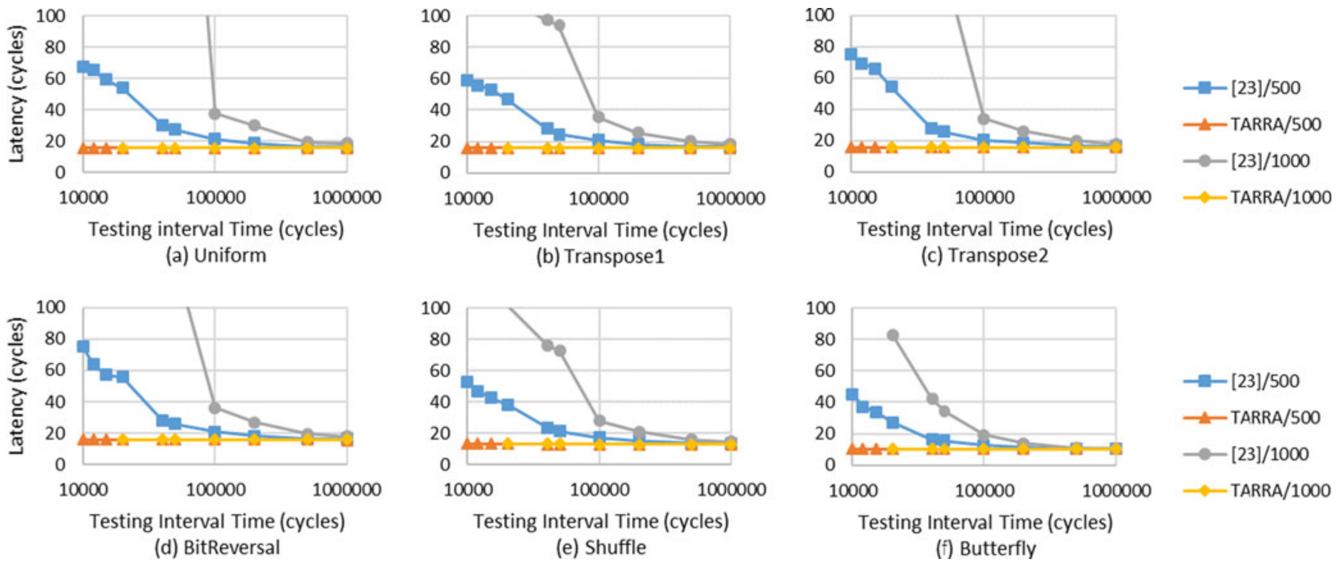


Fig. 14. Average latency under different traffic profiles for the packet injection rate of 0.005 packet/cycle/router. 500 and 1,000 represent the number of testing cycles at each router.

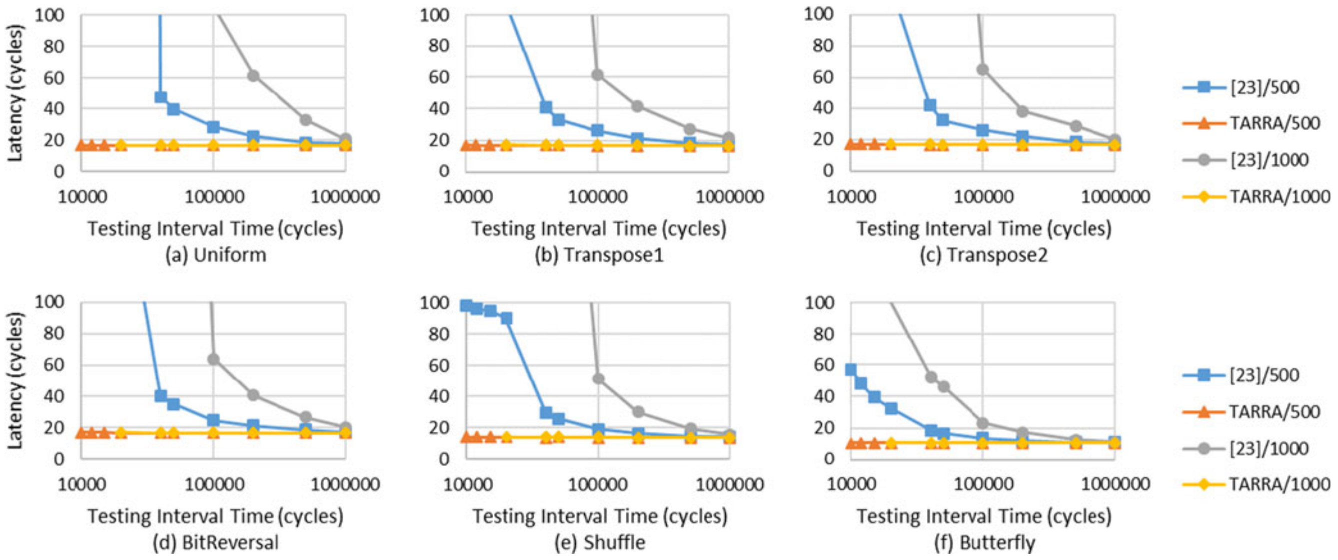


Fig. 15. Average latency under different traffic profiles for the packet injection rate of 0.02 packet/cycle/router. 500 and 1,000 represent the number of testing cycles at each router.

routers. For uniform traffic, the latency only increased by 0.1 cycle and 0.2 cycle at most with two PIRs, respectively. The main reason is that the bypassing approach enables packets to be routed in the network during the testing period which is in contrast with the baseline method where packets have to be blocked in the neighboring buffers of RUT, resulting in the network contention quickly.

Another observation is the significant negative effect of the decreased testing period on the execution time of the baseline approach. The on-line testing approach cannot be applied in the baseline method when the entire testing period is less than 50,000 and 100,000 cycles considering the testing period of 500 and 1000 cycles for a single router, respectively. This is far beyond the capability of testing multiple routers. However, in our approach, the effect of the testing period and TIT is negligible on the latency.

However, for TARRA, latency is not influenced significantly by the testing process and TIT. On-line testing

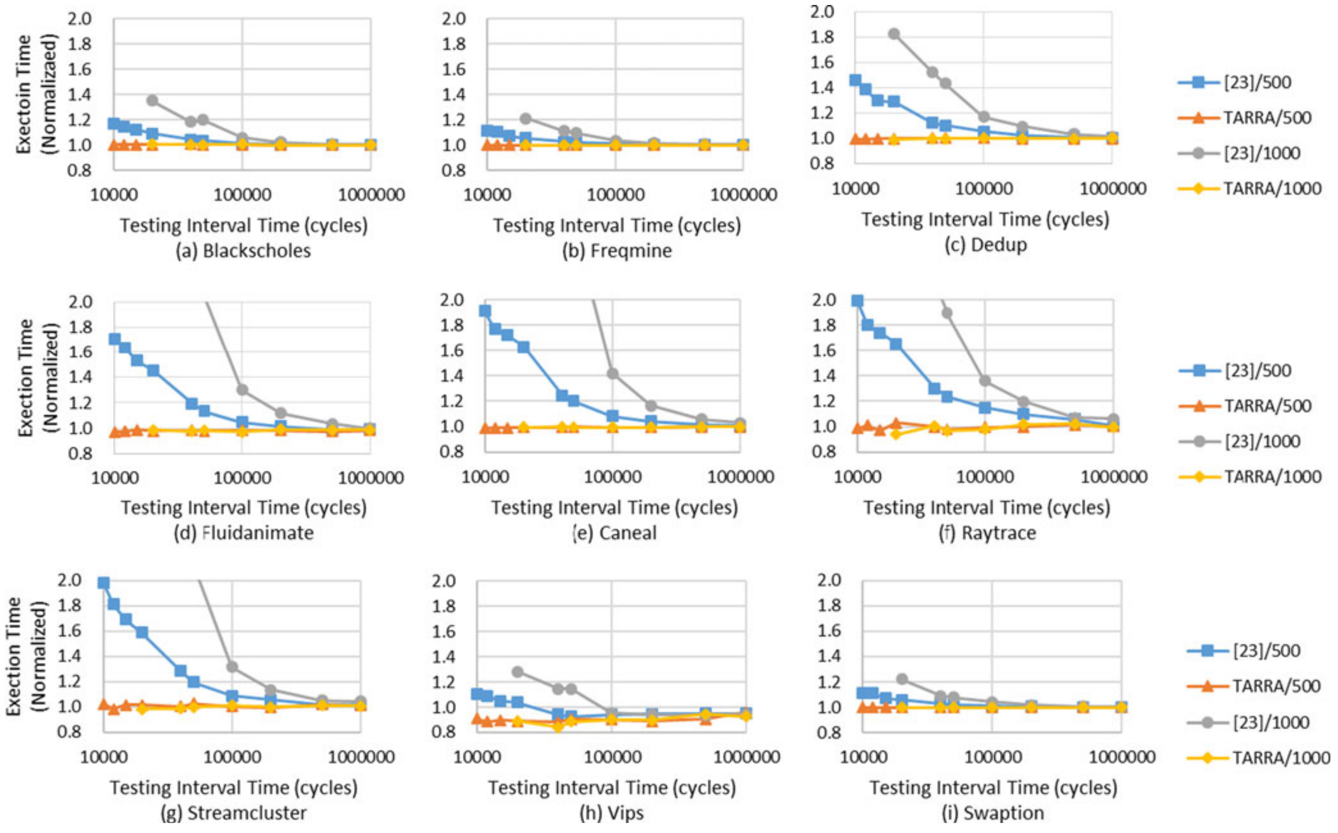


Fig. 16. Execution time of different real application benchmarks normalized to the situation where no router is under test. 500 and 1,000 represent the number of testing cycles at each router.

TABLE 7
Normalized Maximum and Minimum Execution Time of the PARSEC Benchmarks

	500-cycle test		1000-cycle test	
Benchmark	Maximum	Minimum	Maximum	Minimum
blackscholes	1.0065	1.0001	1.0069	1.0002
caneal	1.0002	0.9879	0.9986	0.9933
dedup	1.0034	0.9948	1.0026	0.9941
fluidanimate	0.9848	0.9700	0.9895	0.9699
freqmine	0.9997	0.9985	0.9999	0.9985
raytrace	1.0287	0.9707	1.0204	0.9374
streamcluster	1.0290	0.9832	1.0116	0.9842
swaptions	0.9999	0.9985	0.9999	0.9989
vips	0.9618	0.8802	0.9466	0.8410

**Normalized by the application execution time without testing*

process only introduced lower than 0.07 cycles deviation when PIR is 0.005 packet/cycle/router comparing with the latency without testing. If PIR is 0.020 packet/cycle/router, the deviation is lower than 0.2 cycles. The small impact on the performance is the result of the full connectivity of the network during the testing phase. No packet is blocked during the testing phase but they are temporarily blocked (i.e., the average of blocking period is less than 2 cycles) only in the periods of emptying and recovering which are too short to affect the performance (see also in Section 6.3).

6.5 Execution Time of the PARSEC Benchmarks

The execution time of different applications within the PARSEC benchmark is illustrated in Fig. 16. Simulations are performed on nine applications as Blackscholes, Caneal, Dedup, Fluidanimate, Freqmine, Raytrace, Streamcluster, Swaption, and Vips. The execution time is normalized on the case when there is no router under test.

As expected, the execution time of TARRA is affected neither by the duration of testing (e.g., 500 and 1,000 cycles per router) nor the number of routers under test (i.e., 1 to 4 routers). In some cases, the execution time is also reduced because of employing bypassing links and shortening the path between the working routers. On the other hand, the effect of the testing period on the execution time of the baseline method is significant, particularly in the Canneal, Dedup, Fluidanimate, Raytrace, and Streamcluster applications. This is mainly because of blocking the packets during the testing period. The duration of the testing period has a smaller effect on the execution time of Blackscholes, Freqmine, and Swaption which is because of a looser data dependency in these applications.

To quantitatively evaluate the effect of on-line testing on the execution time of TARRA, the maximum and minimum values of the normalized execution time are listed in Table 7. As shown in this table, the execution time has been increased in some cases (> 1) while decreases in others (< 1). The reason for this phenomenon is that the latency of some packets is reduced because of applying the bypassing channels and thus shortening some paths.

It can be obtained that TARRA can increase the testing frequency with negligible positive/negative impact on the execution time. In addition under the same testing interval time, the execution time can be largely decreased. This approach is essentially important in Multi-core System-on-Chip (McSoC) because of its inherent data dependency.

7 Conclusion

We first proposed a reconfigurable router architecture which allows the network to be fully connected when a router enters the testing mode. The processing core connected to the router under test performs normally without being interrupted by the underlying on-line testing. This has been achieved by using bypassing links and wrappers which isolate the router from both the network and the core, allowing the packets to bypass the router under test. We then proposed a highly adaptive routing algorithm for this reconfigurable architecture. This algorithm is able to work with a single router under test without any packet drop. With multiple routers under test, this algorithm can deliver all packets successfully except a few patterns where some packets may be dropped. To avoid any packet drop in the network, we suggested an odd-even testing sequence which avoids unsupported patterns. This routing algorithm along with the reconfigurable architecture allows testing multiple routers simultaneously, reducing the overall testing period and its effect on the network performance and the applications. Finally, we described the whole process of testing with the router switching from the normal to the testing mode and back. During these transitions, all possible situations of packets were considered in details. Experiments with standard traffic profiles and the PARSEC benchmark show that on-line testing can be applied without interrupting applications and with negligible effect on the latency and the execution time.

Acknowledgments

The authors would like to thank the anonymous reviewers for their constructive and helpful suggestions and comments. This paper was supported by the National Natural Science Foundation of China under grant (NSFC) No.61176025, No.61006027, No.61534002, and the Oversea Academic Training Funds (OATF), UESTC. This work is also supported by VINNOVA-MarieCurie within the CUBRIC and ERoT projects and Academy of Finland.

References

[1] J. McPherson, "Reliability challenges for 45nm and beyond," in *Proc. 43th ACM/IEEE Des. Autom Conf.*, 2006, p. 3.
[2] S. Mitra, K. Brelsford, Y. M. Kim, H.-H. Lee, and Y. Li, "Robust system design to voercome CMOS reliability challenges," *IEEE J. Emerging and Select. Topics in Circuits and Syst.*, vol. 1, no. 1, pp. 30–41, Mar. 2011.
[3] S. Borkar, "Designing reliable systems from unreliable components: The challenges of transistor variability and degradation," *IEEE Micro*, vol. 25, no. 6, pp. 10–16, Nov./Dec. 2005.
[4] A. DeOrio, D. Fick, V. Bertacco, D. Sylvester, D. Blaauw, J. Hu, and G. Chen, "A reliable routing architecture and algorithm for nocs," *IEEE Trans. Comput.-Aided Des. Integr. Circuits and Syst.*, vol. 31, no. 5, pp. 726–739, May 2012.
[5] M. Kakoee, V. Bertacco, and L. Benini, "Relinoc: A reliable network for priority-based on-chip communication," in *Proc. Des., Autom. & Test in Eur. Conf. & Exhibition*, 2011, pp. 1–6.
[6] S. Rodrigo, J. Flich, A. Roca, S. Medardoni, D. Bertozzi, J.Camacho, F. Silla, and J. Duato, "Addressing manufacturing challenges with cost-efficient fault tolerant routing," in *Proc. Fourth ACM/IEEE Int. Symp. Netw.-on-Chip (NOCS)*, 2010, pp. 25–32.

[7] W. Song, D. Edwards, J. Nunez-Yanez, and S. Dasgupata, "Adaptive stochastic routing in fault-tolerant on-chip networks," in *Proc. Third ACM/IEEE Int. Symp. Netw.-on-Chip*, 2009, pp. 32–37.

[8] M. Ebrahimi, M. Daneshtalab, J. Plosila, and F. Mehdipour, "Md: Minimal path-based fault-tolerant routing in on-chip networks," in *Proc. 18th Asia and South Pacific Des. Autom. Conf. (ASP-DAC)*, 2013, pp. 35–40.

[9] Y.-R. Chen, Z.-R. Wang, P.-A. Hsiung, S.-J. Chen, and M.-H. Tsai, "Backward probing deadlock detection for networks-on-chip," in *Proc. Seventh IEEE/ACM Int. Symp. Netw. on Chip (NoCS)*, 2013, pp. 1–2.

[10] R. Al-Dujaily, T. Mak, F. Xia, A. Yakovlev, and M. Palesi, "Embedded transitive closure network for runtime deadlock detection in networks-on-chip," *IEEE Trans. Parallel and Distrib. Syst.*, vol. 23, no. 7, pp. 1205–1215, Jul. 2012.

[11] J. Collet, "A brief overview of the challenges of the multicore roadmap," in *Proc. 21st Int. Conf. Mixed Des. of Integrated Circuits & Syst. (MIXDES)*, 2014, pp. 22–29.

[12] J. Raik, V. Govind, and R. Ubar, "An external test approach for network-on-a-chip switches," in *Proc. 15th Asian Test Symp.*, 2006, pp. 437–442.

[13] C. Concatto, P. Almeida, F. Kastensmidt, E. Cota, M. Lubaszewski, and M. Herve, "Improving yield of torus nocs through fault-diagnoisis-and-repair of interconnect faults," in *Proc. 15th IEEE Int. On-Line Testing Symp.*, 2009, pp. 61–66.

[14] A. Amory, E. Briao, E. Cota, M. Lubaszewski, and F. Moraes, "A scalable test strategy for network-on-chip routers," in *Proc. IEEE Int. Test Conf.*, 2005, pp. 1–9.

[15] Q. Yu and P. Ampadu, "A dual-layer method for transient and permanent error co-management in NoC links," *IEEE Trans. Circuits and Syst. II*, vol. 58, no. 1, pp. 36–40, Jan. 2011.

[16] L. Xie, K. Mei, and Y. Li, "Repair: A reliable partial-redundancy-based router in NoC," in *Proc. IEEE Eight Int. Conf. Netw., Archit. and Storage (NAS)*, 2013, pp. 173–177.

[17] A. Prodromou, A. Panteli, C. Nicopoulos, and Y. Sazeides, "NoCalert: An on-line and real-time fault detection mechanism for network-on-chip architectures," in *Proc. IEEE/ACM 45th Ann. Int. Symp. Microarchitecture*, 2012, pp. 60–70.

[18] Y. Zhang, N. Wu, Y. Wan, F. Ge, and F. Zhou, "Fault-tolerant schemes for NoC with a network monitor," in *Proc. Int. Symp. Commun. and Inf. Technol.*, 2010, pp. 1083–1086.

[19] H. Yi and S. Kundu, "Core test wrapper design to reduce test application time for modular SoC testing," in *Proc. IEEE Int. Symp. Defect and Fault Tolerant of VLSI Syst.*, 2008, pp. 412–420.

[20] Z. Zhang, D. Refauvelet, A. Greiner, M. Benabdenbi, and F. Pecheux, "On-the-field test and configuration infrastructure for 2-d-mesh NoCs in shared-memory many-core architecture," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 6, pp. 1364–1376, Jun. 2014.

[21] N. Caselli, A. Strano, D. Ludovici, and D. Bertozzi, "Cooperative built-in self-testing and self-diagnosis of NoC bisynchronous channels," in *Proc. IEEE Sixth Internal Symp. Embedded Multicore SoCs*, 2012, pp. 159–166.

[22] C. Grecu, P. Pande, A. Ivanov, and R. Saleh, "BIST for network-on-chip interconect infrastructures," in *Proc. IEEE 24th VLSI Test Symp.*, 2006, pp. 1–6.

[23] M. Kakoee, V. Bertacco, and L. Benini, "At-speed distributed functional testing to detect logic and delay faults in NoCs," *IEEE Trans. Comput.*, vol. 63, no. 3, pp. 703–717, Mar. 2014.

[24] M. Botelho, F. Kastensmidt, M. Lubaszewski, E. Cota, and L. Carro, "A broad strategy to detect crosstalk faults in network-on-chip interconnects," in *Proc. 18th IEEE/IFIP VLSI Syst. Chip Conf. (VLSI-SoC)*, 2010, pp. 298–303.

[25] Y. Zheng, H. Wang, S. Yang, C. Jiang, and F. Gao, "Accelerating strategy for functional test of NoC communication fabric," in *Proc. IEEE 19th Asian Test Symp.*, 2010, pp. 224–227.

[26] E. Cota, F. Kastensmidt, M. Cassel, and M. Herve, "A high-fault-coverage approach for the test of data, control, and handshake interconnects in mesh networks-on-chip," *IEEE Trans. Comput.*, vol. 57, no. 9, pp. 1202–1215, Sept. 2008.

[27] M. Herve, P. Almeida, F. Kastensmidt, E. Cota, and M. Lubaszewski, "Concurrent test of network-on-chip interconnects and routers," in *Proc. IEEE 11th Latin Am. Test Workshop*, 2010, pp. 1–6.

[28] C. Grecu, A. Ivanov, R. Saleh, and P. Pande, "Testing network-on-chip communication fabrics," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 12, pp. 2201–2214, Dec. 2007.

[29] A. Strano, C. Gómez, D. Ludovici, M. Gavalli, M. Comez, and D. Betozzi, "Exploiting network-on-chip structural redundancy for a cooprative and schalable built-in self-test architecture," in *Proc. Des., Autom. & Test Eur. Conf. & Exhib.*, 2011, pp. 1–6.

[30] X. Tran, Y. Thonnard, J. Durupt, V. Beroulle, and C. Robach, "Design-for-test approach of an asynchronous network-on-chip architecture and its associated test pattern generation and application," *IET Comput. & Digital Techn.*, vol. 3, no. 5, pp. 487–500, 2009.

[31] L. Chen and T. Pinkston, "Nord: Node-router decoupling for effective power-gating of on-chip routers," in *Proc. IEEE/ACM 45th Ann. Int. Symp. Microarchitecture*, 2012, pp. 270–281.

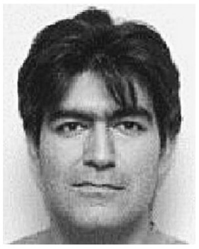
[32] M. Ebrahimi, J. Wang, L. Huang, M. Daneshtalab, and A. Jantsch, "Rescuing healthy cores against disabled routers," in *Proc. IEEE Int. Symp. Defect and Fault Tolerance in VLSI and Nanotechnology Syst. (DFT)*, 2014, pp. 98–103.

[33] *The PASRC Benchmark Suite 2.0.* http://parsec.cs.princeton.edu/, 2009.

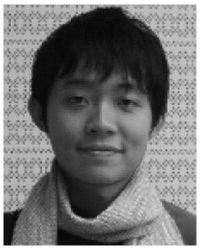
Letian Huang received the BS and MS degrees from the University of Electronic Science and Technology of China (UESTC), Chengdu, where he is currently working toward the PhD degree in 2006 and 2009, in communication and information system. He is also a lecturer of UESTC. His research interests include power adaptive computing and communication, FPGA based heterogeneous computing, signal processing for communication, and mixed signal IC design. His scientific work contains more than 30 publications including books, journal articles and conference papers. He is a member of the IEEE.

Junshi Wang received the BS degree in communication engineering from the University of Electronic Science and Technology of China (UESTC), Chengdu in 2012. He is currently working toward the PhD degree with the Department of Communication and Information Engineering, UESTC. His research interests include reliability of network-on-chip and many-core system. He is a student member of the IEEE.

Masoumeh Ebrahimi received the PhD degree with honours from the University of Turku, Finland. Currently, she is an Academy of Finland fellow at the University of Turku, Finland. She also holds an EU-VINNOVA-MarieCurie fellowship, conducting her research at KTH Royal Institute of Technology, Sweden. Her scientific work contains more than 70 publications including book chapters, journal articles and conference papers. The majority of work has been done in the Networks-on-Chip domain as multicast communication, congestion-aware techniques and fault-tolerant methods. She actively acts as a guest editor, organizer, and program chair in different workshops and conferences in the NoC-related areas. She is a member of the IEEE.



Masoud Daneshtalab is currently a European Marie Curie fellow in the Department of Electronic and Embedded Systems at KTH Royal Institute of Technology, Sweden. He has been appointed as associate editor of *Elsevier Journal of Computers and Electrical Engineering (CAEE)* along with *World Research Journal of Computer Architecture (JCA)*; and in the *Editorial Board of The Scientific World Journal*, *International Journal of Distributed Systems and Technologies (IJDST)*, *International Journal of Adaptive, Resilient and Autonomic Systems (IJARAS)*, *International Journal of Embedded and Real-Time Communication Systems (IJERTCS)*, and *International Journal of Design, Analysis and Tools for Integrated Circuits and Systems (IJDATICS)*. His research interests include on/off-chip interconnection networks, many-core embedded systems, reconfigurable architectures, and neuromorphic computing. He is a member of the IEEE and has published 1 book, 4 book chapters, and more than 150 refereed international journals and conference papers along with more than 80 different co-authors. He is currently in a technical program committee member of different IEEE and ACM conferences, including NOCS, DATE, ASPDAC, ESTIMedia, VLSI Design, SOCC, VDAT, DSD, PDP, ICESS, MCSoC, CADS, EUC, DTIS, NESEA, CASEMANS, NoCArc, MES, HPIN, PACBB, MobileHealth, and JEC-ECC.



Xiaofan Zhang received the BS degree in information engineering from the University of Electronic Science and Technology of China (UESTC), Chengdu in 2013. He is currently working toward the MS degree with the Department of Communication and Information Engineering, UESTC. His research interests include reliability of network-on-chip, parallel and heterogeneous computing.

Guangjun Li received the MS degree from the University of Electronic Science and Technology of China (UESTC), Chengdu, in 1985. Since then, he has been with UESTC. From 1991 to 1992, he was a visiting scholar with RETH Aachen University, Aachen, Germany. He is currently the chair of the Communication Integrated Circuits and Systems Engineering Center of UESTC. He has published more than 60 publications including book chapters, journal articles and conference papers in the area of communication systems, wireless communication networks, SoC and NoC for wireless communication systems.

Axel Jantsch received the dipl ing and dr tech degrees from the Technical University of Vienna, Vienna, Austria, in 1988 and 1992, respectively. He was with Siemens Austria, Vienna, Austria, as a system validation engineer from 1995 to 1997. Since 1997, he has been an associate professor with the Royal Institute of Technology (KTH), Stockholm, Sweden. Since 2000, has been a Docent, and since December 2002, a full professor of Electronic System Design with the Department of Electronic Systems. He has published more than 200 papers in international conferences and journals. He has served on a large number of technical program committees of international conferences, such as FDL, DATE, CODES ISSS, SOC, NOCS, and others, and one book in the areas of VLSI design and synthesis, system level specification, modeling and validation, HW/SW codesign and cosynthesis, reconfigurable computing, and networks on chip. At KTH, he is heading a number of research projects involving a total number of ten PhD Students, in two main areas: system modeling and networks-on-chip. He received the Alfred Schrodinger Scholarship from the Aus-trian Science Foundation while a guest researcher with KTH between 1993 and 1995. He has served on a large number of technical program committees of international conferences, such as FDL, DATE, CODES ISSS, SOC, NOCS, and others. He has been the TPC Chair of SSDL/FDL 2000, the TPC Co-Chair of CODES ISSS 2004, the General Chair of CODES ISSS 2005, and the TPC Co-Chair of NOCS 2009. From 2002 to 2007, he was a subject area editor for the *Journal of System Architecture*. He is a member of the IEEE.