Efficient Design-for-Test Approach for Networks-on-Chip

Junshi Wang, Masoumeh Ebrahimi, Letian Huang, Xuan Xie, Qiang Li, Guangjun Li, Axel Jantsch

Abstract—To achieve high reliability in on-chip networks, it is necessary to test the network continuously with Built-in Self-Tests (BIST) so that the faults can be detected quickly and the number of affected packets can be minimized. However, BIST causes significant performance loss due to data dependencies. We propose EsyTest, a comprehensive test strategy with minimized influence on system performance. EsyTest tests the data path and the control path separately. The data path test starts periodically, but the actual test performs in the free time slots to avoid deactivating the router for testing. A reconfigurable router architecture and an adaptive fault-tolerant routing algorithm are proposed to guarantee the access to the processing core when the associated router is under test. During the whole test procedure of the network, all processing cores are accessible, and thus the system performance is maintained during the test. At the same time, EsyTest provides a full test coverage for the NoC and a better hardware compatibility comparing with the existing test strategies. Under the PARSEC benchmark and different test frequencies, the execution time increases less than 5% at the cost of 9.9% more area and 4.6% more power in comparison with the execution where no test procedure is applied.

Index Terms—network-on-chip, non-blocking testing, reliability monitoring, reconfigurable router architecture, adaptive routing algorithm, built-in self-test.

1 Introduction

By the rapid scaling of many-core System-on-Chip, the computational resources are not as crucial as before. Instead, a communication-centric design became a challenge where a scalable communication structure such as Network-on-Chip (NoC) is demanded [1]. NoC is typically not perceived as contributing to the system performance directly in the way processing cores do. However, it should not be the performance bottleneck and should not slow down the operation of the cores. It is expected that NoC provides a fast and reliable infrastructure for delivering packets between the cores.

As technology advances, the design of integrated circuit faces serious wear-out issues, including Electro-migration (EM) [2] as well as Hot Carrier Injection (HCI), Negative Bias Temperature Instability (NBTI) and Time Dependent Dielectric Breakdown (TDDB) [3]. Wear-out slowly causes variations on shapes or parameters of the components which results in the misbehavior or component breakdown in a long run. The failures caused by wear-out first appear as the intermittent faults which eventually lead to permanent errors. NoCs cannot escape from these trends and due to the faults in the data path and/or the control path, data might be delivered incorrectly, packets might be misrouted or lost, and the integrity of packets might be damaged [4]. NoCs usually offer natural redundancy with the potential for fault tolerance. Area, power and network performance are also critical factors for a reliability design.

There are four main steps to design a reliable NoC: fault detection, diagnosis, reconfiguration, and recovery [5]. Fault detection methods can be implemented in different layers of systems [6]. Although fault detection methods like Error Correction Codes (ECC) [7], [8] and Assertion-based methods [4] can detect errors when flits go through the components, Built-in Self-Test (BIST) has the advantages of fine-grain diagnosis and high fault coverage. BIST can be triggered by fault detection methods or a timer [9]. Fault-detection delay is defined as the difference between the time a fault is occurred and the time it is detected. Such delay is determined by the frequency in which the BIST is triggered so that a more frequent BIST means a shorter detection delay. Consequently, more frequent BIST results in a smaller amount of fault flits because proper actions might be taken as soon as a fault is detected. Figure 1 shows the relationship between the test frequency and the flits' fault rate. Let us assume that intermittent faults occur independently and the flits' content is changed when packets are delivered through a faulty component. By applying the fault model described in [10] and the parameters provided in [11], two scenarios are examined: $FM_1 = \left(10^{-6}, 0.5, 10^{-5}\right)$ and $FM_2 = \left(10^{-6}, 0.5, 10^{-6}\right)$, which means on average the faults occur every 10^6 cycles, last for 10^5 (in FM_1) or 10^6 (in FM_2) cycles, and trigger the errors by the probability of 0.5. The flit injection rate is 0.01 flit/cycle. Further, we assume that BIST is able to detect all errors with the fault coverage of 100%. As shown in Figure 1, without any test, the fault rate is about 4.6‰for FM_1 and 2.2‰for FM_2. By including BIST, the fault rate reduces when shortening the test period (increasing the test frequency). With the test period of 20K cycles, the fault rates have been reduced by

- *Junshi Wang was with University of Electronic Science and Technology of China, Chengdu, China. He is now with Beijing Zhaoxin Electronic Technology Co., Ltd., Beijing, China.*
- *Letian Huang, Xuan Xie, Qiang Li, Guangjun Li are with University of Electronic Science and Technology of China, Chengdu, China.*
- *Masoumeh Ebrahimi is with KTH Royal Institute of Technology, Stockholm, Sweden.*
- *Axel Jantsch is with Technische Universität Wien, Vienna, Austria.*
- *Corresponding author: Letian Huang, E-mail: huanglt@uestc.edu.cn.*

Manuscript received April 19, 2005; revised August 26, 2015.

two orders of magnitude, which are around 0.03‰ for both scenarios. Therefore, increasing the test frequency helps in detecting faults faster and thus reducing the fault rate if a proper action is taken.

However, the NoC's throughput is influenced significantly by BIST because the circuit under test should be isolated from the network. The effect of BIST can be seen both at the network and system level. At the network layer, BIST disturbs the connectivity and integrity of NoCs (e.g. by disabling a router). At the system layer, the processing core connected to the router-under-test (RUT) is isolated from the network and disconnected from the system. Therefore, processing cores have to wait longer to get the necessary data. As a result, increasing the test frequency reduces the application throughput. It is also reported in [12] that the average packet latency of PARSEC benchmarks rapids up as the test period reduces. Hence, suitable mechanisms are required to increase the test frequency while maintaining the system throughput.

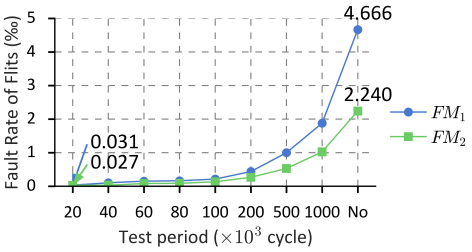

Fig. 1. Flits fault Rate under periodic BIST with different test periods.

To achieve the goals of full test coverage, ignorable performance loss and high test frequency, we propose a novel strategy, EsyTest, shortly described as follows:

1) The test strategy is divided into two parts: the data path test and the control path test. The data path test focuses on the errors introduced by the links, memories, and registers. The control path test deals with the errors introduced by the state machines, flow control units, routing calculation units, and allocators. As a result, all links and routers are covered by EsyTest.
2) During the data path test, the test vectors are packaged into the test packets. Instead of injecting test packets immediately after the test timer triggers, the test packets are injected to the links and routers at free time slots within a test window time. This strategy prevents increasing packets' latency as data packets are delivered normally while free slots are utilized to inject the test packets.
3) For testing the control path, the aforementioned components are isolated from the network by wrappers. Meanwhile, the connections of data paths are active but set to the fixed configuration. In other words, the local port will be statically connected to the east (or west) port so that the processing core connected to the RUT can still access the network. North and south ports will be directly connected to each other. To support this new configuration, an adaptive routing algorithm is proposed which allows packets to use these channels for the packet delivery during the test. Since RUTs are not totally disconnected from the network, the performance does not drop when the network is tested with a high frequency.

To sum, in EsyTest all processing cores including those connected to RUTs are active all the time with their full capacity. Thereby, EsyTest minimizes the influence of test on the system performance. It is also able to provide a test coverage of 100%.

The remainder of this paper is organized as follows: Section 2 summarizes the published works on BIST for NoCs. The EsyTest strategy is explained in Section 3. In Section 4 and Section 5, the router architecture and routing algorithm to support EsyTest are proposed. Section 6 illustrates and discusses the simulation results. Finally, the conclusion is given in the last section.

2 Related Work

In this section, we first review some published works on the NoC test methods and categorize them into assertion-based methods and BIST. Then we discuss the works on the impact of the test procedure on the network performance.

2.1 Test Methods

Fault detection methods for NoCs always drive the circuit under test with specific test vectors. By observing and analyzing the responses, errors can be detected and diagnosed.

Some designs [7], [8] use the data packets as test vectors and apply error detection/correction codes (EDC/ECC) to detect faults in the data path (i.e. links and buffers). Hamming codes and parity check codes are among the most commonly used methods, but their capabilities to detect/correct faults are limited to a few bits. Two different assertion-based methods are proposed in [4] and [13] to detect errors in the data and control path. They employ some monitors and checkers to watch the values of input and output signals. If signals do not match certain rules, alerts are raised. The major advantages of EDC and assertion-based fault detection methods are the small area overhead and simple implementation. However, both of these methods are unable to diagnose the causes and location of faults. Moreover, EDC and assertion-based fault detection methods cannot detect all faults as some faults need special trigger conditions which are not easy to be satisfied by these methods.

Built-In Self-Test (BIST) is another kind of commonly used test method, that is based on injecting test vectors and comparing the received responses with the expected ones. BIST consists of more complex components, but it can detect most of the physical failures. The unit under test will be temporarily isolated from the rest of the network by wrappers. Wrappers are in the form of multiplexers and demultiplexers that are located in the inputs and outputs of the unit under test.

The size of the unit under test varies from a single component (e.g. an arbiter) to the whole router or even the entire network. [5] investigates wrappers that isolate one or several components inside a router for a fine-grain diagnosis of special components. [14] proposes a method to test the

input buffers in the routers while in [15] only the links are examined by the test method. In [16] and [17], the wrappers isolate the output register, the transmission link, and the input buffer of the neighbor router.

Commonly, the wrapper isolates an entire router plus some components from the neighbor routers. [18] presents a wrapper for the whole router based on IEEE 1500. The wrappers in [12] and [19] isolate the router along with the neighbor ports. In [12], test packets collect faults on the paths across the RUT. In [19], the test packets collect faults on the paths between neighbor routers and the RUT.

In some designs, wrappers cover bigger units than a router. For example, the wrappers in [20] and [21] cover a 2×2 network area to detect the circuit faults in both the data and control path. The path for delivering test packets can be organized through the whole network [22], [23]. By analyzing the faults captured by test packets, the faults can be located in the granularity of routers and links. [24] and [25] proposed different organizations of test packets based on multicast routing algorithms to reduce the duration of test procedures.

In this work, to test the data path, EsyTest uses wrappers around one router and its belonging links. To test the control path, wrappers isolate individual components (i.e. the routing calculator, virtual channel arbiter, switch arbiter, and buffer controller) but test them in parallel.

2.2 State-of-the-Art

In recent years, researchers and designers make special attention to intermittent faults caused by process variation and wear out. To capture intermittent faults, the components should be tested periodically, which means periodic test runs while applications use the network for packet delivery. Periodic on-line BIST [9] is very efficient to identify the faults that appear only under specific environmental conditions.

In on-line BIST, there is a contention between test vectors and data communication for receiving resources. Traditional test wrappers do not allow data communication during the test procedure so that both the router and the connected processing core are disconnected from the network. The disabled part interrupts the normal traffic flow in the network which causes heavy traffic congestion [12].

Most published papers have not clearly pointed it out how and when the BIST method should be triggered. [26] presents a token-based approach to determine the test sequence. Only one router is chosen at a time, and routers are tested sequentially without interrupt.

Moreover, only a few work have discussed the influence of the test procedure on the NoC performance. [12] proposed a test method based on BIST with wrappers to disconnect the RUT. The token-based method is applied to control the test sequence. Further, between the test procedure of routers in sequence, a test interval time is considered to allow the network to recover from the congestion and avoid a significant performance drop. Based on simulations, [12] reports that when the test interval time is less than 2,000 flits, the average latency raises up by 4 times compared to the average latency under the test interval time of 20,000 flits. An interval of 10,000 flits is considered reasonable. However, in [12], a processing core has to be disconnected along with the RUT, which may affect the system functionality. In addition, packets are blocked in the input ports of the RUT, which creates hotspot and congestion in the network.

Liu et al. [27] observe that the traffic load on the links is not heavy. Based on this observation they suggest that the test can be operated while a link is free. As a result, the reliability of links can be monitored while the network is functioning normally and thus the test does not affect the NoC performance. The major drawback is that this method can only test the links and it cannot be applied to test routers.

TARRA [28] addresses the shortage of [12] by employing bypass channels within the routers as shown in Figure 2. During the test procedure, packets can be still delivered through the RUT using bypass channels. The advantages of TARRA are as: 1) the packets are not blocked during the test; 2) the processing core connected to the RUT can still access the network. As a result of using bypass channels, the influence of the test procedure on performance is reduced significantly. In addition, by suggesting a proper test sequence, multiple routers can be tested at the same time and the test frequency can be increased.

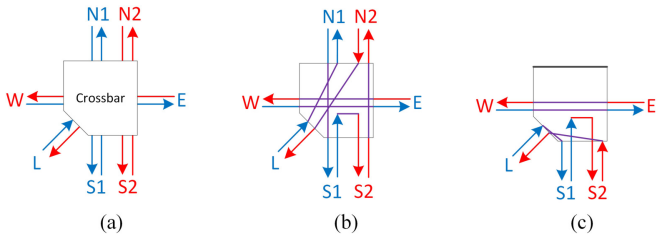

Fig. 2. Router architecture in TARRA [28]. (a) A router without test. (b) A router during the test procedure using bypass channels. (c) A router on the northern border during the test procedure using bypass channels.

However, TARRA has some limitations as: 1) it cannot test global links between routers as packets use the global links during both the normal and test phases; 2) the router architecture must have two physical channels on the north-south directions. During the test procedure, one channel is used for bypassing the router in the vertical direction, and the other one is used for accessing the connected processing core. Due to the requirement of physical connections, TARRA needs adaptation to different router architectures.

EsyTest is designed for the periodic test, and the test procedure of one router repeats on predetermined intervals. Different from TARRA which relies on additional resources such as bypassing channels, EsyTest takes advantage of the available time and hardware resources in a router to perform the test. In EsyTest, no bypassing channels are used but rather the components of the data path are utilized to deliver packets during the test. As a result, the influence of the test procedure on performance is limited to similar degree as TARRA. Meanwhile, EsyTest can test all components in a router with the test coverage of 100% and hardware compatibility to different router architectures. Moreover, using the proposed test sequence, more routers can be tested simultaneously. EsyTest provides mechanisms to test all components of NoCs efficiently. Note that the design of proper test vectors is outside the scope of this

paper.

Table 1 summarizes the test coverage of TAFD, TARRA, and ESYTest. As discussed, TAFD covers the links between routers while TARRA covers the routers but does not cover the links. ESYTest takes advantage of both free-slots on the data path and the reconfiguration router architecture. As a result, ESYTest can test all circuits in NoC with a negligible impact on the performance.

TABLE 1
Test coverage of test strategies

Components		TAFD [27]	TARRA [28]	ESYTest
Router	Input buffers	×	√	√
	Crossbars	×	√	√
	Output registers	×	√	√
	Routing calculators	×	√	√
	State machines	×	√	√
	Arbiters	×	√	√
Links between routers		√	×	√

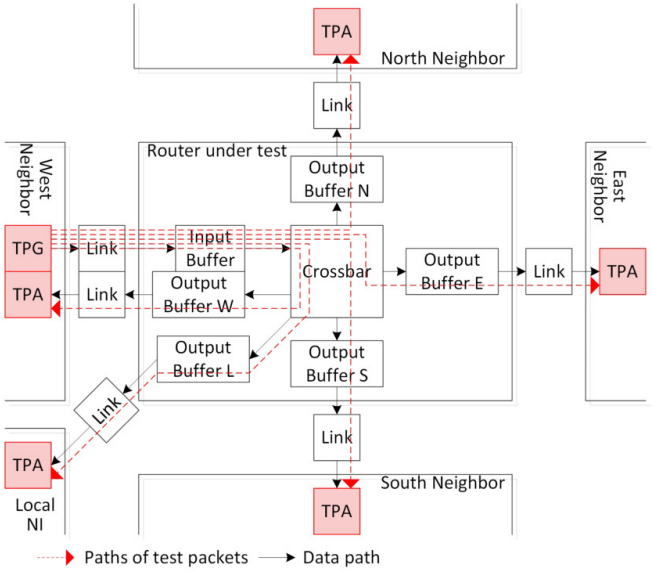

Fig. 3. BIST for the data path test.

3 Test Strategy

To minimize the impact of the test procedure on the system performance, EsyTest deploys different test strategies for the data path and the control path. Moreover, EsyTest proposes a special test sequence for periodic BIST that increases the test frequency significantly.

3.1 Data Path Test

In the data path, wrappers isolate one router and its links from the other parts of the network. As shown in Figure 3, the Test Packet Generator (TPG) is located in the output buffer of the neighboring router while the Test Packet Analyzer (TPA) is placed in the input buffer of the neighboring router. A TPG generates test packets that pass through the RUT and reach the next neighbor router. The TPA checks the content of the test packets in the receiving neighbor.

A common test approach is that the test packets are injected directly when the test procedure starts. During the test, the transmission of data packets is stopped which increases the packets' delay. This situation is shown in Figure 4(a) where an orange and white box refer to a test and data flit, respectively. Increasing the number of test vectors exacerbates the problem and worsens the latency.

One alternative approach is to use the free time slots to perform the test. Free time slots are the times when there is no request on a channel, and thus the channel can be used by test packets. Table 2 reports the flit injection rate and the link utilization rate of the PARSEC benchmarks [29]. The test environment is the same as in Section 6. The link utilization is defined as the ratio of the time a link is occupied by flits over the total execution time. As reported in Table 2, the maximum link utilization is around 19% for canneal, meaning that 80% of time slots are free and can be used for testing. In the swaptions, the link utilization is as low as one occupied link per thousand cycles. These results show that in the real environment, the data path is idle for most time. These free slots can be used to run the test without affecting the performance.

TABLE 2
Flit injection rate and link utilization under different PARSEC benchmark programs

Benchmark program	Flit injection rate (flit/cycle)	Link utilization	
		Max	Min
blackscholes	0.898	0.044	6.502e-05
canneal	1.582	0.190	2.517e-06
dedup	0.234	0.042	1.328e-07
fluidanimate	2.976	0.088	8.631e-05
freqmine	0.017	0.004	3.864e-07
raytrace	0.640	0.115	7.859e-07
streamcluster	0.345	0.065	4.346e-06
swaptions	0.024	0.001	2.294e-07
vips	0.053	0.014	2.469e-07

Ideally, if test vectors are only injected in the free slots, the test procedure does not influence the system performance at all as shown in Figure 4(b). However, it should be guaranteed that a test packet (with 5 flits in this example) is completely injected before serving any new packet. Therefore, the data packets still need to wait for the test packet for a short time, as illustrated in Figure 4(c). In this example, the packets' delay depends on the size of test packets. Thereby, smaller test packets can reduce the latency of data packets and limit the performance drop. The test packets are given the same priority as data packets in the RUT.

Traffic load on different routers is not often the same. The test procedure on busy routers takes longer than the ones with a lighter load as there are fewer free time slots to be used for the test. To limit the duration to test different routers within a narrow range, one deadline should be considered. As shown in Figure 4(d), a test procedure is divided into two phases denoted as Free-Slot and Block. In the Free-Slot phase, test packets are only injected during the free slots. The time to inject all test packets is variable depending on the traffic load. The maximum duration of the Free-Slot phase is defined by T_{free}. In the Block phase, the remaining test packets are injected, and the data traffic

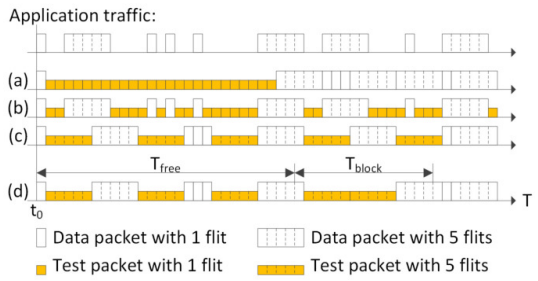

Fig. 4. The sequence of data and test packets under different test strategies during the data path test. The data path test procedure is triggered at t_0. (a) the baseline strategy. (b) single-flit test packets are injected at free time slots. (c) multiple-flit test packets are injected at free time slots. (d) the EsyTest strategy.

is blocked. In an ideal case, no data packets compete with test packets. In the worst case, the data path is so loaded in which no test packets can be injected during the Free-Slot phase.

T_{block} should be long enough to inject all the test packets into the data path considering the worst case scenario where no packets are injected into the network during the Free-Slot phase due to a heavy traffic. In this case, T_{block} is determined by the total number of flits in test packets. T_{free} defines the test frequency where longer(shorter) T_{free} means more(less) available free slots to inject test packets. By choosing T_{block} and T_{free}, designers trade-off between the applications' execution time and the test intervals.

3.2 Control Path Test

The control path includes buffer units' controller, state machines, routing calculators and allocators. Testing the data path can detect some errors on the control path but not all of them.

To test the control path and diagnose all the faults, test vectors should be directly injected to the circuit-under-test as shown in Figure 5. For this purpose, wrappers are used to disconnect the control path from the data path by isolating state machines, buffer controllers, flow controllers, routing units and allocators from the other components of the router. The normal pipeline cannot operate, and the RUT cannot route any packets. However, to reduce the impact of the RUT on the network latency, the connections of the data path will be fixed to enable packets passing through the RUT in predetermined directions. For this purpose, the port connected to network interface (NI) will be directly connected to the east port (or connected to the west port if the router is located in the eastern border), allowing the local core to send/receive packets to/from the other cores. The north port is directly connected to the South port and vice versa. Thereby, the data path of the RUT can be seen as some short-cut connections. The packets on W/E direction should turn around the RUT. Due to the fact that the control units are disabled, virtual channels cannot be utilized during the test.

The control path test includes three phases as Emptying, Testing, and Recovery, shown in Figure 6. First, the buffers of the RUT should be emptied from normal packets during the Emptying phase. Then, the control path is disconnected

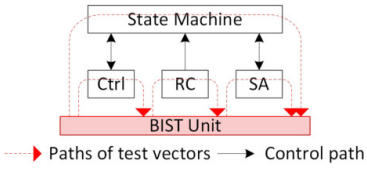

Fig. 5. BIST for the control path test.

from the data path by wrappers, and the fixed connections of the data path are established. The RUT enters the Testing phase where test packets are injected and the control path is tested. Meanwhile data packets pass through the RUT using fixed connections. When the Testing phase is over, the buffers of the RUT are emptied from the test packets during the Recovery phase. At the end of the Recovery phase, the control path takes control of the RUT, back to the normal mode.

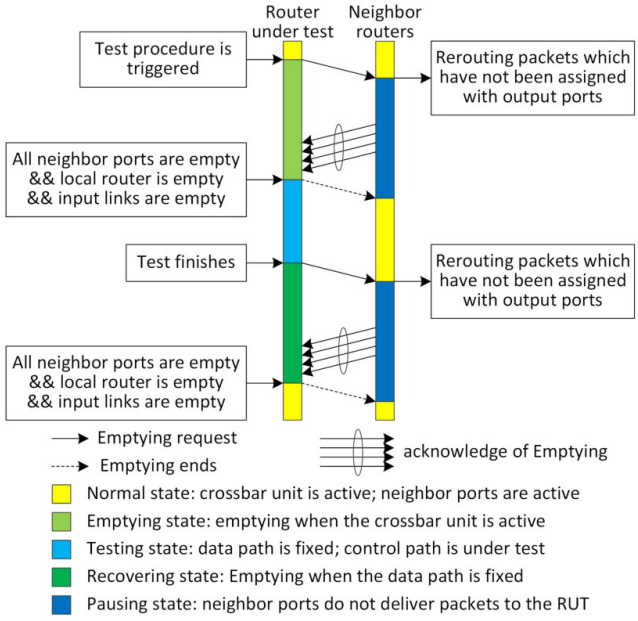

Fig. 6. Test state machine and control over the transition phases. The FSM of the router-under-test and the neighbor routers are illustrated using blocks in the left and right side of the figure, respectively. The handshake signals between the router-under-test and the neighbor ports are shown in the center.

Under the wormhole flow control, the direction of packets is determined by the head flits; and the body flits follow that direction. During the emptying phase, if the direction of a packet has already been decided, the packet has to be completely delivered to the next router before entering the testing phase. This is to avoid dropping any packets. Some flits of one partially delivered packet are already transferred to the router that is going to be disabled soon for the test (RUT). The rest flits are still in the neighbor routers. Thereby, these incomplete packets should be processed first in RUT. Meanwhile, neighbor routers stop sending any new complete packets to the RUT. Therefore, the RUT does not need to distinguish between new or partially delivered packets. RUT completes the delivery of partially delivered packets but it will not receive any new packets from neighbors as

neighbors are asked to stop sending new packets.

Figure 6 shows the protocol between RUT and neighbor routers to guarantee a complete delivery of incomplete packets. At the start of the Emptying and Recovery phases, RUT sends out Emptying requests to the neighbor routers as well as the local port. Upon receiving this request by the neighbor routers and the local port, they temporarily stop sending any new packets to the RUT but proceed to deliver all packets which have already been partially delivered. This phase is called "Pausing" phase in Figure 6. After delivering the packets on-the-fly, an Emptying acknowledgment is sent by each neighbor as well as the local port to the RUT. In another case, a packet is located in the input buffers of the neighbor router, but it has already been decided that it should be routed through the RUT. This kind of packets is rerouted first because the architecture of the RUT is reconfigured. If these packets are still ahead to the RUT after rerouting, they are blocked for a few cycles in the input buffers until the new architecture is configured.

To sum up, incomplete data packets are routed in the Emptying phase similar to the normal functioning mode. During the testing and recovery phases, although the control path is under test, the data path is used in the North-South and West/East-NI directions, driven by predefined fixed control signals (to the crossbar, buffers, etc.). In this way, the injection of new data packets is only stopped for few cycles during the Emptying and Recovery phases. A routing algorithm is proposed in Section 5 to support these packets in the neighboring routers to successfully deliver them from/to the router under test. As a result, the impact of the test procedure is small and ignorable. This observation is confirmed in the experiments and discussed in the result section.

3.3 Timing control of test procedure

Figure 7 describes the test procedure of one router. The test procedure contains five phases and the total period equals the sum of the duration of these five phases. The duration of Free-Slot phase T_{free}, Block phase T_{block} and Testing phase T_{test} is predetermined by the test strategy. The duration of Emptying phase D_{empty} and Recovery phase $D_{recovery}$ is related to the network behavior. If the network is more congested, the D_{empty} and $D_{recovery}$ last longer. So, under the zero-load latency, the lower bound of the test time is $T_{free} + T_{block} + T_{test}$.

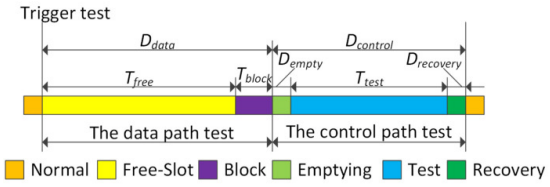

Fig. 7. Timing of test procedure.

The test procedure is triggered by a test interval timer and repeated with a test frequency of TIT (Test Interval Time). Different routers start testing at different times, which is controlled by the initialization value (TIV) of the test interval timer (Figure 8(b)).

It is worth noting that the data path test of one RUT should not overlap with the control path test of the neighbor routers. For example in Figure 8, when Router 4 is under test, Routers 0, 5 and 8 must not be under test. To avoid this overlap, the status of the neighbor routers is checked prior starting a test procedure. EsyTest introduces a four-group test sequence and also sets some conditions on TIT and the duration of different phases to avoid the overlap.

Figure 8(a) shows an example of the four-group test sequence, which ensures the minimum distance of 2 hops between RUTs. For example, Group 0 contains router 0, 2, 8 and 10. None of them is the neighbor of each other. Four groups are tested in a loop from Group0 to Group3. Within each group, routers are tested in sequence from top-left to bottom-right as illustrated in Figure 8(b). The test sequence in Figure 8(b) is:

$$\underbrace{0,2,8,10}_{Group0},\underbrace{1,3,9,11}_{Group1},\underbrace{4,6,12,14}_{Group2},\underbrace{5,7,13,15}_{Group3}$$

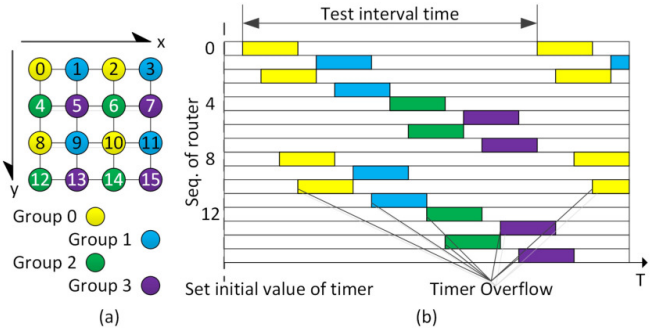

Fig. 8. An example of the four-group test sequence. (a) The division of groups. The number in circles is the router's id. (b) Gantt chart.

The routers belong to one group form a subnetwork with the size of $x_g \times y_g (g = 0, 1, 2, 3)$. x_g and y_g are determined by the network size as:

$$x_g = \begin{cases} x/2 & x \text{ is even} \\ (x+1)/2 & x \text{ is odd, g=0 or 2} \\ (x-1)/2 & x \text{ is odd, g=1 or 3} \end{cases} \quad (1)$$

$$y_g = \begin{cases} y/2 & y \text{ is even} \\ (y+1)/2 & y \text{ is odd, g=0 or 1} \\ (y-1)/2 & y \text{ is odd, g=2 or 3} \end{cases} \quad (2)$$

where x and y are the network dimensions. Based on this formulation, in Figure 8(a), the subnetwork size of each group is 2×2.

Let Router A start the test at time 0 and finish it at $t_1 = t_{test}$ and Router B is the first router in sequence that should not be under test during the test procedure of Router A.

As the example in Figure 8(a), the router 2 (Group 0) can be tested together with routers 0, 8, and 10 but the test of router 2 should be terminated before starting the test of the router 1 (Group 1). Extending this principle, the maximum number of routers that can be tested together is $\min\{x_g y_g - 1\}$. The testing of Router B starts at $t_2 = TIT \times \frac{\min\{x_g y_g - 1\}}{N_R}$, where N_R is the number of routers in the network. The router B's test must happen after the

router A's test, so that $t_2 > t_1$. Therefore, the lower bound of TIT can be obtained by:

$$TIT_l = (T_{free} + T_{block} + T_{test}) \times \frac{N_R}{\min\{x_g y_g - 1\}} \quad (3)$$

As Equation (3) shows, EsyTest can increase the test frequency which is an indicator of a better performance. For example, in a 10×8 mesh network, each group contains 20 routers which form a 5×4 subnetwork ($x_g = 5$ and $y_g = 4$). The maximum number of routers under test at the same time is $\min\{x_g y_g - 1\} = 19$. Because the network contains $N_R = 80$ routers, TIT should be larger than:

$$\begin{aligned} TIT_l &= (T_{free} + T_{block} + T_{test}) \times \frac{80}{19} \\ &= (T_{free} + T_{block} + T_{test}) \times 4.21 \end{aligned} \quad (4)$$

The presented four-group test sequence allows testing around one-forth of routers at the same time while only $x/2$ routers can be tested simultaneously as presented in TARRA [28]. More routers under test means a higher test frequency.

Moreover, the test impact on the performance is minimized by applying EsyTest. In traditional test procedures, packets are blocked during the whole test run that leads to the creation of hotspots and congestion in the network. As a result the next test procedure cannot be started until the network backs to its normal condition. This issue has been addressed in EsyTest by blocking packets only for few cycles during the Emptying and Recovery phases. EsyTest does not lead to hotspots around the router under test and does not impact on network performance significantly. Thus, the test frequency can increase so that the test procedures on different routers can run in parallel following specified test sequence. This statement is confirmed in the result section by observing a negligible performance loss in the case of multiple RUTs.

4 Design-for-test Router Architecture

Each router has a Test Control Unit (TCU) to control the test procedure such as activating/deactivating components, enabling/disabling wrappers, and triggering the timer. The TCU controls the timer for the test interval time and the phases of the test procedure.

4.1 Test wrapper for the data path

In EsyTest, the global links are also included into the test wrapper of the RUT. Test packets are generated in TPGs, located in output ports of neighbor routers, and injected into the data path (Figure 9). In the data path, a test packet goes through the global link of the input port, the input buffer, the crossbar and the global link of the output port. In detail, the switch allocator receives the requests from TPGs and determines the granted port. Except for the Free-Slot and Block phase, the switch allocator does not give any grant to TPGs at all. During the Free-Slot phase, the switch allocator gives the lower priority to the requests from TPGs. In other words, the TPG can access an output port and inject the test packets only if there is no request to this port from the input ports. The test packets are injected continuously as long as there is no request to that output port, or all the test packets are injected. On the other hand, during the Block phase, a higher priority is given to the requests from the TPG in order to inject all test packets in the given test period and to meet the test frequency. The matrix arbiter can easily apply the adjustment of priority [30].

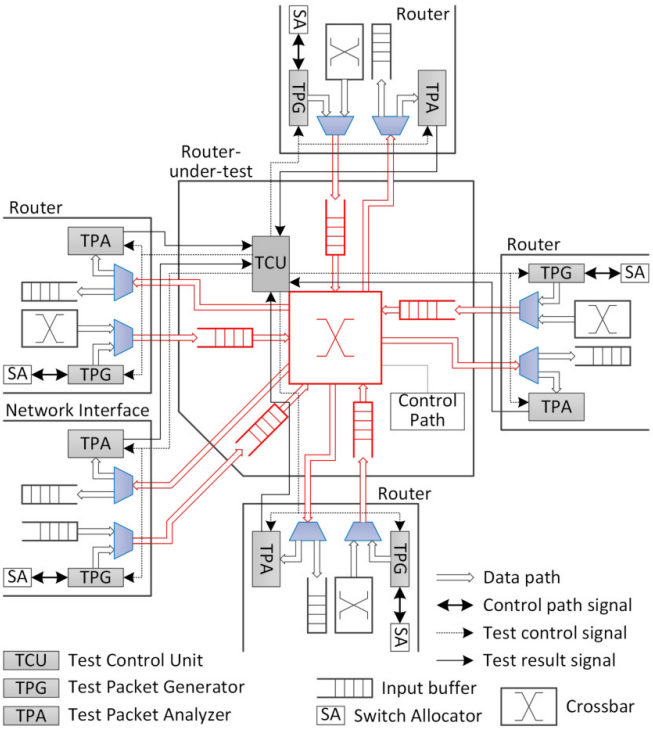

Fig. 9. Test wrapper of the data path. components in red color are under test. Wrapper components are marked by blue color.

TPAs are implemented in the input ports to check the responses from the RUT and to detect errors. TPAs send the results back to the TCU to activate a proper fault-tolerant method. A `TEST` bit is added to the head flit in order to distinguish between test and data packets. Demuxes switch between input buffers and TPAs by the `TEST` bit. Handshake signals are driven by the TPA as well to release flow control signals to the RUT.

4.2 Test wrapper for the control path

The control path includes the state machines, buffer controller, routing unit, and switch allocator. As shown in Figure 10(a), the control path collects the necessary information from the data path by checking the status signals (e.g. buffer full/empty and head flit) and controls the data path by control signals (e.g. buffer read/write and crossbar request).

BIST test units are integrated inside routers. TCU sends out the enable signals to the BIST test units to start the test. When the router is not under test, state and control signals directly cross the test wrappers (Figure 10(c)) so that the control path takes the full control over the data path. The connections during the test procedure are shown in Figure 10(d). The state signals are driven by BIST test units to inject test vectors. The control signals from the control path are also connected to BIST test units so that BIST test

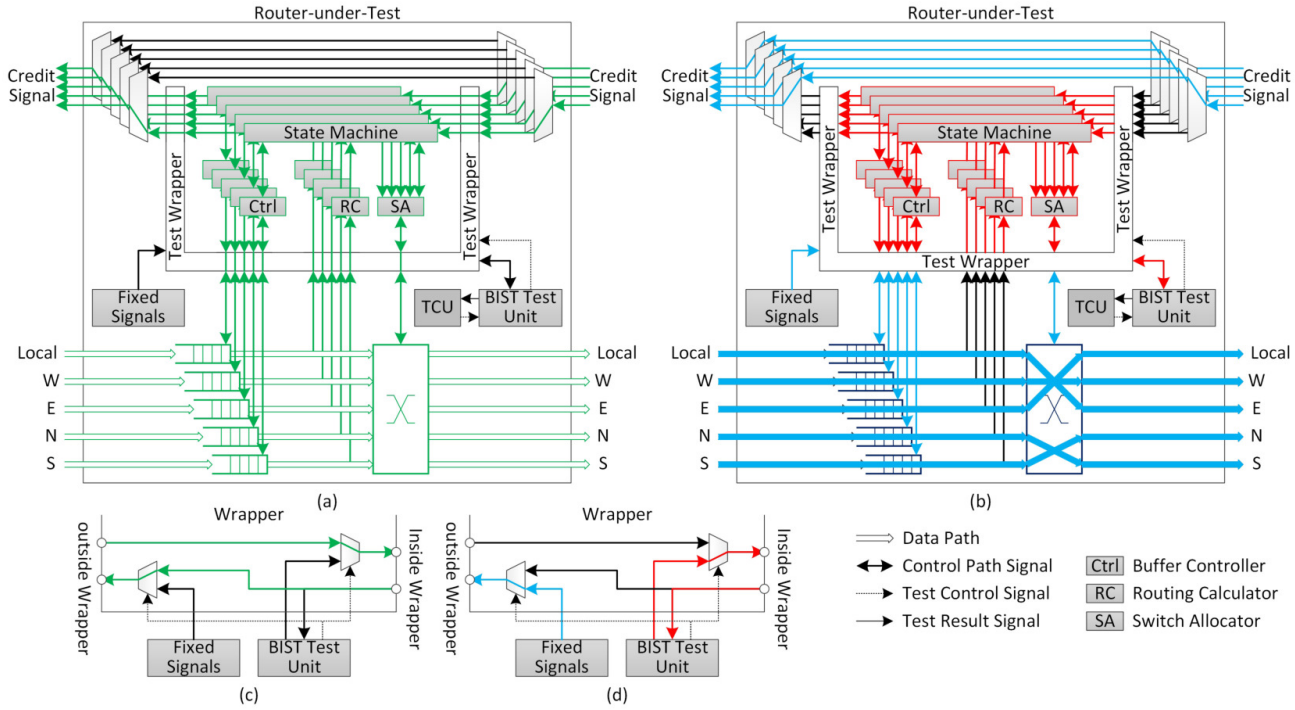

Fig. 10. Test wrapper of the control path. (a) and (b) show the architecture of test wrappers. (c) and (d) show the circuit within the test wrappers. (a) and (c) highlight the connections for data transmission, which are colored by green. (b) and (d) highlight the connections for test. The red components are under test while the data path and credit signals controlled by fixed signal are marked by blue color.

units can check the responses from the control path and thus diagnosing faults.

During the control path test, the data path is controlled by predetermined fixed signals as shown in Figure 10(b). The buffers' pointers are constant, and the write and read operations are always enabled. Thereby, the buffer units pipe out the flits through one-flit registers. The crossbar is also fixed because the control signals are constant. Therefore, the RUT becomes two shortcut paths: north-south, local-west/east. The credit signals and handshake signals are directly connected from the output ports to the input ports to maintain the correct behavior of the flow control.

4.3 Synchronous Signals

This section introduces the signals that are used for the synchronization between the RUT and the neighbor routers. Direct neighbor routers are the north, south, east and west neighbors. Indirect neighbor routers are the routers at the corners (NW, NE, SW, and SE).

A TCU sends two-bit signals `TestPhase` (TP) to neighbor routers to control the test procedure. Different values of the two-bit signal are as: "`00`" normal functioning mode; "`01`" Free-Slot phase in the data path test; "`11`" Block phase in the data path test; and "`10`" the control path test. TPAs send results (TR) back to the TCU, using the test vectors.

During Emptying and Recovery phases, RUTs request their neighbors to clear the output ports through handshake signals `EmptyRequest` (ER) and `EmptyAck` (EA). ER requests the neighbor router to empty the port connected to the RUT while EA acknowledges the receipt of the empty signal from the neighbor router.

To limit the influence of the RUT on the network performance, this work defines a 3×3 distinct region with an adapted routing algorithm. The default adaptive routing algorithm applies to all the areas that are outside of the distinct region. Each direct and indirect router should know the RUT status. A router sends/receives `DirectNeighborStatus`(DNS) and `IndirectNeighborStatus`(INS) signals through two unidirectional wires (four wires in total). DNS signals transfer the status of the local router to the neighbor routers and vice versa. INS signals transfer the status of a neighbor router to the next neighbor router as shown in Figure 11. For example, the status of the east neighbor router is transferred to the north neighbor router as the red line in Figure 11. The routing algorithm based on the distinct region is described in Section 5.

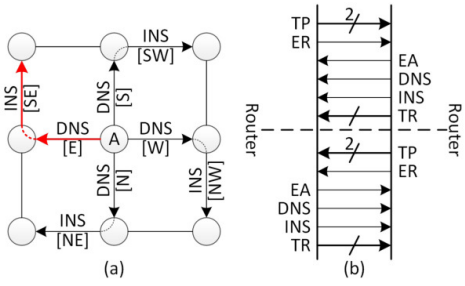

Fig. 11. (a) Propagation of DNS and INS signals. (b) Synchronous signals between two routers.

5 Adaptive Routing Algorithm for Test

An adaptive routing algorithm to support the reconfigurable router architecture is proposed in this section. The algorithm tries to maintain the network connectivity and maximize the performance while the routers are under test.

Similar to DyXY, the proposed routing algorithm is fully adaptive, and packets can be routed through the shortest paths between the source and destination routers. Unlike DyXY, the proposed algorithm is able to tolerate RUT with fixed connections in the crossbar during the control path test. For simplicity, we call the RUT with fixed crossbar **FC-RUT**.

As shown in Figure 12(a), each router has five ports: one port per neighbor router and one port connected to the processing core. Each port has one physical/virtual channel except the north and south ports which have two channels. The extra channels on north and south ports are utilized to avoid deadlock in the adaptive routing algorithm. Because the processor connected with the RUT has to access the network through the east neighbor router, we define the east neighbor of RUT as the ladder router (Figure 12(b)) unless the RUT is on the eastern border in which the west neighbor will be the ladder router (Figure 12(c)).

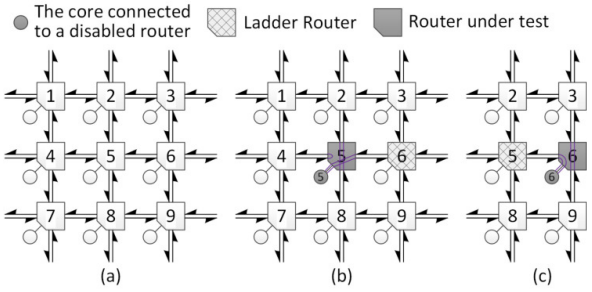

Fig. 12. The baseline topology. (a) without RUT. (b) fixed crossbar of RUT during the control path test- the general case. (c) fixed crossbar of RUT during the control path test- if RUT is on the eastern border.

5.1 Routing rules

This section introduces the routing rules applied in EsyTest. We refer to Figure 13 and the corresponding lines in the pseudo-code (Appendix) as we explain.

The status of FC-RUT is propagated to the direct and indirect neighbors within a 3×3 distinct area with the FC-RUT in the center. The routing algorithm should be able to deliver packets to all destinations inside and outside the distinct region (Figure 13). Packets choose a direction based on the following rules. Note that each subfigure may be applied to more than one rule.

RULE 1: This is the default rule when the current node is not in the distinct region. Packets choose the shortest paths to the destinations (line 1-3). Among possible directions, packets choose the one with less congestion (e.g. more free slots in the input buffer of the neighbor router). To prove deadlock freedom, we assume two virtual subnetworks as A and B. North, South, West, Northwest and Southwest-bound packets are assigned to Subnetwork B covering the channel sets (W, N2, S2) while East, Northeast and Southeast-bound packets are assigned to Subnetwork A covering the channel sets (E, N1, S1) (line 4). Figure 13(a) illustrates the example paths for RULE 1 where Subnetworks A and B are differentiated by the blue and green colors, respectively.

In addition, to tolerate FC-RUT, packets need to follow RULE 2 and RULE 3 inside district regions as follow:

RULE 2: When the current router is in the distinct region, packets choose the direction which is not connected to an FC-RUT and has the least congestion (line 8, 14, 22, 42). Packets are delivered through FC-RUTs if there is no more possible direction. If the FC-RUT is the west/east neighbor of the current router, packets choose north and south direction to turn around the FC-RUT (Figure 13(b7-b8)) (line 14). Moreover, packets can directly pass through the FC-RUT in the north-south direction when the data path connections are fixed (Figure 13(b1-b6)) (line 22).

RULE 3: If the destination router of a packet is an FC-RUT, packets are routed to the ladder router first (line 10, 20, 31). When they reach the north and south neighbor of the destination router, these packets are switched to Subnetwork A (E, N1, S1) as shown in Figure 13(b10) (line 20).

Normally packets are delivered to destinations using RULE 1. Following RULE 2 and RULE 3, packets can be delivered to all routers inside a distinct region. Thus, this routing algorithm provides the reachability to tolerate one FC-RUT within the distinct region. As discussed in Section 3.3, the test sequence and TIT make sure that two routers within a distinct region are not under test simultaneously. Therefore, this routing algorithm can guarantee the reachability to all destinations.

To avoid deadlock, some turns are prohibited in this algorithm. First, U-turns are not allowed in RULE 4. Second, packets can switch from Subnetwork B to Subnetwork A but not vice versa. Packets from/to the routers in the eastern border follow RULE 5.

RULE 4: If packets arrive at the indirect neighbors of an FC-RUT, the routing algorithm considers the status of FC-RUT in advance to avoid taking a U-turn (line 40). For example in Figure 13(b11), packets are routed to the west direction if delivered from the current router toward the ladder router. As a result, packets will not reach the east neighbor of the FC-RUT. A similar case applies to Figure 13(b12).

RULE 5: If the destination router is on the eastern border, packets are routed to the west neighbor of the destination first (line 19, 30, 36). The dot lines in Figure 13(a) are not allowed if the destination router is in the eastern border unless packets meet other FC-RUTs during the transmission. If the packets meet FC-RUTs on the adjacent column of the eastern border, the packets can turn to the eastern border (line 37). Moreover, if a packet is north or south-bound packet and the source router is an FC-RUT on the eastern border, packets are delivered to the west column of the destination through the fixed channel. The packet does not turn east until it reaches the destination row (line 12). The example is shown in Figure 13(c6).

RULE 4 and RULE 5 sacrifice adaptiveness to guarantee the livelock and deadlock-freedom, and they do not affect the reachability.

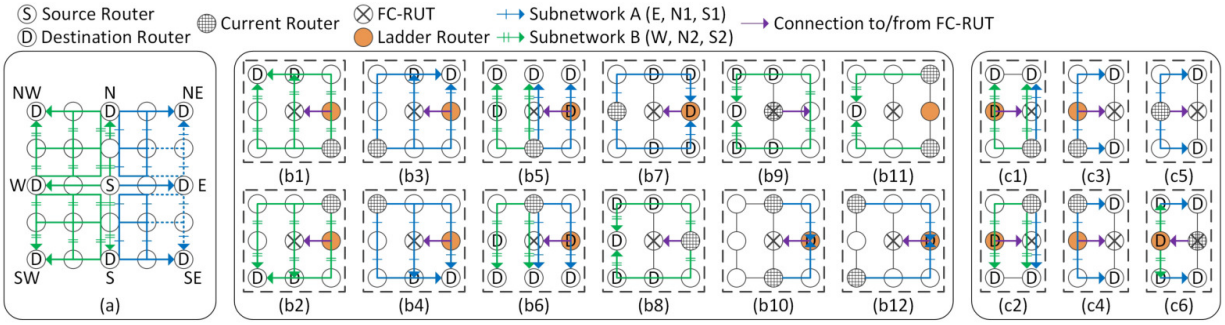

Fig. 13. Examples of routing rules. (a) the default path selection and virtual channel assignment. Dot lines are not allowed if the destination is on the eastern border. (b1-b12) examples of routes within the distinct region. (c1-c6) examples of routes within the distinct region if the FC-RUT is on the eastern border. Each subfigure shows possible paths from one router (marked as the current router) to several routers (marked as the destination router). The path between the ladder routers to the NI connected to FC-RUT is marked by purple color. The marks of '-' and '=' on the path distinguish the chosen virtual channels. packets are assumed currently at the current router and going to be delivered to a destination marked by D or FC-RUT. Packets follow the paths highlighted in the figures and reach the destination router. From the destination routers, they are either delivered to other routers outside the distinct region or NI.

5.2 Proof of deadlock-freedom

If the allowed turns in the network have the possibility of forming a closed cycle, the NoC faces the risk of deadlock. On the other hand, forbidding turns limits the adaptiveness of packets and thus the performance. To prove the freedom from deadlock, we divide the network into disjoint subnetworks, each includes a set of channels. Only three directions are used in each subnetwork to prevent a closed cycle. This method of proof is inspired from the EbDa theory [31].

The allocation of channels is in a way that north and south directions have two virtual/physical channels (N1/S1 and N2/S2) while the east and west directions have one virtual/physical channel. The network can be partitioned into two disjoint subnetworks: Subnetwork A covers channel sets (E, N1, S1) and Subnetwork B covers channel sets (W, N2, S2), respectively. RULE 1 ensures that each packet is assigned to one subnetwork. The allowable turns in the Subnetwork A are as E-N1, E-S1, N1-E and S1-E while the allowable turns in Subnetwork B are as W-N2, W-S2, N2-W and S2-W. In addition, packets can switch from Subnetwork B to Subnetwork A, allowing the turns N2-E and S2-E. Based on the proof on [31], this transition does not close a cycle.

If the source of a north/south-bound packet is a FC-RUT, packets are first routed to the ladder router, located on the east side of the destination. Deadlock can be avoided by assigning the north/south-bound packets into Subnetwork B (Figure 13(b9)). West/northwest/southwest-bound packets are also assigned to the same subnetwork. Figure 13(b7-b8) illustrate the allowed turns in each subnetwork. As described in RULE 2, to reroute packets around the FC-RUT on the east-west direction, some turns are needed. For example, the west-bound packets may require turns between W and N2/S2 directions which are already allowed.

If the destination is FC-RUT, packets should be first delivered to the ladder router (that is the east neighbor of the destination). For the west-bound packets, it is necessary to deliver them from north/south neighbor to the east neighbor of the destination as Figure 13(b5-b6) shows. RULE 3 allows the turns from N2/S2 to E but the opposite directions are forbidden, which means packets switch from Subnetwork B to Subnetwork A but not vice versa. These packets stay in Subnetwork A until they reach their destinations as shown in Figure 13(b10). However, if the destination is on eastern border, the ladder router will be the west neighbor of the destination, which means the turns from E to N2/S2 appear under the RULE 1 and 2. Thus, RULE 5 is set so that the turn from N2/S2 to E is allowed between two subnetworks (Figure 13(c6)).

To sum up, the network is divided into two disjoint subnetworks and packets are assigned to one of these subnetworks depending on the position of the source and destination nodes. Packets in subnetwork B can safely switch to the Subnetwork A but packets in Subnetwork A cannot switch to Subnetwork B. In this routing algorithm, we use the turns that can be formed in Subnetwork A, Subnetwork B, and by switching from Subnetwork B to A. According to [31], this routing algorithm does not lead to a deadlock because there is no possibility of a closed cycle.

6 Experiment and Discussion

In this section, EsyTest is analyzed and evaluated from five aspects: test coverage, hardware compatibility, network performance, system performance and hardware overhead. EsyTest is compared with test strategies adapted from [12], [27], [28] and details are given in Section 6.1.

6.1 Experiment Setup

The experiments are performed by simulation environment built with Simplescalar [32] and ESYNet [33]. Table 3 summarizes the parameters for the experimental setup. The experimental environment is a 64-core many-core system. The system is consist of 64 Alpha 21264 processing cores and 4 memory controllers. Each processing core contains a private 32KB L1 cache. The entire system shares the 8MB L2 cache. L2 cache is distributed into processing cores, each containing part of it (128KB). The interconnection architecture of this system is a 10x8 mesh NoC. The routers in the western and eastern border are used to connect memory controllers, and the inner eight columns (8x8 mesh) are used to connect the processing cores. Each router has 5 physical ports connecting to local, east, west, north and

south directions. The local, east, and south ports have two virtual channels while the north and south ports have only one virtual channel. Each virtual channel has a 12-flit buffer. The implemented routing algorithm is DyXY. The PARSEC benchmarks [29] are executed on the system.

TABLE 3
Table of Parameters

Parameters	Value
Number of cores	64
L1 cache	32 KB
total L2 cache	8 MB
L2 cache in each core	128KB
Network size and topology	10×8 Mesh
Number of physical ports	5
Number of VC channel	1(E,W,L) or 2(N,S)
Size of input buffer	12
Routing algorithm	DyXY

We compare EsyTest with three other methods as Baseline [12], TAFD [27] and TARRA [28]. TAFD is a selected acronym for Traffic-Aware Fault Detection method [27]. The Baseline, TAFD and EsyTest methods have the same router architecture and test wrapper as shown in Figure 3. These methods are adapted to fit the router architecture of EsyTest with some modifications described as follows. The Baseline and TAFD methods utilize DyXY as the routing algorithm while new routing algorithms are developed for EsyTest and TARRA. Unlike EsyTest, in the Baseline method, packets have to be blocked in the neighbor routers of the RUT during the entire test procedure including both data and control path test. On the other hand, TAFD presents the idea of using the free slots to test the data path, but packets are still blocked during the test of the control path. Obviously, the duration of blocking in TAFD is shorter than the Baseline method.

The router architecture of TARRA is different from EsyTest as it requires seven physical ports in total. It requires two channels in North/South direction, and one channel in East/West direction, thus it has seven physical ports in total. During the test procedure, packets pass through the RUT using bypass channels. An adaptive routing algorithm is designed to optimize the network performance.

In this section, the test methods published in [12] and [13] are adapted to the EsyTest, TARRA, and Baseline test strategies. [12] suggests test packets with 34 test vectors for the data path. Each test vector has 34 bits the same as the width of the data path. These packets are sent from neighbor routers and the NI of the RUT to the other neighbor routers and NIs. We assume that it needs about 1,000 cycles to run test to cover all data paths through the RUT as shown in Figure 3. [13] proposes one methodology to optimize and minimize the test procedure of a router. For each module in router (e.g. routing calculator and arbiters), only 2,000 vectors are necessary to achieve 100% fault coverage. Because these modules are tested in parallel, the test procedure needs about 2,000 cycles. The T_{free} and T_{block} are set to 1,000 cycles so that the test can be finished even with heaviest traffic load, and T_{test} is set to 2,000 cycles. According to Equation 1, the lower bound of TIT is $(1,000 + 1,000 + 2,000) \times 4.21 = 16,840$ cycles. Thereby, we selected TIT from $20K$ to $1000K$ cycles. The TAFD method uses the same timing sequence as EsyTest. However, the packets are blocked during the control path test (2,000 cycles) but are not blocked during the data path test (2,000 cycles). With the Baseline and TARRA, the test procedure needs 3,000 cycles (1,000 cycles for the data path test and 2,000 cycles for the control path) to finish both the data path and control path tests.

For the data path test, test packets contain all or part of the test vectors. Two extreme cases are experimented: one test packet contains only one test vector, or one test packet contains all test vectors. In addition to the test vectors, one test packet has one head flit and one tail flit. Therefore, there are two different sizes of test packets: 34 packets with 3 flits (1 head flit, 1 body flit and 1 tail flit, which is called EsyTest-3) and 1 packet with 36 flits (1 head flit, 34 body flits and 1 tail flit, which is called EsyTest-36). For each experiment, only one size of packet is used. EsyTest-3 and EsyTest-36 are two extreme cases with largest and smallest possible packets.

6.2 Analysis of Test Coverage and Hardware Compatibility

By the test coverage, we mean the components that can be tested during the test procedure. In the Baseline method, all the components can be tested in the network. Meanwhile, the RUT is completely isolated during the test procedure. This full test coverage comes at the cost of blocking packets in the neighbor routers during the test which leads to a significant performance loss. The test coverage in the TAFD method is limited to only global links, as reported in Table 1. In this section, we mainly discuss the test coverage and hardware compatibility of EsyTest and TARRA because both methods do not block packets during the test.

In TARRA, bi-directional bypass channels are used to maintain the connections in the north-south direction, east-west direction and access to the local core. The bypass channels connect the input ports and output ports of the routers. These bypass channels can be tested when the router is not under test as shown in Figure 14(a). On the other hand, the components within the router, including the data path and control path can be tested by BIST during the test procedure (Figure 14(b)). However, the global links (router-to-router links) cannot be tested and must be available all the time, both in the normal and test modes.

As shown in Figure 9 and Figure 10, EsyTest tests the data path and control path separately. The test wrappers of data paths contain not only the buffers, crossbars, and registers within the router but also the global links. The test vectors can detect the faults on the wires, memories, and registers. The test wrappers of the control path cover all the control logic which should be tested by well-designed test cases. By applying the test procedure of the data and control path, all components in the network can be tested.

Both TARRA and EsyTest have extra circuits to implement the function of non-blocking BIST. These additional circuits change the micro-architecture of a router. TARRA requires two physical ports on the North and South directions such that one port is used to access the local NI while the other port is used to make the connection in the North

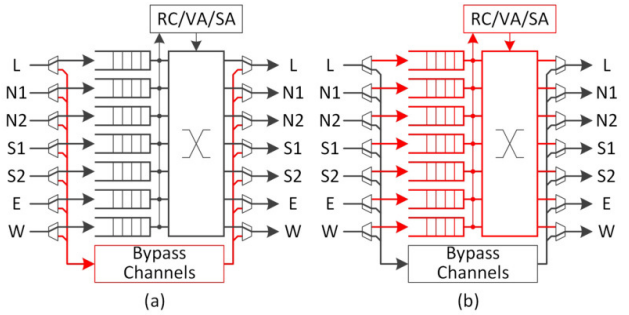

Fig. 14. Test coverage of TARRA. Components in red color are under test. (a) During the Normal phase, the router operates normally, and the bypass channels can be tested. (b) During the Test phase, the packets are delivered through bypass channels, and the components within the RUT can be tested.

TABLE 4
Traffic profiles used in simulation

Traffic profile	Rule to generate traffic	Latency (cycles)	Link utilization	
			Max	Min
Uniform	To each router chosen randomly.	22.31	0.387	0.015
Transpose1	(x,y) → (7-y,7-x).	23.14	0.602	0.001
Transpose2	(x,y) → (y,x).	23.14	0.630	0.001
BitReversal	Destination id is the bit reversal of source id.	22.89	0.630	0.001
Shuffle	Destination id is the right loop shift of source id.	18.63	0.447	0.001
Butterfly	Get destination id by swapping the LSB and MSB of source id.	14.24	0.460	0.001

and South direction. Therefore, in TARRA, the topology must be adjusted to meet the requirement of additional physical ports. On the other hand, EsyTest does not require additional physical channel and thus it is compatible with different router architectures. The hardware compatibility is one of the advantages of EsyTest over TARRA.

In both TARRA and EsyTest, the additional circuits added for the test can be examined during the Normal phase.

6.3 NoC Performance under Synthetic Traffic

In this section, the performance of NoC is simulated separately under six synthetic traffic profiles as listed in Table 4. The 8×8 mesh network is used for the experiments. To keep the destination addresses, generated by synthetic rules, within the network, the network should be a square shape under BitReversal, Shuffle, and Butterfly traffic patterns. To examine the methods for higher injection rates than PARSEC, the packet injection rate is set to 0.03 packet/router/cycle in synthetic traffics. Each packet contains 5 flits which is approximately 9.6 flit/cycle. The simulation takes 100,000 cycles, and about 192,000 packets will be injected into the network. The reported latency is the average latency of these 192,000 packets.

Table 4 reports the characteristics of traffic profiles used in the simulation. First, the rules to generate the traffic is described where these rules determine the destination of each packet. Second, the average latency without the test procedure is reported. The maximum link utilization raises up to 0.630 that is more than 3 times higher than the link utilization under PARSEC. Only one-third time-slots can be used for test procedures. The experiment under synthetic traffic presents the situation when the link is under heavy traffic.

In Figure 15, each point demonstrates the average packet latency under a specific traffic pattern and TIT. The latency of test packets is not considered in the reported latency values but their resource occupations affect the latency of data packets. The points belonging to the same test strategy and test packet size are connected by one curve. The average latency without any test procedure is also illustrated in the figure for comparison.

The Baseline and TAFD methods have a serious impact on the NoC performance. As shown in Figure 15, the average latency increases rapidly as the test interval time reduces (i.e. the test frequency increases). By decreasing the TIT to 200K cycles, we observe a dramatic latency increase under all traffic patterns, except the shuffle traffic that shows a smoother latency change. The TAFD method blocks the packets only during the control path test (2,000 cycles) while the Baseline method blocks packets for the whole test period (3,000 cycles). This is the main reason why the Baseline method leads to a higher latency than the TAFD method.

Regarding the EsyTest, when the TIT is 1000K cycles, the average latency increases no more than 1 cycle. As TIT decreases, the average latency increases accordingly. The latency does not grow more than 5 cycles when TIT is as low as 60K cycles. When TIT is 20K cycles, the average latency increases for over 10 cycles. This is the result of both high-frequency testing and high traffic pressure.

Moreover, the growing range of EsyTest-3 is smaller than the growing range of EsyTest-36. In fact each test packet has to carry a head and a tail flit in addition to the test vectors. Assuming 34 test vectors in this paper, EsyTest-3 requires nearly 3 times more flits (34 packets, each with 3 flits) than EsyTest-36 (1 packet with 36 flits). However, as was discussed in Section 3.1, small test packets introduce a lower delay on data packets as the waiting time to receive resources becomes shorter. In general, the performance loss in the EsyTest method is influenced by both test interval time and traffic load.

Similar to EsyTest, the performance of TARRA drops when the TIT gets lower than 60K cycles. TARRA has a marginal better performance than EsyTest, shown in subplots of Figure 15. A better performance is mainly due to using an extra channel in the north and south directions and also the adaptive routing algorithm with a higher adaptiveness. TARRA even shows better results than NoTest. This is due to the fact that bypass channels skip the latency of a router, and resulting in a lower latency [28]. On the other hand, TARRA is unable to test global links and has a limitation on the choice of the router architecture.

6.4 System Performance Under PARSEC benchmark

In this section, the execution time of the PARSEC benchmark on the Baseline, TAFD, TARRA and EsyTest test strategies is measured to evaluate the influence of test on the system

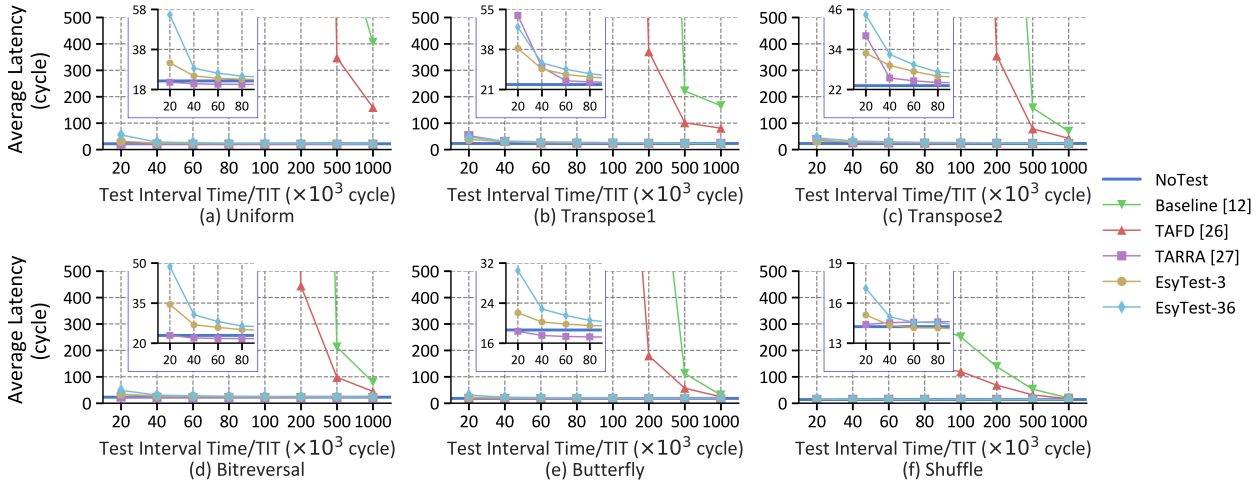

Fig. 15. Average latency of Network-on-Chip under synthetic traffic. Subplots zoom the graph on low test interval times. EsyTest-3 means the test vectors are packaged into 34 packets of 3-flit; EsyTest-36 means the test vectors are packaged into 1 packet of 36-flit.

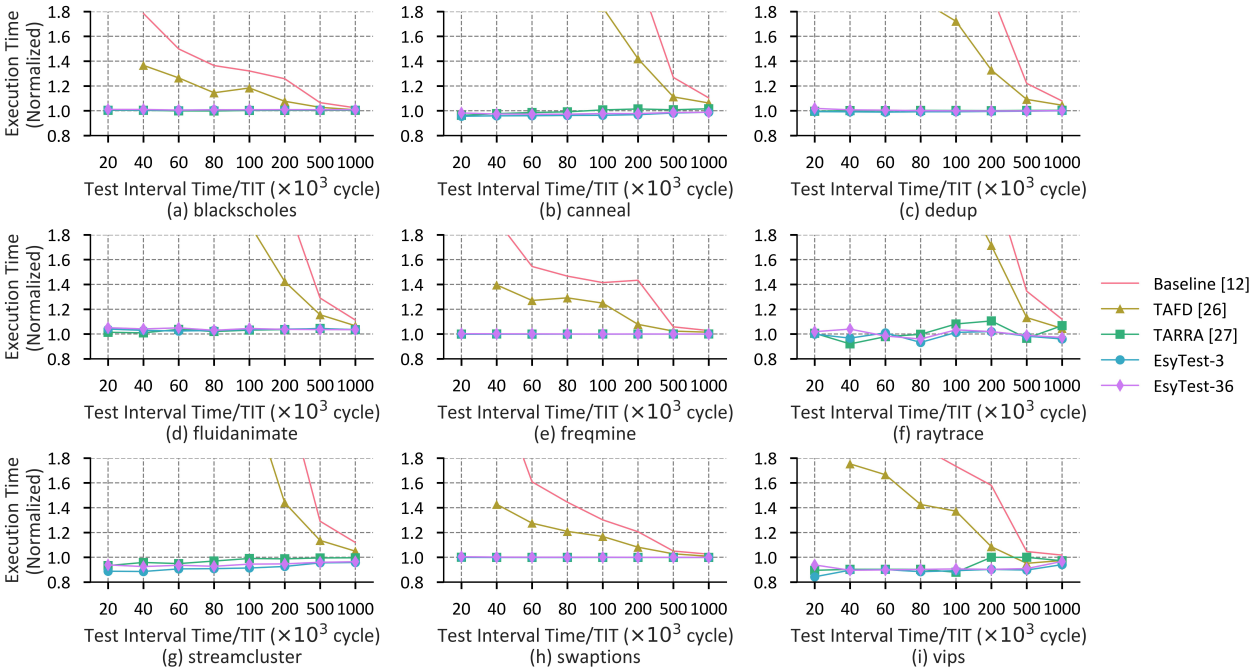

Fig. 16. The execution time of PARSEC benchmarks. Execution time is normalized to the situation where no router is under test. EsyTest-3 means the test vectors are packaged into 34 3-flit packets; EsyTest-36 means the test vectors are packaged into 1 36-flit packet.

performance. Benchmarks run for 10M instructions for different TITs. The execution times for the same benchmark are normalized by the execution time without any test procedure. 1.0 means the execution time in the presence of the test procedure is equal the one in the absence of the test procedure, meaning that the test strategy does not influence the system performance.

Each subfigure in Figure 16 gives the normalized execution times of simulations for different TIT and test strategies. Each point indicates the normalized execution time for a valuation of TIT and test strategy. Points for the same test strategy and test packet size are connected in one line. To get a smooth curve, four experiments are executed for each combination of the test method and test interval time. Each experiment has a different offset on the test sequence. For example, experiment 0 starts at router 0 in group 0; experiment 1 starts at router 1 in group 1, experiment 3 starts at router 10 in group 2 and experiment 4 starts at router 11 in group 3. By taking an average on these four experiments, the curves are smooth enough and demonstrate the trend of EsyTest and TARRA.

The Baseline and TAFD methods show a rapid growth in the execution time under all benchmark programs as TIT reduces. The reason is that packets are blocked during the whole test procedure in Baseline (3,000 cycles) and the control path test in TAFD (2,000 cycles). The trend of the Baseline and TAFD methods confirms the fact that traditional methods have a major problem in running periodic

BIST at high frequency.

In contrast, EsyTest shows small fluctuations on the execution time; and the curve remains close to the value of 1.0. In the account of all simulations, the execution time of EsyTest increases less than 1% in 83% of simulations and less than 5% in 99% of simulations. The maximum increase in the execution time (5.1%) is observed when the test strategy is EsyTest-36, the traffic pattern is fluidanimate, and TIT is 20K cycles. The growth of execution time, even negligible, is due to three reasons: 1) packets are blocked for only a few cycles during the Emptying and Recovery phases; 2) prioritizing test packets over data packets in the Block phase of testing; and 3) sacrificing some degree of routing adaptiveness to tolerate FC-RUTs and guarantee deadlock freedom.

The execution time of EsyTest reduces under canneal, streamcluster and vips (Figure 16(b, g, i)) when TIT decreases. The reason is that a packet needs three cycles to pass through a normal router while only one cycle is needed to pass through FC-RUT because it is not necessary to do routing calculation, virtual channel allocation and switch allocation. This reduced number of cycles helps in lowering the traffic and thus a better performance.

The execution times of EsyTest-3 and EsyTest-36 are very close. As shown in Figure 16, the curves of EsyTest-3 and EsyTest-36 are mostly overlapped. The small difference is because of the longer test packets in EsyTest-36 that forces data packets to wait longer to receive the resources, as was discussed in Section 3.1.

To conclude, EsyTest can increase the test frequency significantly while keeping the performance in the same level. The small fluctuations of execution times are because of the communication characteristics, the use of fixed channels, and cache behaviors.

The results shown in Figure 15 and Figure 16 also illustrate the performance loss caused at different packet injection rates. As shown in Table 2, the average flit injection rate in the PARSEC benchmark is between 0.017 flit/cycle and 2.976 flit/cycle, and the maximum link utilization of the PARSEC benchmark under the DyXY routing algorithm is no more than 0.2. Hence, the link is free most of the time. Table 4 shows that the maximum link utilization under synthetic traffic can reach 0.63, which means the link is occupied most of the time. As discussed in the paper, if there are more free slots on the data path, more test packets can be delivered in free slots, and fewer test packets need to be delivered during the Blocking phase. Since the packet injection rate is lower under the PARSEC benchmark, the performance loss caused by the testing procedures is also lower than the synthetic traffic patterns. In general the performance loss is very small regardless of the business of the data path.

6.5 Hardware Overhead

Table 5 lists the area overhead and power consumption of EsyTest, TARRA, TAFD, Baseline and NoTest (a router without a test structure). The data is provided by Synopsys Design Compiler under TSMC45nm technology. The overhead of the BIST circuit is not included since the test circuit design is outside the scope of this paper.

As the values in Table 5 shows, test strategies do not have a high area overhead. Baseline and TAFD increase the area and power by nearly 3% that is mainly because of the test wrappers. TAFD has a larger area due to the required logic to detect free slots on links. TARRA needs about 9% more area and 4.67% more power due to the bypass channels and the adaptive routing algorithm.

TABLE 5
Power and area analysis

	NoTest	Baseline	TAFD	TARRA	EsyTest
Area (mm^2)	0.0251	0.0258 (2.79%)	0.0258 (2.86%)	0.0273 (9.02%)	0.0275 (9.88%)
Power (μW)	391.00	399.77 (2.24%)	400.18 (2.35%)	409.27 (4.67%)	409.10 (4.63%)

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

Table 5 shows that EsyTest costs 9.88% more area and 4.63% more power, belonging to the test wrappers, test sequence controller and routing calculator units. It also indicates that EsyTest does not need significant overhead but it benefits reliability with full test coverage, higher test frequency, and better hardware compatibility.

7 Conclusion

Built-In Self-Test (BIST) is commonly used in NoC to test and diagnose the faults and improve the reliability. In traditional BIST design, the test procedure imposes a large negative impact on performance so that test cannot be performed at high frequency or in parallel. We proposed the EsyTest strategy to minimize the impact of test procedures on the system performance and to increase the test frequency. EsyTest applies different test wrappers on the data path and the control path so that all components can be tested. During the data path test, test vectors are injected at the free time slots of the router to minimize the impact on regular traffic. During the control path test, the network can still guarantee the reachability to all destinations including the cores connected to the router under test. This is achieved by proposing a reconfigurable router architecture and an adaptive routing algorithm. During the test procedure of the control path, the data path is configured to fixed connections and thus can be used to deliver packets to neighbors. A special test sequence is proposed as well so that routers can be tested in parallel, e.g. at most one-fourth of routers can be tested simultaneously. The experiments under the PARSEC benchmark show that the EsyTest strategy maintains the performance level at the cost of negligible hardware overhead. On top of that, the test frequency can be increased dramatically which can enhance the network reliability.

Appendix

*The value of X increases from left to right. The value of Y increases from top to bottom.
*DNR(n): direct neighbor router on direction n.
*INR(mn): indirect neighbor router on direction mn.

Require: Current: (X_C,Y_C); Source: (X_S,Y_S); Destination: (X_D,Y_D); Δ_X: $|X_D - X_C|$; Δ_Y: $|Y_D - Y_C|$;in: input port;
Ensure: Selected port: Sel;

$pos \leftarrow$ L, N, S, E, W, NE, NW, SE, or SW according to the current and destination position.
$dir_X \Leftarrow$ E **when** $X_D > X_C$ **else** W; ▷ RULE 1
$dir_Y \Leftarrow$ S **when** $Y_D > Y_C$ **else** N; ▷ RULE 1
$vc \Leftarrow$ vc1 **when** $X_D > X_S$ **else** vc2; ▷ RULE 1
if $pos =$ L **then** $Sel \leftarrow$L;
else if $pos =$ E or W **then**
    if DNR(dir_x) is available **then**
        $Sel \leftarrow dir_X$; ▷ RULE 2
    else if DNR(W) is destination **then**
        $Sel \leftarrow W$; ▷ RULE 3
    else if DNR(E) is destination and on the eastern border **then**
        $Sel \leftarrow E$; ▷ RULE 5
    else $Sel \leftarrow N(vc)$ or $S(vc)$ according to
        congestion and availability; ▷ RULE 2
    end if;
else if $pos =$ N or S **then**
    if DNR(dir_Y) is FC-RUT and DNR(dir_Y) is destination **then**
        if DNR(dir_Y) is on the eastern border column **then**
            $Sel \leftarrow W$; ▷ RULE 5
        else $Sel \leftarrow E$; ▷ RULE 3
        end if;
    else $Sel \leftarrow dir_Y(vc)$; ▷ RULE 2
    end if;
else if $pos =$ NE, SE, NW or SW **then**
    if $\Delta_X = 1$ and $\Delta_Y = 1$ and INR(pos) is FC-RUT **then**
        if $pos =$ NW or SW **then**
            $Sel \leftarrow dir_Y(vc)$; ▷ RULE 3
        else
            if Destination is on the eastern border column **then**
                $Sel \leftarrow dir_Y(vc)$; ▷ RULE 5
            else $Sel \leftarrow dir_X$; ▷ RULE 3
            end if;
        end if;
    else if $\Delta_X = 1$ and destination is on the eastern border column **then**
        if DNR(dir_Y) is available **then**
            $Sel \leftarrow dir_Y(vc)$; ▷ RULE 5
        else$Sel \leftarrow dir_X$ ▷ RULE 5
        end if;
    else if $\Delta_Y = 1$ and INR(pos) is FC-RUT **then**
        $Sel \leftarrow dir_X$; ▷ RULE 4
    else $Sel \leftarrow dir_X$ or $dir_Y(vc)$ according to
        congestion and availability; ▷ RULE 2
    end if;
end if;

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.61534002, No.61761136015, No.61701095, Central Universities under Grant ZYGX2016J042, ZYGX2015J007. This work is also supported by VR and VINNOVA-MarieCurie.

REFERENCES

[1] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Computing Surveys (CSUR)*, vol. 38, no. 1, p. 1, 2006.
[2] R. Baumann, "Soft errors in advanced computer systems," *IEEE Design Test of Computers*, vol. 22, no. 3, pp. 258–266, 2005.
[3] J. Keane and C. H. Kim, "An odomoeter for cpus," *IEEE Spectrum*, vol. 48, no. 5, pp. 28–33, 2011.
[4] 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 *45th Annual IEEE/ACM International Symposium on Microarchitecture*. ACM/IEEE, 2012, pp. 60–70.
[5] A. DeOrio, D. Fick, V. Bertacco, D. Sylvester, D. Blaauw, J. Hu, and G. Chen, "A reliable routing architecture and algorithm for nocs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 5, pp. 726–739, 2012.
[6] G. Schley, A. Dalirsani, M. Eggenberger, N. Hatami, H.-J. Wunderlich, and M. Radetzki, "Multi-layer diagnosis for fault-tolerant networks-on-chip," *IEEE Transactions on Computers*, vol. 66, no. 5, pp. 848–861, 2017.
[7] Q. Yu and P. Ampadu, "A dual-layer method for transient and permanent error co-management in noc links," *IEEE Transaction on Circuits and System II*, vol. 58, no. 1, pp. 36–40, 2011.
[8] L. Xie, K. Mei, and Y. Li, "Repair: A reliable partial-redundancy-based router in noc," in *IEEE 8th International Conference on Networking, Architecture and Storage (NAS)*. IEEE, 2013, pp. 173–177.
[9] H. Al-Asaad, B. T. Murray, and J. P. Hayes, "Online bist for embedded systems," *IEEE Design Test of Computers*, vol. 15, no. 4, pp. 17–24, 1998.
[10] J. Wang, M. Ebrahimi, L. Huang, A. Jantsch, and G. Li, "Design of fault-tolerant and reliable networks-on-chip," in *IEEE Computer Society Annual Symposium on VLSI*, 2015, pp. 545–550.
[11] L. Rashid, K. Pattabiraman, and S. Gopalakrishnan, "Characterizing the impact of intermittent hardware faults on programs," *IEEE Transactions on Reliability*, vol. 64, no. 1, pp. 297–310, 2015.
[12] M. Kakoee, V. Bertacco, and L. Benini, "At-speed distributed functional testing to detect logic and delay faults in nocs," *IEEE Transactions on Computers*, vol. 63, no. 3, pp. 703–717, 2014.
[13] P. Saltarelli, B. Niazmand, J. Raik, V. Govind, T. Hollstein, G. Jervan, and R. Hariharan, "A framework for combining concurrent checking and on-line embedded test for low-latency fault detection in noc routers," in *9th International Symposium on Networks-on-Chip*, 2015, pp. 1–8.
[14] S. R. Suja and M. Deivakani, "Testing of fifo buffer of noc router using bist," in *IEEE International Conference on Electrical, Instrumentation and Communication Engineering (ICEICE)*, 2017, pp. 1–6.
[15] B. Aghaei, A. Khademzadeh, M. Reshadi, and K. Badie, "Link testing: a survey of current trends in network on chip," *Journal of Electronic Testing*, vol. 33, no. 2, pp. 209–225, 2017.
[16] 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 Tramscations on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 6, pp. 1364–1376, 2014.
[17] N. Caselli, A. Strano, D. Ludovici, and D. Bertozzi, "Cooperative built-in self-testing and self-diagnosis of noc bisynchronous channels," in *IEEE 6th Internal Symposium on Embedded Multicore SoCs*. IEEE, 2012, pp. 159–166.
[18] H. Yi and S. Kundu, "Core test wrapper design to rreduce test application time for modular soc testing," in *IEEE International Symposium on Defect and Fault Tolerant of VLSI System*. IEEE, 2008, pp. 412–420.
[19] 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 *Design, Automation & Test in Europe Conference & Exhibition*. IEEE, 2011, pp. 1–6.
[20] 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 Transactions on Computers*, vol. 57, no. 9, pp. 1202–1215, 2008.
[21] B. Bhowmik, J. K. Deka, and S. Biswas, "Towards a scalable test solution for the analysis of interconnect shorts in on-chip networks," in *IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2016, pp. 394–399.

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

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

[24] C. Grecu, A. Ivanov, R. Saleh, and P. Pande, "Testing network-on-chip communication fabrics," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 12, pp. 2201–2214, 2007.

[25] M. Ke, Y. Zhang, and J. Jiang, "High-level fault diagnosis on network-on-chip using path tracking," in *International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, 2017, pp. 1–4.

[26] 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 Computers & Digital Techniques*, vol. 3, no. 5, pp. 487–500, 2009.

[27] J. Liu, J. Harkin, Y. Li, and L. Maguire, "Online traffic-aware fault detection for networks-on-chip," *Journal of Parallel and Distributed Computing*, vol. 74, no. 1, pp. 1984–1993, 2014.

[28] L. Huang, J. Wang, M. Ebrahimi, M. Daneshtalab, X. Zhang, G. Li, and A. Jantsch, "Non-blocking testing for network-on-chip," *IEEE Transactions on Computers*, vol. 65, no. 3, pp. 679–692, 2016.

[29] *the PASRC Benchmark Suite*, http://parsec.cs.princeton.edu/.

[30] W. J. Dally and B. P. Towles, *Principles and practices of interconnection networks*. Elsevier, 2004.

[31] M. Ebrahimi and M. Daneshtalab, "Ebda: A new theory on design and verification of deadlock-free interconnection networks," in *44th International Symposium on Computer Architecture (ISCA)*, 2017, pp. 1–13.

[32] T. Austin, E. Larson, and D. Ernst, "Simplescalar: An infrastructure for computer system modeling," *Computer*, vol. 35, no. 2, pp. 59–67, 2002.

[33] J. Wang, Y. Huang, M. Ebrahimi, L. Huang, Q. Li, A. Jantsch, and G. Li, "Visualnoc: A visualization and evaluation environment for simulation and mapping," in *the Third ACM International Workshop on Many-core Embedded Systems*, 2016, pp. 18–25.

Junshi Wang was born in Liaoning, China in 1989. He received the Ph.D and B.S. degree in communication engineering from the University of Electronic Science and Technology of China (UESTC), Chengdu, China in 2017 and 2012. He is currently working at Beijing Zhaoxin Electronic Technology Co., Ltd. His research interests include reliability of network-on-chip, many-core system and micro-architecture modeling.

Masoumeh Ebrahimi received a PhD degree with honours from University of Turku, Finland in 2013. She is currently a senior researcher at KTH Royal Institute of Technology, Sweden. Her scientific work contains more than 80 publications including book chapters, journal articles and conference papers. The majority of work has been performed on interconnection networks, fault-tolerant methods, multicast communication, and congestion-aware techniques. She is a member of the IEEE.

Letian Huang was born in Sichuan Province, China, in 1984. He received the M.S. and Ph.D. degrees from University of Electronic Science and Technology of China (UESTC), Chengdu, China in 2009 and 2016, respectively, in Communication and Information System. He is an Associate Professor of UESTC. His scientific work contains more than 40 publications including book chapters, journal articles and conference papers. His research interests include Heterogeneous Multi-Core System-on-Chips, Network-on-Chips, and Mixed Signal IC Design.

Xuan Xie received the M.S. and Ph.D. degrees from University of Electronic Science and Technology of China (UESTC), Chengdu, China in 2009 and 2016, respectively. She is currently working with UESTC. Her scientific work contains more than 20 publications including journal articles and conference papers. Her research interests include signal processing, fault diagnosis, and information theory.

Qiang Li received the B.Eng. degree in Electrical Engineering from Huazhong University of Science and Technology (HUST), Wuhan, China, in 2001, and the Ph.D. degree from Nanyang Technological University (NTU), Singapore, in 2007. Since 2001, he has been working on analog/RF circuits in both academia and industry, holding positions of Engineer, Project Leader and Technical Consultant in Singapore, and Associate Professor in Denmark. He is currently a full Professor at the University of Electronic Science and Technology of China (UESTC), heading the Institute of Integrated Circuits and Systems. His research interests include low-voltage and low-power analog/RF circuits, data converters, and mixed-mode circuits for biomedical and sensor interfaces. Dr. Li was a recipient of the Young Changjiang Scholar award in 2015, the National Top-Notch Young Professionals award in 2013 and the UESTC Teaching Excellence Award in 2011. He serves on the Student Research Preview (SRP) committee of ISSCC and the Technical Program Committee of ESSCIRC. He is the Founding Chair of IEEE Chengdu SSCS/CASS Joint Chapter.

Guangjun Li received M.S. 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 has published various 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 has been an Associate Professor (1997-2000) and a Full Professor (2000-2014) with the Royal Institute of Technology (KTH), Stockholm, Sweden. Now, he is a Full Professor with Technical University of Vienna, Vienna, Austria. He has published over 200 papers in international conferences and journals, and one book. 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.