At-Speed Distributed Functional Testing to Detect Logic and Delay Faults in NoCs

Mohammad Reza Kakoee, *Member*, *IEEE*, Valeria Bertacco, *Senior Member*, *IEEE*, and Luca Benini, *Fellow*, *IEEE*

Abstract—In this work, we propose a distributed functional test mechanism for NoCs which scales to large-scale networks with general topologies and routing algorithms. Each router and its links are tested using neighbors in different phases. The router under test is in test mode while all other parts of the NoC are operational. We use triple module redundancy (TMR) for the robustness of all testing components that are added into the switch. Experimental results show that our functional test approach can detect stuck-at, short and delay faults in the routers and links. Our approach achieves 100 percent stuck-at fault coverage for the data path and 85 percent for the control paths including routing logic, FIFO's control path, and the arbiter of a 5×5 router. We also show that our approach is able to detect delay faults in critical control and data paths. Synthesis results show that the area overhead of our test components with TMR support is 20 percent for covering stuck-at, delay, and short-wire faults and 7 percent for covering only stuck-at and delay faults in the 5×5 router. Simulation results show that our online testing approach has an average latency overhead of 3 percent in PARSEC traffic benchmarks on an 8×8 NoC.

Index Terms—NoC testing, online testing, delay fault, functional test

1 Introduction

Main microprocessor manufacturers are migrating to chip multiprocessors (CMPs) for their latest products. In CMPs, many cores are put together in the same chip and, as Moore's law continues to apply in the CMP era, we can expect to see a geometrically increasing number of processors and memories [1]. Intel developed a research chip called *single-chip cloud computer* (SCC) with 48 cores, under the tera-scale computing research program [2] and a chip prototype that included 80 cores (known as TeraFlops research chip) [3]. More recently, Intel has announced the *many integrated core* (MIC) architecture [4] as the latest breakthrough in multicore processors. The MIC project is the fruit of three research streams: The 80-core Tera-scale research chip program, the single-chip cloud computer initiative, and the Larrabee many-core visual computing project [5].

Embedded systems have also been shifting to multicore solutions (multiprocessor system-on-chips; MPSoCs). A clear example of high-end MPSoCs are the products offered by Tilera [6], where multicore chips provide support to a wide range of computing applications, including high-end digital multimedia, advanced networking, wireless infrastructure, and cloud computing.

Current trends indicate that multicore architectures will be used in most application domains with high energy-efficiency requirements. However, aggressive CMOS scaling accelerates transistor and interconnect wear out, resulting in shorter and less predictable lifespans for CMPs and MPSoCs [7]. It has been predicted that future designs will consist of hundreds of billions of transistors, with upwards of 10 percent of them being defective due to wear out and process variation [8]. Consequently, to support the current technology trends we must develop solutions to design reliable systems from unreliable components, managing both design complexity and process uncertainty [9], [10], with minimal nominal performance degradation.

Network on chip (NoC), a high performance and scalable communication mechanism, is being increasingly investigated by researchers and designers to address the issues of interconnect complexity in both CMPs and MPSoCs [11]. The reliability of NoC designs is threatened by transistor wear out in aggressively scaled technology nodes. Aging and wear-out mechanisms, such as oxide breakdown and electromigration, become more prominent in these nodes as oxides and wires are thinned to their physical limits. These breakdown mechanisms occur over time, so traditional post burn-in testing will not capture them. NoCs provide inherent structural redundancy and interesting opportunities for fault diagnosis and reconfiguration to address transistor or interconnect wear out.

A network on chip consists of routers, links, network interfaces (NIs), and cores. Failures can occur in any of these components. In this work, we focus on failures in routers and links, and propose an online functional testing solution for the NoC infrastructure. A number of works has evolved NoC reconfiguration to counteract faults in both routers and links [12], [13], [14], [15], [16], [17], [18]. For instance, routing algorithms for fault-tolerant networks have been extensively explored for network-level reconfiguration. These algorithms direct traffic around failed network

- *M.R. Kakoee and L. Benini are with the Department of DEIS, University of Bologna, Italy. E-mail: {m.kakoee, luca.benini}@unibo.it.*
- *V. Bertacco is with the CSE Department, University of Michigan, Ann Arbor, MI. E-mail: valeria@umich.edu.*

Manuscript received 31 Oct. 2011; accepted 22 Sept. 2013; published online 11 Oct. 2013.
Recommended for acceptance by R. Ginosar and K.S. Chatha.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TCSI-2011-10-0793.
Digital Object Identifier no. 10.1109/TC.2013.202.

components to avoid network deadlock. However, to take countermeasures against NoC faults, we must first detect and diagnose them. Most of the works proposed for fault-tolerant NoCs do not address the testing mechanism and how they find faulty components. Some of them rely on offline testing mechanisms, such as BIST and scan chains, which require external test sources, are not scalable and cannot be applied while the system is in operation. Others rely on traditional error detection and correction mechanisms, such as CRC and parity checkers, to detect data path faults in both links and routers [19], [20], [21]. However, their fault coverage is very limited and, more importantly, they cannot be used for fault detection in control paths of the router.

Recently, Lubaszewski et al. [22], [23] proposed a new post burn-in testing for NoCs, which is based on the at-speed functional testing of several 2×2 meshes in an N $\times$ N NoC. They test each 2×2 mesh using a set of test sequences and patterns. However, as they showed in [23], the 2×2 mesh can give good fault coverage for the links but not for the routers, especially for the routing logics, FIFO's control paths, and arbiters. To obtain higher fault coverage for the routers, they added more than 12 other mesh configurations including 3×1, 1×3, 2×2, and so on. However, having those additional configurations is not efficient for online testing and keeps several switches busy in testing procedure, impacting the NoC's performance and packet latency. Moreover, their approach is suitable for manufacturing testing using automatic test equipment (ATE), and implementing it in hardware leads to overhead especially with its enhanced version.

Motivation and contribution. The motivation behind this work is having a distributed and scalable online testing mechanism for NoC that can detect both logic and delay faults in data path and control path of the routers as well as the links with a reasonable fault coverage and with minimum hardware overhead. In this paper, we propose a scheme for online detection and diagnosis of faults in NoC's routers and links. First, we propose a functional test architecture, which can be implemented in routers and NIs with small hardware overhead. Similar to the work in [23], we use test pattern generator (TPG) and test response analyzer (TRA). However, to have minimum performance overhead on the NoC in online mode we take advantage of TPG and TRA in both router and NI. We also propose a hardware-efficient fault diagnosis module (FDM) in each router, which can diagnose the location of the fault. Second, we propose a new test sequence to test each router. Moreover, we use TMR for the reliability of testing components added to the router. Unlike the work proposed in [23], we do not use 2×2 meshes to test the entire NoC, but we test each router separately and using its neighbors in online and at-speed functional mode. We use logic and delay fault models and the corresponding system-level failure modes to better tune the test pattern sequences. Note that, in this work, we are seeking the integration of the test of interconnects and routers, at the lowest possible cost, but considering gate-level fault models. We try to cover most of the logic and delay faults in the entire router and in all data and control wires that build the NoC infrastructure.

Experimental results show that our functional test approach can detect all the stuck-at, delay, and short-wire faults in both interrouter and intrarouter wires. The results also show that our approach has a 100 percent stuck-at fault coverage for the data path and 85 percent for the control paths including routing logic, FIFO's control path, and the arbiter of a 5×5 switch. Synthesis results show that the hardware overhead of our test components with TMR support is 20 percent for covering stuck-at, delay, and short-wire faults and only 7 percent for covering stuck-at and delay faults. Simulation results show that our online testing approach has an average latency overhead of 3 percent in the real traffic benchmarks on an 8×8 NoC.

2 Related Work

NoC infrastructure is composed of three main parts: Routers, links, and NIs. Since the functionalities of these components are decoupled from each other, it is possible to test each of them separately. Testing each component separately in NoC means, we are testing NoC structurally. An alternative way is to functionally test the whole NoC. In this case testing, different elements of NoC are not decoupled from each other and they will be tested all together. The latter method is cheaper, but does not identify where is the defected element unless we have a fault diagnosis mechanism. Since the focus of this work is on routers and links, we give an overview of previous works performed on testing these parts.

2.1 Link Testing

As density in chip increases, the distance between wires decreases and accompanied with higher functional frequency, links are more susceptible to crosstalk noises. Other faults that can happen in links are shorts between wires of links (so-called bridging), opens, shorts to VDD (stuck-at-1), shorts to Ground (stuck-at-0), and delay fault, which is caused by noises or process, voltage, and temperature (PVT) variations.

Grecu et al. [19] propose a built-in self-test methodology for testing the channels of the communication platform. The proposed methodology targets crosstalk faults assuming the MAF fault model [24]. The authors also suggest that shorts between wires within a single channel are detected as well. The test strategy is based on two BIST blocks: Test data generator (TDG) and the test error detector (TED). TDG generates the test vectors capable of detecting all possible crosstalk faults in a channel connecting two adjacent routers. The test vectors are launched on the channel under test from the transmitter side of the link and then are sampled and compared for logical consistency at the receiver side of the link by the TED circuit. Their technique is suitable only for testing interswitch links in offline mode; moreover, they do not provide any data on the gate-level fault coverage of their technique.

In another approach, Pande et al. [25] propose to incorporate crosstalk avoidance coding and forward error correction schemes in the NoC data stream to enhance the system reliability.

The authors of [26] propose a functional test method for detecting delay faults in asynchronous NoC links connecting

between two switches, which are in two different clock domains. In this method, they are sampling data which is passed in interconnect before and after the handshake signals. If there is a timing error in the line, the data will not be the same before and after the handshake signals. Therefore, by comparing the two sampled data they can detect the delay fault. However, their method is offline and is suitable only for links.

Raik et al. [27] proposed a technique for diagnosing faulty links by applying test patterns at the border I/Os of the network. While very high fault coverage was achieved, their method is offline and the time complexity of the test configurations is square with respect to the rank of the NoC matrix. Moreover, to apply test patterns from network boundaries at-speed, a large number of test pins are necessary.

Lubaszewski et al. [22] proposed a post burn-in testing for NoC interconnects, which is based on the at-speed functional testing of several 2×2 meshes in an $N \times N$ NoC. Their method is also capable of detecting faults between distinct interswitch channels. However, their method is a postburn testing that cannot be applied in online mode and is suitable only for mesh-like networks.

2.2 Router Testing

One of the first works performed for router testing is proposed by Aktouf et al. [28]. They use IEEE 1149 boundary scan standard (JTAG standard) to transfer test data in router ports and as wrapper for a group of routers-under-test. All routers are full scan. In this way, the full scan provides test access to inside of the routers. The authors of [29] propose to use partial scan for each router and a single modified IEEE 1500 standard as wrapper for the whole NoC. In this method, they have used a single wrapper that covers the whole NoC. Liu et al. [30] and Cota et al. [31] propose to reuse NoC as TAM for transferring test data to its own routers. For this reason, they assume that the links between routers and network interfaces are tested before and they are fault free. Amory et al. [32] propose a scalable methodology for testing NoC routers by using scan chains and a specific router configuration during test so that the same set of vectors can be broadcast to all routers while responses can be compared within the chip. Grecu et al. [33] propose two schemes for testing the combinational blocks of the NoC routers based on unicast and multicast transmission of test data packets through the switching fabric.

The above structural testing approaches are not at-speed and cannot be applied in online mode and are not suitable for detecting delay faults. Stewart and Tragoudas [34] proposed a functional fault model and a functional test strategy for routers and NIs of regular NoCs with a grid topology. However, their fault model is not at the gate level and they do not provide any data on low-level fault coverage. More recently, Zheng et al. [35] proposed an accelerating technique for a functional test scheme. The test strategy is based on the technique proposed by Raik et al. [36] and assumes an external tester and the definition of several test path configurations to exercise all possible communications in the network. Lubaszewski et al. [23] proposed a post burn-in testing for both links and routers, which is based on the functional testing of several 2×2 meshes in an $N \times N$ NoC. However, both these techniques are offline and suitable for manufacturing testing using automatic test equipment, and to get a reasonable fault coverage and implementing them in hardware leads to overhead.

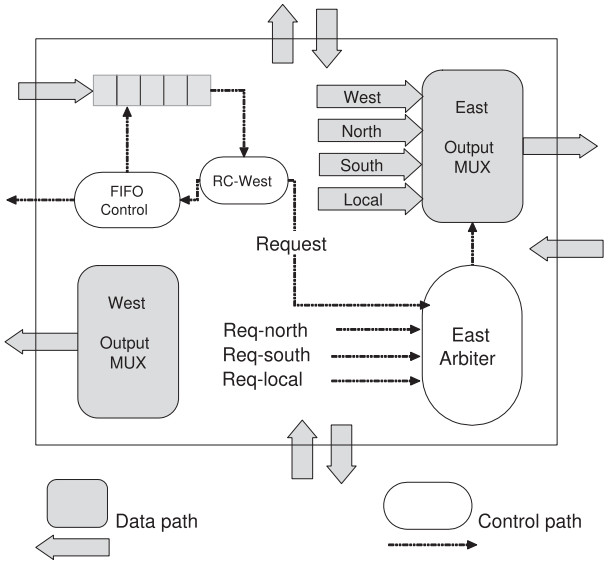

Fig. 1. Xpipes switch used in this work (RC: Routing Computation).

Lin et al. [37] propose a built-in self-test and self-diagnosis architecture to detect and locate faults in the FIFOs and in the MUXes of the switches. Unfortunately, they do not cover the control path of the router.

Strano et al. [38] propose a built-in self-test/self-diagnosis procedure at startup of the NoC by exploiting the inherent structural redundancy of the NoC architecture in a cooperative way. They detect faults in test pattern generators and achieve a high coverage. However, their approach is not online, and moreover, they do not discuss about the format of test patterns to obtain a good fault coverage. In addition, they do not consider the reliability of testing components.

3 NoC Case Study

Without lack of generality, in this work, we base our analysis on a packet-switching network model, called Xpipes, introduced in [39]. Each router in the Xpipes network is composed of N input and M output ports, and for our work we consider a typical router with N and M equal to 5 as shown in Fig. 1. Note that, in this figure, we show only one input port (West) and one output port (East); other input and output ports have the same architectures.

The switch contains control path and data path as shown in the figure. Data path includes interswitch links (except flow-control signals), output multiplexers (crossbar), input/output buffers, and switch's data path wires. Control path consists of routing computation (RCs), arbiters, control logic of the buffers, control signals inside the switch, and interswitch flow-control signals.

One pair of input/output ports is dedicated to the connection between the router and the core, while the remaining four pairs connect the router with the four adjacent routers. Switching is based on the wormhole

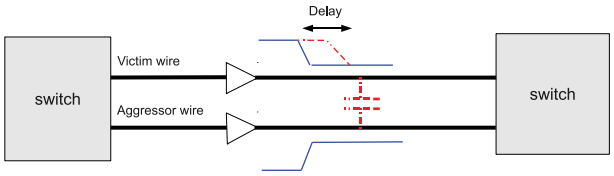

Fig. 2. Delay fault due to the cross-talk coupling.

approach, where a packet is partitioned into multiple flits (flow control units): a header flit, a set of payload flits followed by a tail flit. Flit is the smallest unit over which the flow control is performed and its size equals the channel width. For the case-study NoC topology, the communication channels between routers are defined to be 32-bit wide. Without loss of generality, we consider a 2D mesh as our topology with XY routing, where a packet is first routed on the X direction and then on the Y direction before reaching its destination. However, our testing methodology can work with any topology and routing algorithm as described in Section 5.6.

4 NoC Fault Space Model

In this section, we describe the fault models we used for this work. We classify faults into two groups: logic faults and delay faults. Both of these faults may lead to a system failure, which shows itself in the NoC's operation.

4.1 Logic Fault Model

We define logic fault as a hardware failure in different locations of the NoC. This hardware failure may result in a completely wrong behavior in the NoC functionality. To demonstrate the effectiveness of our testing mechanism, we consider realistic gate-level fault models. We consider two types of gate level faults: stuck-at faults, and bridging faults.

Stuck-at Fault. Stuck-at faults are the most common logic faults at the gate level. They are actually shorts to VDD (stuck-at-1) and shorts to Ground (stuck-at-0). Both routers and links are subject to these types of faults. Any wire in the links is susceptible to stuck-at faults. Stuck-at faults can occur in any component in the router including FIFOs, routing logics, MUXes, arbiters, and wires. In this work, we cover stuck-at faults in both routers and links.

Bridging fault. Bridging faults can occur mostly on the interrouter links as well as the internal wires of the router. They are shorts between wires of a link. These types of faults are different from stuck-at faults and need different test patterns to be detected. In this work, we also cover bridging faults in links.

4.2 Delay Fault Model

Delay fault is not a logic failure, but a timing failure where a combinational path between two stages of flip-flops has higher delay than expected. The delay fault shows itself only at speed while design is working with the target clock frequency. Delay faults are mostly due to aggressive place and route [40], circuit aging, PVT variations, and transistor wear out [41], [42]. Fig. 2 shows a delay fault on an interswitch wire (victim) caused by the cross-talk coupling effect of an aggressor signal (e.g., clock).

TABLE 1
Number of Faults Inside Different Switches

	# of faults		
Switch size	*Stuck-at*	*Transition delay*	*Path delay*
5x5	17080	14880	$8.6 * 10^5$
8x8	35546	32223	$1.5 * 10^6$
10x10	60608	55344	$> 2 * 10^6$
12x12	100257	92452	$> 2 * 10^6$
15x15	165335	155379	$>> 2 * 10^6$

Scaling down of process technologies has increased process variations and transistor wear out. Because of this, delay variations increase and impact the performance of the design. Delay faults in deep submicron are one of the primary reasons of system failure and need to be detected [43]. At-speed functional testing is one of the well-known methods to detect delay faults [44] and we are going to use this technique in our NoC testing approach.

There are two classical fault models that have been developed in recent past to represent delay defects: transition delay fault and path delay fault [45]. Transition delay fault assumes a large delay defect concentrated at one logical node such that any signal transition passing through that node will be delayed past the clock period. The transition delay fault model may miss the distributed and small delay defects in a combinational path which is more common in deep submicron than having a large delay on a single node. The fault list and coverage metrics of the transition delay faults are similar to those of stuck-at faults. Therefore, the stuck-at fault tools can support transition delay faults with minor modifications [46].

On the other hand, path delay assumes a distributed delay along the combinational path. A path is a sequence of connected gates between two stages of clocked elements or between IOs and flip-flops. A path delay fault is said to have occurred if the delay of a path (including the delay of wires and gates) is more than the specified clock period of the circuit. The number of paths in a circuit grows exponentially with circuit size, and hence, typical circuits contain millions and millions of paths making it impossible to test all the paths in the circuit. Table 1 shows the number of faults in various fault models inside different NoC switches ($M \times N$ switch has M input ports and N output ports). As can be seen, the number of path delay faults is much more than stuck-at and transition faults.

Due to large number of timing paths, a practical approach to detect path delay faults is identifying a set of critical paths using a timing analyzer tool and applying functional test vectors to the design and measuring the output using operational clock frequency [46]. We use a similar technique in this work .

4.3 Fault Location

Considering a NoC as a network of routers and NIs, a logic or delay fault can occur in different places including: routers, links, and NIs. In this work, we focus on the routers and links. We assume that NIs are already tested or can be tested together with cores. Faults in the links can be detected using a small number of test patterns. However, to

fully test the routers several test patterns are needed. Faults in router can occur in two main parts: data path and control path. Data path contains flip-flops in FIFOs, router wires, and output MUXes; control path includes routing logics, FIFO's control part, and arbiters. In this work, we cover faults in both control and paths.

4.4 System-Level Failure Modes

As our approach is at-speed functional testing, we need to define some system level failure modes to better tune the test patterns that can give us higher gate-level fault coverage. Note that these system level failure modes help us only to generate better test patterns, but the quality of testing approach is measured considering gate-level fault coverage.

Gate level faults may have different effects on the functional behavior of the system. They can put the system into a failure mode. In case of routers and links, as proposed in [47], we categorize the system level failures into four modes: dropped data, corrupted data, misrouting, and multiple copies. We will describe each failure mode in more details in the following.

Dropped data. In this failure mode, the switch receives the data, but never sends it to the intended output port. Dropped data failure can cause by any fault in the control path of FIFOs, arbiters, or multiplexers of the switch. For instance, consider a case where the head counter of the FIFO is faulty and its current value is increased by 2 or more each time a buffered flit is being removed from the head of FIFO to be sent to its appropriate destination. In this situation, some of the flits in the FIFO will be lost and never exit the switch. Faults in arbiters may cause the select signals of the output multiplexers to not be activated and, therefore, the flit will be dropped. If the output multiplexer is faulty, for example, one or more bits of its select signal are stuck at one or zero, one of its unintended inputs is selected and consequently the original flit will be dropped. Also, a delay fault on the output multiplexers or on the interswitch links will lead to a dropped data failure. Detecting faults related to this failure mode is not easy and requires several test patterns probing all possible combinational paths.

Corrupted data. In this failure mode, the switch receives the data, but it corrupts the data and sends it to the intended output port. This failure mode can also happen on the links. Any logic fault in the data path of FIFOs (flip-flops), multiplexers, links, and internal wires may lead to this failure. As this failure mode can only be created by logic faults in the data path of the switch and links, faults related to this failure are easier to detect compare to those related to the dropped data.

Misrouting. Misrouting is the result of the faulty behavior of the switch router. Assume that the router has a faulty behavior and makes wrong decisions while routing its incoming data. As these faults happen in the control path of the switch, several test patterns are needed to cover all possible routing paths. Logic faults in the header bits of the FIFOs that are related to the destination address can also create this failure as they may change the intended output port. However, these faults are already covered using the patterns related to the corrupted data failure.

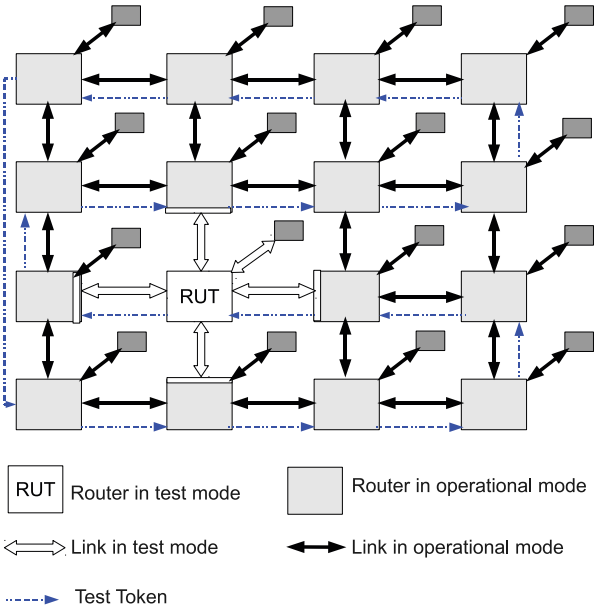

Fig. 3. Overall architecture of our online NoC testing on mesh. Each router which gets a token can start testing itself with the help of its neighbors. Only one router and its links are in the test mode and others are in the operational mode.

Multiple copies. Multiple copies in time failure are originated from faults in FIFOs control path. Consider a FIFO with a faulty "empty" signal. The false empty signal may cause the FIFO send its old data out of the switches port. These old data are usually an earlier packet's flits, and sending the same sequence of flits out of the switch's ports leads to repetition of a packet, i.e., multiple copies in time. The faults related to this mode can be detected easily using test response analyzer that checks the order of received flits in a test packet.

5 The Proposed NoC Online Testing

We propose an at-speed functional testing technique which is capable of detecting logic and delay faults in routers and links of any network with minimum overhead on NoC performance. Fig. 3 shows the overall architecture of our online NoC testing on a mesh.

As can be seen, in our approach only the router under test (RUT) and its links will put in the test mode, while other parts of the network are in the operational mode. All the packets that want to traverse through RUT will be held in the RUT's neighbors until the test is finished. This may have a little latency degradation for some packets. However, we use a token-based technique to make sure that only one router is under test at any given time. In other words, two or more routers cannot be put in test mode simultaneously. Token is simply 1 bit traversing in the network. When a router receives the token, it decides whether it wants to test itself or not; this can be done using a counter in the switch or a request from the core connecting to it. After finishing the testing, the current switch passes the token to the next switch. As token is only 1 bit, we can use TMR for its reliability to make sure all the routers receive the token correctly.

TABLE 2
Acronyms

Acronym	*Meaning*
TPG	Test Pattern Generator
TRA	Test Response Analyzer
FDM	Fault Diagnosis Module
RUT	Router Under Test
TR	Test Response

When a router is in test mode, it should be tested by its neighbors. Each router contains three main test blocks: *test pattern generator*, *test response analyzer*, and *fault diagnosis module*. TPG is responsible for generating test patterns and sending them to the neighbor router which is under test (RUT). TPG of the local port which is inside the router under test's NI sends test packets to the neighbors. Test response analyzer receives test patterns from the neighbor that is under test and analyzes them to detect any fault in the corresponding router and links. TRA of the local port, which is inside router under test's NI, receives test packets sent by neighbors and verifies them. Fault diagnosis module gets the status signals from the neighbors' TRAs as well as the local TRA and diagnoses the faulty channel. There is only one fault diagnosis module in each router. Since only one router can be tested at any given time, TPG and TRA can be shared between all the ports (except the local port). Therefore, for our testing mechanism, we need one pair of TPG and TRA inside each router and each NI. Also, we need one FDM for each router.

Table 2 shows the list of acronyms used during the rest of the paper.

5.1 Test Architecture for Fault Detection

As mentioned before, in our testing approach each router is tested with the help of its neighbors. Thus, this mechanism is scalable to any size of the network with any topology. The architecture of testing one router with four neighbors and one NI is shown in Fig. 4.

When a router is under test, all its neighbors send the test packet generated by their TPG to it. Also, RUT sends test packet to the neighbors using the TPG inside its NI. This enables us to test the local channels connected to the NI as well. At all output ports (except the local) of the routers, we add a 2×1 multiplexer, which selects data from either crossbar or TPG. This additional multiplexer is quite small and we use TMR for its reliability.

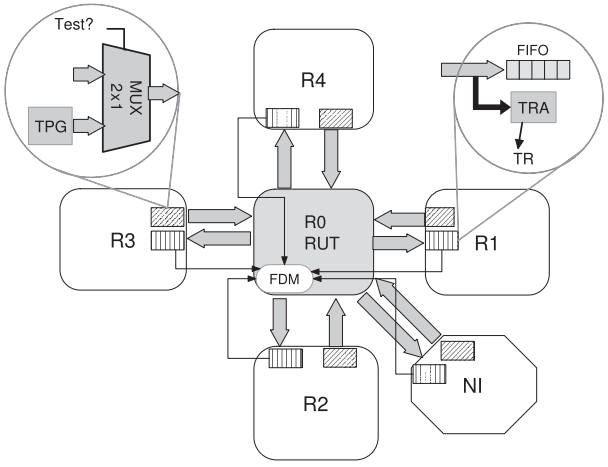

Fig. 4. Each router and its links are tested using neighbors (TPG: Test pattern generator, TRA: Test response analyzer, FDM: Fault diagnosis module).

TABLE 3
Two-Bit TR Signal Generated by Test Response Analyzer

TR (bits)	*Meaning*
00	No Error - Data is received correctly
01	Error - Data is received but is corrupted
10	Error - Data is not received (Dropped)
11	No Info - (reserved for unconnected ports)

TPG generates a sequence of predefined packets based on the test phases. We will describe test phases in details in Section 5.3. The packets generated in different phases are the same in terms of data flits. The header flit of the packet depends on the test phase. Therefore, the same data are sent to different destinations in different phases. The rationale behind this is that the data flits in each packet are responsible to cover the faults in data paths and different phases of testing cover control paths' faults. We will describe the format of test packet in more details in the next section.

TRA verifies whether the incoming data corresponds to the predefined sequence generated by TPG or not. TRA sends a 2-bit signal (TR) to the RUT, which shows the result of pattern checking. RUT receives TR signal from all neighbors and its NI, and diagnoses the faulty channel using fault diagnosis module which is described in Section 5.5. In addition to the corrupted data checking, TRA is able to check the dropped packet failure. In each phase of the test, TRA waits for a specific pattern to arrive. If it receives the packet within a limited number of cycles, it checks the data and sends the results to the RUT. If TRA does not receive any packet, it generates a time-out error. For time-out checking, we use a 5-bit counter inside TRA. If the counter reaches the limit, TRA generates a time-out signal (1 bit of TR) and sends it to the RUT. Table 3 shows the meaning of different bits of TR signal generated by TRA. Note that, since TR is only 2 bits, parity checks can be used for its reliability with a small area overhead.

5.2 Test Packet Format

As mentioned earlier, TPG generates the same packet targeting different destinations at different phases during test of the router. The format of the data flit in the packet is chosen so that it covers all stuck-at and bridging faults in the data path which includes links, FIFO's flip-flops, switch internal wires, and output MUXes. This format is independent from the topology of the network and from the RUT's number of ports.

Two data flits of All-One and All-Zero are enough to cover all stuck-at faults in the data path. However, to detect all bridging faults, we need a set of Walking-One data flits.[1] Therefore, the number of data flits required for bridging

1. Walking-One refers to a set of data patterns where, in each data word, a single bit is set to one and the rest to zero.

Header-flit (different for each phase)	Data-flit All-Zero	Data-flit All-One	32 Data-flits Walking-One (Optional)	Tail-flit

Fig. 5. Format of test packet generated by TPG and verified by TRA at each phase of testing. This format is fixed and independent from network topology.

faults is equal to the flit-width, i.e., 32 in our switch. As a result, to cover all stuck-at and bridging faults in the data path, we need at least 34 data flits. If we target only stuck-at faults, the number of data flits will decrease to 2. This can help to reduce the overhead of both TPG and TRA and to decrease the latency overhead as well. We will show this tradeoff, in more details, in experimental results section. Fig. 5 shows the packet format which is generated at each phase of testing in TPG. Note that in TRA the same packet format is verified at each phase of the testing.

Since all packets are sent with headers and tails, faults may affect those flits, thus modifying the packet routing. When this occurs, the packet can be routed to any other node of the NoC or a tail flit may not be received. In both cases, TRA at the target node will not receive the packet or the tail within a predefined time interval, thus reporting a time-out error.

Since the network is in the normal mode of operation during the test, one might consider the possibility of other errors caused by the logic fault, such as a deadlock, in addition to packet losses and payload errors. We note that a fault cannot cause a deadlock within our test configuration because of the packet format and paths established during testing.

5.3 Test Phases

As described earlier with the specified packet format, we are able to detect all stuck-at, delay, and bridging faults in the data path traversed by the packet. However, to provide coverage for faults in the control path we need different test rounds or phases. Each phase is responsible to cover some of the combinational paths inside the router, while the data path is being tested multiple times. These phases depend on the network topology and the RUT's number of connected ports. In the following, we describe the suitable phases for a router with four neighbors and one NI. In Section 5.6, we discuss how we can modify them for other topologies.

For a router with four neighbors and one NI, we consider nine different phases to cover all the control paths including different routing, arbitration, and FIFO's control. Fig. 6 shows these phases. Arrows in the figure show the source and destination of the test packet. At each phase, one test packet is sent from source to the destination through RUT (R0 in this figure) and its links. Those test packets that come from the neighbors are generated at the output port of the neighbors using their TPGs; while, packets whose sources is RUT are generated in the TPG of the RUT's NI. Therefore, packets originated from RUT also cover the local input channel. Similarly, those packets targeting the neighbors are verified immediately at the input port of the neighbor and before the FIFO. While packets whose destinations are RUT are verified in the RUT's NI. Thus, packets targeting RUT cover the local output channel as well.

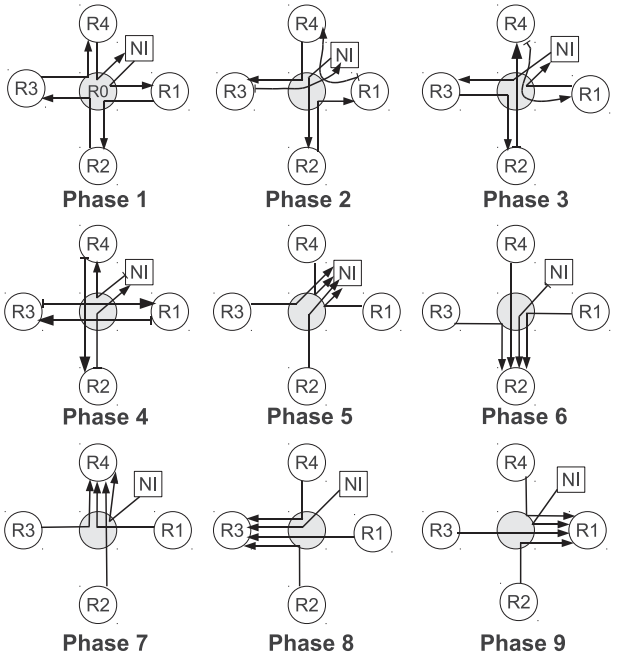

Fig. 6. Nine test phases for a router with four neighbors and one NI. Phases 1 to 4 cover data path and all routings without any arbitration. Phases 5 to 9 cover arbiters as well as FIFO's control logic.

As can be seen in Fig. 6, test phases are chosen in such a way that they cover all functional failures of the router related to the misrouted data, dropped data, and multiple copies failure modes, which are the results of logic faults in the RC, FIFO's control path, or the arbiter.

Since for each input port there could be four different destinations, we need at-least four test phases to fully cover all the routing logics. Test phases 1 to 4 cover all different routing possibilities inside the switch without any arbitration policy. These test phases (1 to 4) are chosen so that we can test all RC blocks inside the router with a minimum number of tests. However, in phases 1 to 4 the packet's destination (stored in header flit) is chosen so that for each output port of the RUT there is only one request. Thus, although some parts of the arbiter are being tested during these phases, these phases (1 to 4) are not sufficient to test the functionality of the arbiters. Also, they cannot fully test the control logic of the FIFO. The FIFO's control logic decides based on the arbiter grants. During these test phases, the arbiter grant is always given to the requester immediately and, therefore, the output's busy signal which goes to the FIFO's control logic is always zero. Thus, in these phases FIFO's control logic always pop the data out of the buffer without waiting. We need more test vectors to put the FIFO in the waiting state.

Both arbiter and FIFO control block can be covered by a set of test packets targeting the same destination. As we have five output and input ports, we need at least five test phases to cover all arbiters and FIFOs. Test phases 5 to 9 are created to perform this task. At each of these phases, there are four requests to the related arbiter. We should note that, there can be some other phases to completely test arbiters where there are two or three requests to each arbiter. However, we ignore since sending four (maximum) requests to each arbiter can test most of its functionality. In

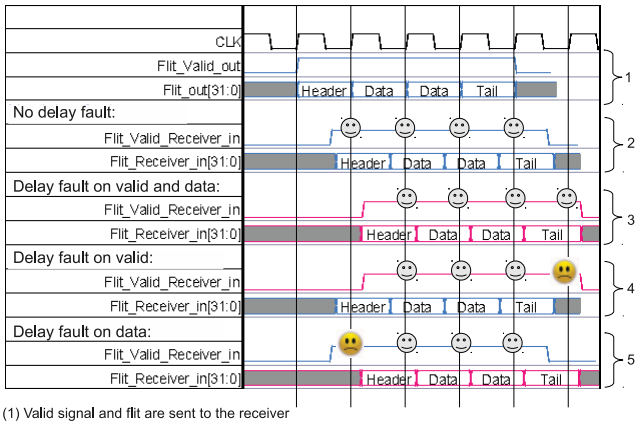

Fig. 7. Detecting delay faults which occurred in links and the router's data path.

Section 5.6, we show a general formula of the number of phases based on the number of ports.

As can be seen in Fig. 6, at each phase of this set (phases 5 to 9) one arbiter together with four FIFOs are tested. At the end of these phase, each arbiter is tested once and each FIFO (both control and data paths) is tested four times. These test phases (5 to 9) are mainly responsible to cover the dropped data, misrouting, and multiple copies failure modes. In the TRA, at each phase we check whether all the packets are arrived or not. In these phases of the test, TRA should receive four consecutive packets from four different sources provided that there were no time-outs in the first four phases. If it does not receive the related packets, it means that the arbiter has not given the grant to one of the requests and TRA generates time-out error. We note that if there are time-outs in the first four phases, TRA does not check it again in phases 5 to 9 and only validates those packets that are supposed to arrive.

5.4 Delay Faults

Since our testing technique is at-speed, it can detect the delay faults in the functional paths. Delay faults can happen in both control and data paths. In the following, we describe how our technique handles delay faults in different functional paths:

5.4.1 Control paths

Based on our timing analysis there are two critical control paths in our switch: 1) request to the arbiter for an output port and 2) grant from the arbiter.

Request paths to the arbiter. If there is a new packet, a request signal is sent to the related arbiter from routing computation block or from the FIFO controller (depending on the implementation). If a delay fault occurs on the path related to request signal, it does not affect the functionality of the router because the arbiter will get the request one clock cycle later. This is because the request signal remains high until the grant is given by the arbiter.

Arbiter's grant paths. If a grant is given by the arbiter, the related FIFO controller pops out the flit from the buffer and the arbiter will drive the appropriate signaling of the crossbar to send the flit out of the switch. Here, if a delay fault occurs on the path from arbiter to the crossbar, the related flit is not sent out of the switch and since it is already popped out of the buffer, we miss that flit. This delay fault will eventually lead to a dropped packet failure, which can easily be detected in TRA.

Other control paths inside the switch are not critical enough to be considered for the path delay fault testing. However, if a huge delay fault occurs in these paths and leads to a functional failure, the packet may be either dropped or misrouted and the fault will be detected in TRA.

5.4.2 Data Path and Links

If a delay fault occurs on the router's data path or on the links, depending on the path where the delay fault occurred the receiver may or may not capture the right data. Fig. 7 shows the timing diagram related to delay fault in data path and links. As can be seen, if the same delay fault occurs on both control signal (*valid*) and *flit* bus the receiver is able to capture the right data one or more cycle after the expected cycle. In this case, since TRA waits for the expected packet to arrive, it can get the data eventually (before the timeout) and does not generate any error. However, if the delay occurs only on *valid* signal or only on *flit* bus, as shown in the figure, it leads to a dropped packet failure that can be detected in TRA. (Note that, we say a delay fault occurs on the *flit* bus if there is at least one delay fault on the bus's bits.)

This timing behavior is valid for both input and output links of RUT. For input links, the receiver is the RUT buffer and fault may occur on input wires; for output links, the receiver is the TRA of the related neighbor and fault may occur either on the data path or on output wires of the RUT.

5.5 Fault Diagnosis

Using TPG, we generate test packets in different test phases and in TRA, we verify the incoming packet and detect if there is any error in it (either corrupted data or misrouted). TRA sends the results of checking to RUT using 2-bit TR signal. However, we still need a mechanism to diagnose the location of the fault. We perform this using fault diagnosis module inside RUT.

For data path faults that are related to the corrupted data failure, FDM is able to diagnose fault location in the input or output channels of RUT. We define input channel as the data path covering the input link and the FIFO, and the output channel as the path covering internal link, output MUX, and the output link. Fig. 8 shows input and output channels for West and East ports. Using 2-bit TR signals, FDM is able to find out which channel (either input or output) is not faulty.

For a 5×5 router, we have a 10-bit register that shows the status of each channel; we name this register channel status register (CSR). Each bit of CSR corresponds to one channel as shown in Fig. 9. FDM receives TR signals and in each phase of the test updates CSR. FDM is based on finding nonfaulty channels.

As seen in Fig. 6, each test packet covers two channels considering local channels. Thus, if a TR signal shows a corrupt error (01), FDM cannot diagnose which of the two

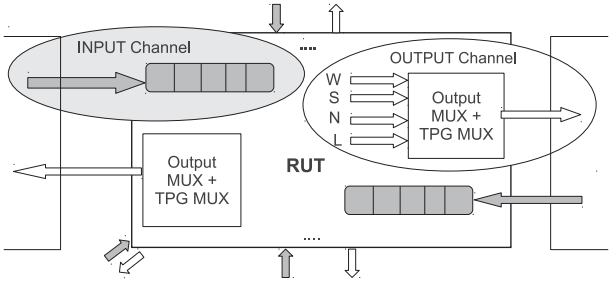

Fig. 8. Channels in which the fault diagnosis module is able to diagnosis data path faults.

channels is faulty. However, if a TR signal shows no error (00), FDM can diagnose that both channels are nonfaulty. At the beginning of the test, FDM sets all the bits in CSR to "0" meaning all channels are faulty. During the test, when FDM receives a TR signal (from either any neighbor or its NI) that shows no-error (00), it sets the corresponding bits of the CSR to "1," which are related to the channels in which the packet has traversed meaning they are nonfaulty. If the bit is already set to "1," it does not change it. At the end of the testing, those bits of CSR which are "1" show the nonfaulty channels. FDM is a simple combinational block which is decided based on the TRs and the testing phase.

FDM is also able to diagnose which RC or arbiter is faulty. We have two 5-bit registers showing the status of RCs and arbiters in a 5 × 5 router; we call these registers routing status register (RSR) and arbiter status register (ASR), respectively. At the beginning of the test FDM sets, all bits of both registers to "1" meaning all the RCs and arbiters are nonfaulty. During the first four phases, if the FDM receives a TR signal of dropped error (10), it sets the corresponding bit in RSR to "0," which is decided based on the test phase. During phases 5 to 9, if FDM receives a TR of dropped error, it sets the corresponding bit in ASR to "0." As mentioned earlier, if TRA detects a time-out error in the

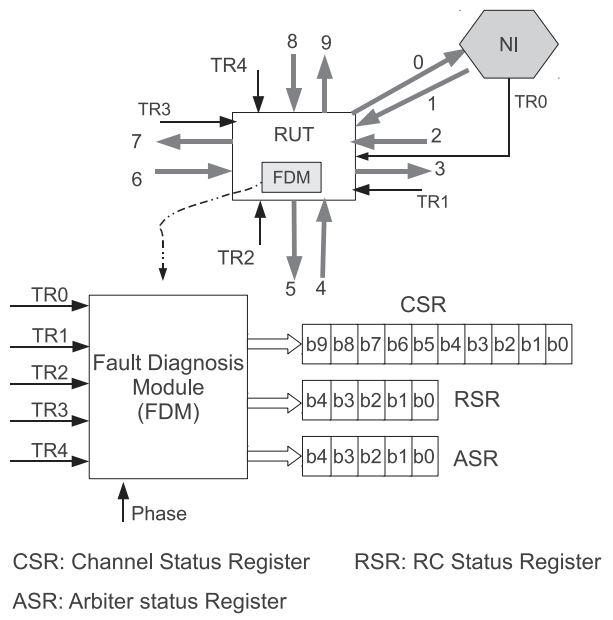

Fig. 9. FDM gets TR signals and updates CSR, RSR, and ASR.

Header-flit	Header-flit	Data-flit All-Zero	Data-flit All-Zero	Data-flit All-One	Data-flit All-One	Tail-flit	Tail-flit

Fig. 10. Packet for bypassing delay faults in data path and links.

first four phases, it does not check it again; therefore, if there is a fault in one RC, it does not have any effect in the results of phases 5 to 9. Thus, if TR shows a dropped error (10) in phases 5 to 9 it is related to the arbiter but not to the RC.

5.5.1 Distinguishing Delay Faults

During the first four phases of the test, if FDM detects a time-out error, the failure is caused by either a delay fault in router/links or a logic fault in RC. We cannot differentiate these faults from each other using the current test pattern and phases. This is because both these faults (delay fault in router/links and logic fault in RC) lead to a dropped packet failure mode. Therefore, with the current test pattern we are only able to detect delay faults, but not to diagnose their locations.

To distinguish delay faults from logic fault in RC, we can use a dedicated test packet and repeat all nine phases with this new packet. To decrease the testing time, we can repeat only those phases that did not pass the test. This requires a modification in TPG, TRA, and FDM to store the pass/fail information. The new packet is formatted so that it can bypass the delay faults, but is still vulnerable to the logic faults. Fig. 10 shows the format of the new packet that can bypass the delay faults in data path or links.

As can be seen in the figure, in the new packet each flit is repeated one time. Note that TPG and TRA should be a little modified to accept two similar consecutive flits during the second round of test phases. Fig. 11 shows how this new packet can bypass the delay faults on the *valid* signal and on the *flit* bus.

By repeating all nine phases (or just the failed phases) but with the new packet, we are able to distinguish delay faults in data path/links from logic faults. For example, if in phase 2 FDM receives a dropped packet signal but does not get it in the second round (phase 2 with the new packet), it means the dropped packet failure was due to a delay fault in the data path/links and not to the RC's logic fault. To store the information related to delay faults in each channel, we also need another 10-bit register that stores the delay faultiness status of the channels.

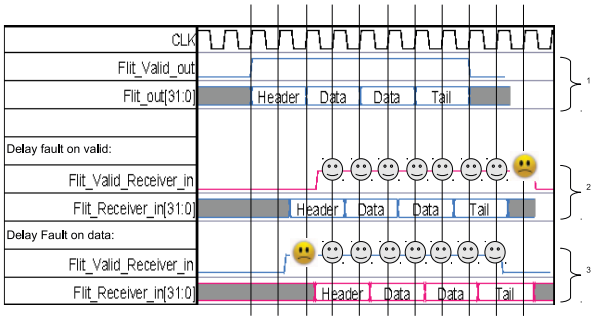

Fig. 11. Delay faults on data path and links are bypassed with the new packet.

The main reason for distinguishing logic faults from delay faults, especially the delay faults on the interswitch links, is the possibility to recover from them by postsilicon tuning techniques or by adjusting the clock frequency. An interested reader can refer to [48], [49] for more information about tuning techniques.

5.6 Testing of Custom Topologies

Our testing methodology is based on testing each router using its neighbors and the NI connected to it. Each router receives TRs from the neighbors as well as its own NI and updates the related status registers. Considering the switches at the boundary of the mesh, there are one or two ports which are not connected to any neighbor. We connect TRs of those ports to "11" meaning no information is available on those specific ports. FDM module does not update anything based on the TR of those unconnected ports. On the other hand, neighbors that are connected to the boundary switches expect test packet from those ports of the RUT which are not connected to any other router. Therefore, they generate time-out error for those specific ports and RUT sets the related bits of RSR to "0" meaning those ports that are not connected to anywhere are faulty. This does not have any effect on the functionality of the router as those ports are not used. Based on this mechanism, our testing technique can be used for any topology and any routing.

TPG and TRA modules should be modified based on the routing algorithm. TPG should send test packet to all output ports of the RUT in different phases. If the routing algorithm and the maximum number of ports in the switch are known, TPG and TRA are the same for all routers and independent from the topology. However, they can be optimized based on the NoC topology to reduce their hardware overhead. For example, TPG does not need to send the test packet to those ports of the RUT which are not connected to any other router. Note that, sending test packet to those ports does not have any effect on the related bits of status registers because the TRs of those port are already connected to "11." Also, TRA does not need to check the time-out error for those input ports of the RUT which are not connected to any other routers. Moreover, CSR, RSR, and ASR can be optimized based on the number of connected ports.

As mentioned earlier, test phases are chosen so that they can cover all the functional paths of the router. They depend on the number of ports in each router. Phases shown in Fig. 6 are suitable for maximum five input and five output ports. However, if a router inside the NoC has more ports, we need to define more test phases to cover those additional ports. Note that for switches with less than five ports, the phases shown in Fig. 6 are more than enough and may be reduced to a smaller number for optimizing test time and hardware cost. For instance, Fig. 12 shows the optimized test phases for a 3×3 switch.

To give a general formula on the number of test phases based on the number of ports of the switch, we divide phases into two groups: 1) phases for RC testing and 2) phases for arbiter and FIFO control testing.

Let say N is the number of input/output ports inside the switch. Since each port can send data to all other ports, we need $N-1$ phases to test each RC component. RCs can be tested in parallel and, therefore, $N-1$ phases are enough for testing all RCs inside the switch.

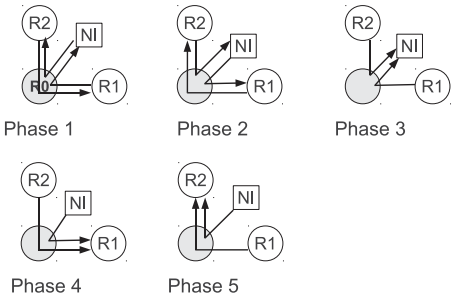

Fig. 12. Test phases for a router with two neighbors and one NI. Phases 1 and 2 cover data path and all routings without any arbitration. Phases 3 to 5 cover arbiters as well as FIFO's control logic.

For the arbiters, the situation is different. Each arbiter can get the request from all other ports; since we have N ports, each arbiter may receive a number of requests in the range of $[1, \ldots, N-1]$. For one request, it is already tested in the previous phases. However, we need to test other situations. For each number of requests $K \in [2, \ldots, N-1]$, K of the input ports send requests to the arbiter. Therefore, we have a *binomial coefficient* $\binom{N}{K}$ for K requests to each arbiter. Thus, the number of test phases for each arbiter is C, where C is the sum of all binomial coefficients as follows:

$$C = \sum_{K=2}^{N-1} \binom{N}{k}.$$

Arbiters cannot be tested in parallel unless we have a complex control logic, and we consider testing them sequentially. Therefore, the number of test phases for all arbiters is $N * C$ and the total number of phases for the router is $M = (N - 1 + N * C)$. Note that, during all these phases links, the data path, and control logics of FIFOs are tested multiple times.

6 Experimental Results

To evaluate our online testing methodology, we used Xpipes [39] which is a packet-switching synthesizable NoC IP. The switch is configured to have five input and five output ports with the flit width of 32 bits and the buffer depth of 2. We used Synopsys design compiler for hardware synthesis and Synopsys Tetramax for logic fault simulation. All designs are mapped on CMOS 45-nm technology from ST-Microelectronics.

6.1 Hardware Overhead

First, we implemented both TPG and TRA in Verilog to see their hardware overhead on a 5×5 router. As mentioned earlier, we can have one TPG and one TRA for the whole switch. We also need one pair of TPG and TRA for each NI. However, TPG and TRA of the NI are simpler than those of the switch because the sequence of test generated/verified by TPG/TRA inside the switch depends on the port while the sequence of those inside NI is fixed. As we mentioned earlier, TPG and TRA are the critical components and we need to use robust techniques to make them reliable. Since

TABLE 4
Synthesis Results of TPG and TRA with Walking-One Sequence (Area of 5×5 Switch + NI $= 8,766\ \mu m^2$)

Module	*Area* (μm^2)	*Area* with TMR (μm^2)	*Area (with TMR) Overhead on 5x5 Switch + NI*
Switch TPG	413	1280	14%
NI TPG	363	1119	12%
Switch TRA	472	1463	16%
NI TRA	410	1270	14%
All TPG + TRA	1658	5132	58%

these module are quite small with respect to the switch and NI, we can use TMR for this purpose.

We implemented two kinds of TPG and TRA. One pair with all-one, all-zero, and walking-one sequence and another pair without walking-one sequence to cover only stuck-at and delay faults. The synthesis results are shown in Tables 4 and 5.

As can be seen, if we want to cover stuck-at, delay, and bridging faults in the 5×5 router the area overhead of all TPGs and TRAs with TMR is about 58 percent, which is pretty high. This is due to the long walking-one sequence that needs a shift register. However, if we target only stuck-at and delay faults the area overhead with TMR is 16 percent, which is acceptable considering TMR support for reliability.

We also implemented the fault diagnosis module together with status registers in Verilog and synthesized them. Based on our synthesis results, the area overhead of the FDM module with TMR is 10 percent on the Xpipes switch. Considering NI and switch together, the area overhead of FDM with TMR is only 7 percent. Therefore, the area overhead of all modules including TPGs, TRAs, and FDM with TMR support is 65 percent to cover stuck-at, delay, and bridging faults and 23 percent to cover only stuck-at and delay faults.

Note that the switch model we used is very small without any output buffer and with an input buffer depth of only two flits and its area is much less than that of the IP cores. For instance, compare to Plasma [50], a small synthesizable 32-bit RISC microprocessor, the switch area is 1/5 of the processor area. Therefore, the overhead on the whole NoC including IPs is very small. Considering switch, NI, and a small sized IP core the area overhead of all modules with TMR is 21 percent to cover stuck-at, delay, and bridging faults and 8 percent to cover only stuck-at and delay faults.

TABLE 5
Synthesis Results of TPG and TRA without Walking-One Sequence (Area of 5×5 Switch + NI $= 8,766\ \mu m^2$)

Module	*Area* (μm^2)	*Area* with TMR (μm^2)	*Area (with TMR) Overhead on 5x5 Switch + NI*
Switch TPG	121	375	4%
NI TPG	73	240	2.5%
Switch TRA	140	462	5%
NI TRA	117	366	4%
All TPG + TRA	451	1443	16%

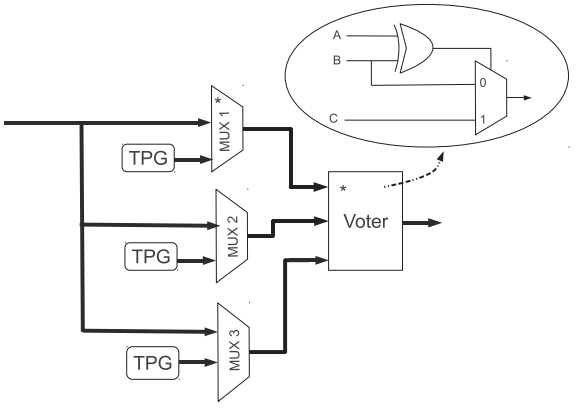

Fig. 13. TMR for TPG and output port.

Test units with their TMR support may have effects on the maximum operational frequency of baseline router. The Xpipes switch is a soft IPs and the timing impact of our testability extensions depends on how it is instantiated. If we do not use any output buffer, the critical path of the router is from the input buffers through output ports and links. As already mentioned and shown in Fig. 4, TPG and TRA operate in parallel with the router and their critical paths are very small, hence they do not decrease the speed of the router. However, as shown in Fig. 13, in case of no output buffers MUXes that are added to the output ports with their related TMR are on the critical path of the router. To have a reliable and high-speed TMR, we used the architecture shown in Fig. 13 for the voter of all TMRs, which is proposed in [51]. We synthesized our testable router and extracted the data related to the delay of the MUX 2×1 and the voter. The nominal delay of the path starting from MUX and ending at the output of the voter is 390 ps. If we consider a baseline Xpipes switch working at 500 mHz (critical path $=$ 2 ns) with no output buffer, we see that the working frequency of the testable switch will be 418 mHz (critical path $=$ 2.39 ns), which is a 15 percent speed overhead.

The no-output-buffer configuration of Xpipes is only for low-speed operation. For high-performance NoCs, output buffers are needed to avoid having critical paths through the output links. In this case, adding testability support has minimal overhead. The critical path is from input buffers to the output buffers and the testing components shown in Fig. 13 are placed after the output buffers. Here, the testing components may reduce the clock frequency only if $P_{clk} - D_t < D_l < P_{clk}$, where D_l is the delay of the output link, P_{clk} is the period of the clock without testing components, and D_t is the delay of the MUX 2×1 and the voter which is 390 ps. We should not that the TMR support which is added to the output ports is not only for the testing purposes but is also useful during the normal operation of the router and makes the output ports fault tolerant.

Focusing on power consumption, it is clear that all test-mode components (except MUXes and the voter shown in Fig. 13) are not used during the normal mode of the NoC. TPG and TRA should be turned on only if one of the

TABLE 6
Power Results of Testable Router in Test Mode

Power	*Baseline router*	*Testable router in test mode*	*Difference test mode*
Dynamic (mW)	1.57	2.08	0.51
Leakage (μW)	2.15	2.28	0.13

neighbors is in the test mode, and FDM should be turned on only if the router itself is being tested. Therefore, we use clock-gating to disable switching activities inside these modules during off-state. This helps us to decrease the power consumption of the router during the normal operation. Table 6 shows the power consumption of our testable router when all testing components are active, and Table 7 shows the results when all of them are turned off. As can be seen, during the test mode the difference between dynamic power consumption of the baseline router and that of our testable router (with all testing components and TMR) is only $510\ \mu$W; this difference for the leakage power consumption is $0.13\ \mu$W. As shown in Table 7, during the normal mode our testable router has a very small overhead of $90\ \mu$W on dynamic power and $0.14\ \mu$W on the leakage power.

We should again note that our baseline switch is very small and if we consider a medium sized switch with NI and a small sized IP core, the power overhead of all testing components is negligible even during the test mode.

6.2 Fault Coverage

We used Synopsys Tetramax to evaluate the fault coverage of our testing approach. We synthesized the Xpipes switch and simulated our nine testing phases on it. Then, we collected the corresponding VCD file to perform fault simulation. Table 8 shows the number of stuck-at faults related to each component in the Xpipes switch. As it was expected, the number of faults inside each component is proportional to the area of that component.

We applied the VCD vectors obtained from the simulation of nine phases on the synthesized design in Tetramax to see the fault coverage of the vectors. Fig. 14 shows the related results. These results are for stuck-at faults. Note that, for bridging faults, the test vectors give us 100 percent coverage on the links in the first four phases. As can be seen, the stuck-at fault coverage of the entire switch for the first four phases of the test is 68 percent, which is not quite good. After applying the next five phases (total nine phases), we could improve it to 85 percent, which is an acceptable coverage for a functional test without any traditional test technique like scan.

TABLE 7
Power Results of Testable Router in Normal Mode (Clock Gating Enabled)

Power	*Baseline router*	*Testable router normal mode*	*Difference normal mode*
Dynamic (mW)	1.57	1.66	0.09
Leakage (μW)	2.15	2.29	0.14

TABLE 8
Number of Stuck-At Faults of Different Components Inside the 5×5 Switch Extracted from Tetramax

Component	*% of the Area*	*# of Faults*	*% of faults*
RCs	9%	1980	11.5%
Arbiters	20.5%	4342	25.4%
Buffers	51.5%	5820	34%
Output MUXes	19.5%	4722	27.6%
other gates	0.5%	80	0.1%
Entire Switch	100%	17080	100%

Using the same VCD vectors, we achieved 67 percent coverage for the transitional delay faults. This is because not all the transitional delay faults lead to a functional failure, and the coverage for transition delay faults is always less than that of stuck-at faults.

For the path delay faults, we first imported the synthesized netlist of the switch into PrimeTime, the static timing analysis tool from Synopsys, and extracted those paths whose delays were more than 80 percent of the clock period. Then, we analyzed the extracted paths and found two groups of paths. The first group was related to sending the flit from the input buffer to the output port when the grant is given by the arbiter. As we already discussed in Section 5.4, delay faults on these paths lead to the dropped data failure mode and we detect them in TRAs. The second group was similar to the first one and was related to shifting the FIFO content when the grant is given by the arbiter and the flit located at the head of the FIFO is sent out from the switch. Delay faults on these paths lead to either a flit duplication or a dropped flit. Both of these failure modes can be detected in TRAs.

Note that the achieved fault coverage can still be increased by adding to the test phases other network functional configurations (arbitration possibilities that were not applied, for example). However, insisting in a completely functional test approach will require an increasing number of test configurations and phases, for a decreasing number of detected faults. The tradeoff seems to be not in favor of pushing further the functional testing approach. From the point of view of a functional test, about 10 percent of the remaining logic faults are undetectable because they are related to unreachable control states [22].

6.3 Online Testing Evaluation in Simulation

To see the effect of our online testing mechanism on the latency of the packets, we performed system simulation. We

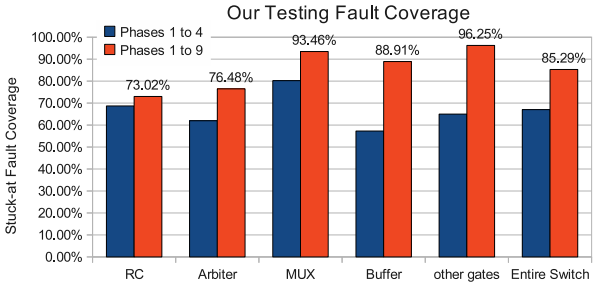

Fig. 14. Stuck-at fault coverage of our online functional test approach.

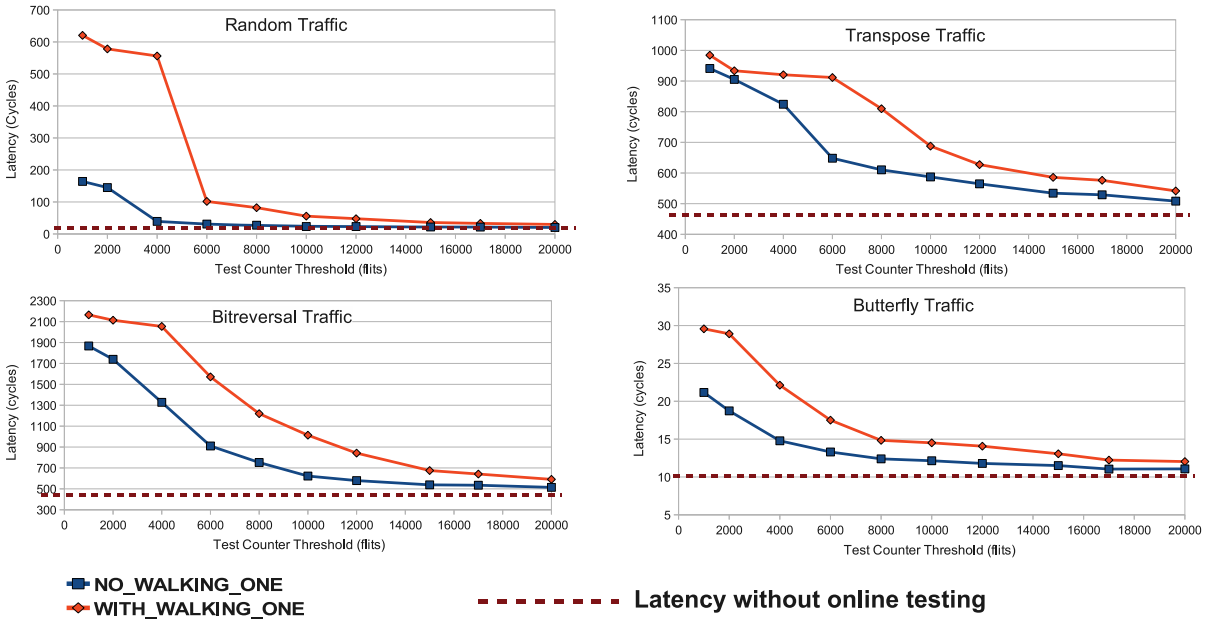

Fig. 15. Simulation results for our online testing approach on different synthetic traffics with various test counter thresholds.

used Noxim [52], a cycle accurate NoC simulator implemented in SystemC. The switch model in this simulator has a two-stage pipeline and, therefore, has two cycles minimum latency. We extended it for performing our online testing approach.

We periodically place a switch under test and hold all the packets traversing it. This is performed by arbiters in the neighbors. They do not give grant to the packets traversing the switch under test, until the test is finished. We compared the average latency of the packets in different synthetic and real traffics with and without online testing. We implemented a token-based mechanism in the simulator to avoid having multiple routers under test at the same time. We have a counter inside each router that counts the number of flits forwarded by the switch. When the counter reaches the threshold value if the router has the token it can start testing. We swept the threshold value from 1,000 to 20,000. The simulation has been performed on an 8×8 Mesh with XY routing algorithm.

Fig. 15 shows the average packet latency of different synthetic traffics in the NoC while using our online testing mechanism. We performed simulation with and without walking-one sequence, which is for covering bridging faults. As can be seen, when the threshold value of the test counter is low (less than 10,000 flits) the latency overhead is quite high, especially when the test packet includes walking-one sequence. This is due to the fact that with lower threshold values the switches go to the test mode more frequently. However, for threshold values greater than 10,000 flits, the latency overhead of our online testing is almost negligible.

In addition to the synthetic traces, we also performed simulation to see the effect of online testing on real traffic traces from the PARSEC benchmarks, a suite of next-generation shared-memory programs for CMPs [53]. The traces used are for a 64-node shared memory CMP arranged as a 8×8 mesh. Each processor node has a private L1 cache of 32 KB and 1 MB L2 cache (64-MB shared distributed L2 for the entire system). There are four memory controllers at the corners. To obtain the traces, we used Virtutech Simics [54] with the GEMS toolset [55], augmented with GARNET [56], simulating a 64-core NoC.

Like for the previous experiments, we swept the test counter threshold from 1,000 to 20,000 flits. We performed the simulation for two types of test packets: with walking-one sequence and without that. The results are shown in Fig. 16. As can be seen, for all benchmarks with a threshold value of 10,000 the latency overhead is almost zero even with walking-one sequence. Based on our experiments, 10,000 flits is an appropriate value for the test counter's threshold for both synthetic and real traffic.

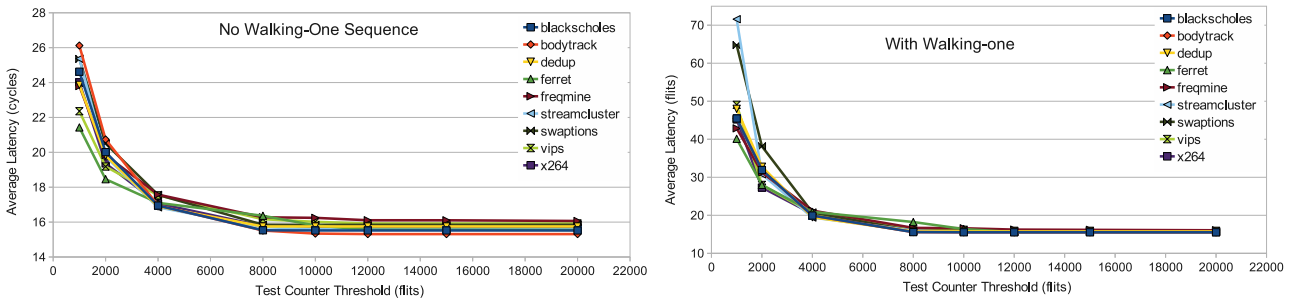

Fig. 16. Simulation results on PARSEC benchmarks with various test counter thresholds.

7 Conclusions

A technique for online detection and diagnosis of faults in NoC's routers and links has been proposed. This method can be applied on any topology and routing algorithm. Each router in the NoC is tested using test pattern generators and test response analyzers in the neighbors and its own NI. During the testing, only the router under test is in test mode, while other parts of the network are operational. The proposed testing mechanism covers all the short wire, and stuck-at faults on the links. It also covers 100 percent of the data path faults and 85 percent of the control-path faults inside the router.

Acknowledgments

This work was supported by EU FP7 project Pro3D (GA n. 248776) and ENIAC project Modern (GA 120003).

References

[1] ITRS Home-2010, Int'l Technology Roadmap for Semiconductors, http://www.itrs.net/home.html, 2010.
[2] http://techresearch.intel.com/articles/Tera-Scale/1421.html, 2013.
[3] http://techresearch.intel.com/articles/Tera-Scale/1449.html, 2013.
[4] Intel, "Intel News Release: Intel Unveils New Product Plans for High Performance Computing," http://www.intel.com/pressroom/archive/releases/20100531comp.htm, May 2010.
[5] L. Seiler et al., "Larrabee: A Many-Core x86 Architecture for Visual Computing," *IEEE Micro,* vol. 29, no. 1, pp. 10-21, Jan./Feb. 2009.
[6] http://www.tilera.com/products/processors.php, 2013.
[7] J.W. McPherson, "Reliability Challenges for 45nm and Beyond," *Proc. ACM/IEEE 43rd Design Automation Conf.,* pp. 176-181, 2006.
[8] S. Borkar, "Microarchitecture and Design Challenges for Gigascale Integration," *Proc. ACM/IEEE 37th Int'l Symp. Microarchitecture (MICRO-37),* p. 3, 2004.
[9] S. Mitra et al., "Robust System Design to Overcome CMOS Reliability Challenges," *IEEE J. Emerging and Selected Topics in Circuits and Systems,* vol. 1, no. 1, pp. 30-41, Mar. 2011.
[10] 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.
[11] T. Bjerregaard and S. Mahadevan, "A Survey of Research and Practices of Network-on-Chip," *ACM Computing Surveys,* vol. 38, no. 1, article 1, 2006.
[12] M.R. Kakoee et al., "ReliNoC: A Reliable Network for Priority-Based On-Chip Communication," *Proc. ACM/IEEE Design, Automation & Test in Europe Conf. & Exhibition (DATE),* pp. 667-672, 2011.
[13] S. Rodrigo et al., "Addressing Manufacturing Challenges with Cost-Efficient Fault Tolerant Routing," *Proc. ACM/IEEE Fourth Int'l Symp. Networks-on-Chip (NOCS),* pp. 25-32, 2010.
[14] D. Fick et al., "A Highly Resilient Routing Algorithm for Fault-Tolerant NoCs," *Proc. ACM/IEEE Design, Automation & Test in Europe Conf. & Exhibition (DATE),* pp. 21-26, 2009.
[15] M.E. Gomez et al., "An Efficient Fault-Tolerant Routing Methodology for Meshes and Tori," *IEEE Computer Architecture Letters,* vol. 3, no. 1, p. 3, Jan.-Dec. 2004.
[16] C.-T. Ho and L. Stockmeyer, "A New Approach to Fault-Tolerant Wormhole Routing for Mesh-Connected Parallel Computers," *IEEE Trans. Computers,* vol. 53, no. 4, pp. 427-439, Apr. 2004.
[17] D. Park et al., "Exploring Fault-Tolerant Network-On-Chip Architectures," *Proc. ACM/IEEE Int'l Conf. Dependable Systems and Networks (DSN),* pp. 93-104, 2006.
[18] W. Song, D. Edwards, J.L. Nunez-Yanez, and S. Dasgupta, "Adaptive Stochastic Routing in Fault-Tolerant On-Chip Networks," *Proc. ACM/IEEE Third Int'l Symp. Networks-on-Chip (NoCS),* pp. 32-37, 2009.
[19] C. Grecu et al., "Online Fault Detection and Location for NoC Interconnects," *Proc. IEEE 12th Int'l On-Line Testing Symp. (IOLTS),* pp. 6-11, 2006.
[20] A. Alaghi, N. Karimi, M. Sedghi, and Z. Navabi, "Online NoC Switch Fault Detection and Diagnosis Using a High Level Fault Model," *Proc. IEEE 22nd Int'l Symp. Defect and Fault-Tolerance in VLSI Systems (DFT),* pp. 21-29, 2007.
[21] Y.H. Kang, T.-J. Kwon, and J. Draper, "Fault-Tolerant Flow Control in On-Chip Networks," *Proc. ACM/IEEE Fourth Int'l Symp. Networks-on-Chip (NOCS),* pp. 79-86, 2010.
[22] E. Cota, F.L. Kastensmidt, M. Cassel, M. Herve, P. Almeida, P. Meirelles, A. Amory, and M. Lubaszewski, "A High-Fault-Coverage Approach for the Test of Data, Control and Handshake Interconnects in Mesh Networks-On-Chip," *IEEE Trans. Computers,* vol. 57, no. 9, pp. 1202-1215, Sept. 2008.
[23] M. Herve, P. Almeida, F.L Kastensmidt, E. Cota, and M. Lubaszewski, "Concurrent Test of Network-on-Chip Interconnects and Routers," *Proc. IEEE 11th Latin Am. Test Workshop (LATW),* pp. 1-6, 2010.
[24] M. Cuviello, S. Dey, X. Bai, and Y. Zhao, "Fault Modeling and Simulation for Crosstalk in System-on-Chip Interconnects," *Proc. IEEE/ACM Int'l Conf. Computer-Aided Design,* pp. 297-303, 1999.
[25] P.P. Pande et al., "Design of Low Power and Reliable Networks on Chip through Joint Crosstalk Avoidance and Forward Error Correction Coding," *Proc. IEEE 21st Int'l Symp. Defect and Fault Tolerance in VLSI Systems,* pp. 466-476, 2006.
[26] T. Bengtsson et al., "Offline Testing of Delay Faults in NoC Interconnects," *Proc. Ninth EUROMICRO Conf. Digital System Design: Architectures, Methods and Tools,* pp. 677-680, 2006.
[27] J. Raik, V. Govind, and R. Ubar, "Test Configurations for Diagnosing Faulty Links in NoC Switches," *Proc. IEEE 12th European Test Symp. (ETS),* 2007.
[28] C. Aktouf, "A Complete Strategy for Testing an On-Chip Multiprocessor Architecture," *IEEE Design & Test of Computers,* vol. 19, no. 1, pp. 18-28, Jan./Feb. 2002.
[29] Y. Wu and P. MacDonald, "Testing ASICs with Multiple Identical Cores," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* vol. 22, no. 3, pp. 327-336, Mar. 2003.
[30] C. Liu, Z. Link, and D.K. Pradhan, "Reuse-Based Test Access and Integrated Test Scheduling for Network-on-Chip," *Proc. Design, Automation and Test in Europe,* vol. 1, pp. 303-308, Mar. 2006.
[31] E. Cota, L. Carro, and M. Lubaszewski, "Reusing an On-Chip Network for the Test of Core-Based Systems," *ACM Trans. Design Automation of Electronic Systems,* vol. 9, no. 4, pp. 471-499, 2004.
[32] A.M. Amory, E. Briao, E. Cota, M. Lubaszewski, and F.G. Moraes, "A Scalable Test Strategy for Network-on-Chip Routers," *Proc. IEEE Int'l Test Conf.,* p. 9, 2005.
[33] C. Grecu et al., "Methodologies and Algorithms for Testing Switch-Based NoC Interconnects," *Proc. IEEE 20th Int'l Symp. Defect and Fault Tolerance in VLSI Systems,* pp. 238-246, 2005.
[34] K. Stewart and S. Tragoudas, "Interconnect Testing for Networks on Chips," *Proc. IEEE 24th VLSI Test Symp.,* p. 6, 2006.
[35] Y. Zheng et al., "Accelerating Strategy for Functional Test of NoC Communication Fabric," *Proc. IEEE 19th Asian Test Symp.,* pp. 224-227, 2010.
[36] J. Raik, V. Govind, and R. Ubar, "An External Test Approach for Network-on-a-Chip Switches," *Proc. 15th Asian Test Symp.,* pp. 437-442, 2006.
[37] S.Y. Lin, C.C. Hsu, and A.Y. Wu, "A Scalable Built-In Self-Test/Self-Diagnosis Architecture for 2D-Mesh Based Chip Multiprocessor Systems," *Proc. IEEE Int'l Symp. Circuits and Systems,* pp. 2317-2320, 2009.
[38] A. Strano et al., "Exploiting Network-on-Chip Structural Redundancy for a Cooperative and Scalable Built-In Self-Test Architecture," *Proc. ACM/IEEE Design, Automation & Test in Europe Conf. & Exhibition (DATE),* pp. 1-6, Mar. 2011.
[39] D. Bertozzi and L. Benini, "Xpipes: A Network-on-Chip Architecture for Gigascale Systems-on-Chip," *IEEE Circuits and Systems Magazine,* vol. 4, no. 2, pp. 18-31, Sept. 2004.
[40] M.R. Kakoee et al., "A New Physical Routing Approach for Robust Bundled Signaling on NoC Links," *Proc. ACM/IEEE 20th Symp. Great Lakes Symp. VLSI (GLSVLSI),* pp. 3-8, 2010.
[41] E. Mintarno et al., "Self-Tuning for Maximized Lifetime Energy-Efficiency in the Presence of Circuit Aging," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* vol. 30, no. 5, pp. 760-773, May 2011.

[42] J. Henkel et al., "Design and Architectures for Dependable Embedded Systems," *Proc. ACM Ninth Int'l Conf. Hardware/ Software Codesign and System Synthesis (CODES+ISSS),* 2011.
[43] P. Songwei, L. Huawei, and L. Xiaowei, "A Unified Test Architecture for Online and Off-Line Delay Fault Detections," *Proc. IEEE 29th VLSI Test Symp. (VTS),* pp. 272-277, May 2011.
[44] S. Natarajan et al., "Path Coverage Based Functional Test Generation for Processor Marginality Validation," *Proc. IEEE Int'l Test Conf. (ITC),* pp. 1-9, Nov. 2010.
[45] A. Krstic and K.T. Cheng, *Delay Fault Testing for VLSI Circuits.* Kluwer Academic Publishers, 1998.
[46] M. Sharma and J.H. Patel, "Testing of Critical Paths for Delay Faults," *Proc. IEEE Int'l Test Conf.,* pp. 634-641, 2001.
[47] T. Bengtsson, S. Kumar, and Z. Peng, "Application Area Specific System Level Fault Models: A Case Study with a Simple NoC Switch," *Proc. IEEE Third Int'l Workshop Electronic Design, Test and Applications (IDT),* 2006.
[48] M.R. Kakoee and L. Benini, "Fine-Grained Power and Body-Bias Control for Near-Threshold Deep Sub-Micron CMOS Circuits," *IEEE Trans. Emerging and Selected Topics in Circuits and Systems,* vol. 1, no. 2, pp. 131-140, June 2011.
[49] A. Ghosh et al., "A Centralized Supply Voltage and Local Body Bias-Based Compensation Approach to Mitigate Within-Die Process Variation," *Proc. ACM/IEEE 14th Int'l Symp. Low Power Electronics and Design (ISLPED),* pp. 45-50, 2009.
[50] Plasma microprocessor description, http://www.opencores.org, 2013.
[51] T. Ban and L.A. de Barros Naviner, "A Simple Fault-Tolerant Digital Voter Circuit in TMR Nanoarchitectures," *Proc. IEEE Eighth Int'l NEWCAS Conf. (NEWCAS),* pp. 269-272, June 2010.
[52] F. Fazzino et al., Noxim: Network-on-Chip Simulator, http://noxim.sourceforge.net, 2013.
[53] C. Bienia et al., "The PARSEC Benchmark Suite: Characterization and Architectural Implications," *Proc. ACM 17th Int'l Conf. Parallel Architectures and Compilation Techniques (PACT),* pp. 72-81, 2008.
[54] P.S. Magnusson et al., "Simics: A Full System Simulation Platform," *Computer,* vol. 35, no. 2, pp. 50-58, Feb. 2002.
[55] M.M.K. Martin et al., "Multifacet's General Execution-Driven Multiprocessor Simulator (GEMS) Toolset," *ACM SIGARCH Computer Architecture News,* vol. 33, no. 4, pp. 92-99, 2005.
[56] N. Agarwal et al., "GARNET: A Detailed On-Chip Network Model Inside a Full-System Simulator," *Proc. IEEE Int'l Symp. Performance Analysis of Systems and Software (ISPASS),* pp. 33-42, 2009.

Mohammad Reza Kakoee received the BS degree in computer engineering from Isfahan University of Technology, Iran, in 1999, the MS degree in computer architecture from the University of Tehran, Iran, in 2003, and the PhD degree in electrical and computer engineering from the University of Bologna in 2012. From 2003 to 2009, he was a research assistant at the University of Tehran, working on hardware verification and system on chips. From June 2010 to December 2010, he was a visiting scholar at the University of Michigan, Ann Arbor. He was a postdoctoral fellow at the University of Bologna from January 2012 to November 2012. In November 2012, he joined Qualcomm Technologies in San Diego, California, as a senior researcher and engineer. His research interests include SoC design and reliability, PVT variation, and low-power design. He has published several papers on these topics in journals and visible conferences. He is a member of the IEEE.

Valeria Bertacco received the computer engineering degree from the University of Padova, Italy, in 1995. She received the MS and PhD degrees in electrical engineering from Stanford University in 1998 and 2003, respectively. She is an associate professor of electrical engineering and computer science at the University of Michigan. Her research interests are in the area of design correctness, with emphasis on digital system reliability, postsilicon and runtime validation, and hardware-security assurance. She joined the faculty at Michigan in 2003, after being in the Advanced Technology Group of Synopsys for four years as a lead developer of Vera and Magellan. During the winter of 2012, she was on sabbatical at the Addis Ababa Institute of Technology. She has served on several conference program committees, including DATE, DAC, and DSN, and has been an associate editor for the *IEEE Transactions on Computer Aided Design* and *Microelectronics Journal*. She is the author of three books on design's functional correctness. She received the IEEE CEDA Early Career Award, US National Science Foundation (NSF) CAREER award, the US Air Force Office of Scientific Research's Young Investigator award, the IBM Faculty Award, and the Vulcans Education Excellence Award from the University of Michigan. She is a senior member of the IEEE.

Luca Benini received the PhD degree in electrical engineering from Stanford University. He is a full professor at the University of Bologna and the chair of Digital Circuits and Systems at ETHZ, Switzerland. His research interests are in energy-efficient system design and multicore SoC design. He is also active in the area of energy-efficient smart sensors and sensor networks for biomedical and ambient intelligence applications. He has published more than 600 papers in peer-reviewed international journals and conferences, four books, and several book chapters. He is a fellow of the IEEE.