In-Field Test for Permanent Faults in FIFO Buffers of NoC Routers

Bibhas Ghoshal, Kanchan Manna, Santanu Chattopadhyay, and Indranil Sengupta

***Abstract*— This brief proposes an on-line transparent test technique for detection of latent hard faults which develop in firstinput first-ouptput buffers of routers during field operation of NoC. The technique involves repeating tests periodically to prevent accumulation of faults. A prototype implementation of the proposed test algorithm has been integrated into the router-channel interface and on-line test has been performed with synthetic self-similar data traffic. The performance of the NoC after addition of the test circuit has been investigated in terms of throughput while the area overhead has been studied by synthesizing the test hardware. In addition, an on-line test technique for the routing logic has been proposed which considers utilizing the header flits of the data traffic movement in transporting the test patterns.**

***Index Terms*— FIFO buffers, in-field test, NoC, permanent fault, transparent test.**

I. Introduction

Over the last decade, network-on-chip (NoC) has emerged as a better communication infrastructure compared with bus-based communication network for complex chip designs overcoming the difficulties related to bandwidth, signal integrity, and power dissipation [1]. However, like all other systems-on-a-chip (SoCs), NoC-based SoCs must also be tested for defects. Testing the elements of the NoC infrastructure involves testing routers and interrouter links.

Significant amount of area of the NoC data transport medium is occupied by routers, which is predominantly occupied by FIFO buffers and routing logic. Accordingly, the probabilities of run-time faults or defects occurring in buffers and logic are significantly higher compared with the other components of the NoC. Thus, test process for the NoC infrastructure must begin with test of buffers and routing logic of the routers. In addition, the test must be performed periodically to ensure that no fault gets accumulated.

The occasional run-time functional faults have been one of the major concerns during testing of deeply scaled CMOS-based memories. These faults are a result of physical effects, such as environmental susceptibility, aging, and low supply voltage and hence are *intermittent* (nonpermanent indicating device damage or malfunction) in nature [2]. However, these *intermittent* faults usually exhibit a relatively high occurrence rate and eventually tend to become permanent [2]. Moreover, wear-out of memories also cause intermittent faults to become frequent enough to be classified as permanent. Thus, there is a need for online test technique that can detect the run-time faults, which are intermittent in nature but gradually become permanent over time.

A. Contribution

In this brief, we have proposed an online transparent test technique for first-input first-output (FIFO) buffers and routing logic present within the routers of the NoC infrastructure. Our contributions are as follows. A transparent SOA-MATS++ test generation algorithm has proposed targeting in-field permanent faults developed in SRAM-based FIFO memories and it has been utilized to perform online and periodic test of FIFO memory present within the routers of the NoC. In addition, we have also proposed an online test technique for the routing logic that is performed simultaneously with the test of buffers. The proposal involves two ways of utilizing the unused portion of the header flits of the incoming data packets in transporting the test patterns. First, deterministic test patterns for the routing logic generated by *Tetramax* are placed in the unused fields of the header flit and are transported during the normal cycle. Second, the pseudorandom patterns in the synthetic data traffic used during normal operation and arriving at the routing logic are considered as test patterns. Fault coverage is estimated for either of the two proposals.

B. Fault Models Considered for the Work

The run-time *permanent* faults considered in this brief are assumed to be *intermittent* faults, which have become permanent over time. Consequently, the fault models considered in this brief are that of intermittent faults. The primary factors that lead to intermittent faults are aging effects, such as time-dependent dielectric breakdown (TDDB), electromigration, negative bias temperature instability (NBTI), and hot carrier injection (HCI), as mentioned in [3]. TDDB is a phenomena where the oxide underneath the gate material of an MOSFET degrades over time resulting in a short circuit, which are modeled as stuck-at-faults [4]. Electromigration reduces interconnect conductivity with passage of time and leads to open circuit [4]. The open circuits caused by electromigration are modeled as stuck-open-faults. NBTI and HCI increase the threshold voltage of transistors leading to decrease in mobility. As a result, the performances of the memory core decreases bringing in read and write failures. The write failures are modeled as transition faults, while read failures are modeled as read disturb faults [5].

To summarize, the target fault models considered for this brief are stuck-at fault, stuck-open fault, read disturb fault, and transition fault. Detailed behavior of these faults can be found in [6].

II. Related Work

As fault tolerance in NoC design has gained importance among research community, a number of papers have been published covering different aspects of fault tolerance, such as failure mechanisms, fault modeling, diagnosis, and so on. A detailed survey summarizing the research work in these papers has been provided in [3]. Over the years, researchers have proposed a number of Design-For-Testability (DFT) techniques for NoC infrastructure testing (testing routers as well as NoC interconnect) [7] and for NoC based core testing [8]. Built-in self test (BIST)-based techniques have also been used for testing routers as well as NoC interconnect, such as [8]. A recent paper on NoC and router testing in [9] provides a summary of the DFT techniques employed for testing NoC interconnects and routers in particular. In addition to novel test architectures, fault tolerant routing algorithms have also been proposed [10].

FIFO buffers in NoC infrastructure are large in number and spread all over the chip. Accordingly, probability of faults is significantly

Manuscript received March 19, 2014; revised September 16, 2014 and November 19, 2014; accepted January 10, 2015. Date of publication March 26, 2015; date of current version December 24, 2015.

B. Ghoshal and I. Sengupta are with the Department of Computer Science and Engineering, IIT Kharagpur, Kharagpur 721302, India (e-mail: bibhas.ghoshal@gmail.com; isg@iitkgp.ac.in).

K. Manna is with the School of Information Technology, IIT Kharagpur, Kharagpur 721302, India (e-mail: kanchanm@sit.iitkgp.ernet.in).

S. Chattopadhyay is with the Department of Electronics and Electrical Communication Engineering, IIT Kharagpur, Kharagpur 721302, India (e-mail: santanu@ece.iitkgp.ernet.in).

Digital Object Identifier 10.1109/TVLSI.2015.2393714

higher for the buffers compared with other components of the router. Both online and offline test techniques have been proposed for test of FIFO buffers in NoC. The proposal in [11] is an offline test technique (suitable for the detection of manufacturing fault in FIFO buffers) that proposes a shared BIST controller for FIFO buffers. Online test techniques for the detection of faults in FIFO buffers of NoC routers have been proposed in [12]. However, the technique considers standard cell-based FIFO buffers, while we consider SRAM-based FIFO designs. Thus, faults considered in this brief are different from those targeted in [12].

To the best of our knowledge, no work has been reported in the literature that proposes online test of SRAM-based FIFO buffers present within routers of NoC infrastructure. Thus, we surveyed online test techniques for SRAM-based FIFOs in general. The survey revealed that SRAM based FIFOs are tested using either of the following two approaches, dedicated BIST approach as proposed by Barbagallo *et al.* in [13] and or distributed BIST proposed by Grecu *et al.* in [11]. However, both dedicated and distributed BIST approaches being offline test techniques fail to detect permanent faults, which develop over time.

III. Proposed Transparent Test Generation

The faults considered in this brief, if applied for SRAMs or DRAMs, can be detected using standard March tests [6]. However, if the same set of faults are considered for SRAM-type FIFOs, March test cannot be used directly due to the address restriction in SRAM-type FIFOs mentioned in [14] and thus we were motivated to choose single-order address MATS++ test (SOA-MATS++) [14] for the detection of faults considered in this brief. The word-oriented SOA-MATS++ test is represented as $\{\updownarrow (wa); \uparrow (ra, wb); \downarrow (rb, wa)\}; \updownarrow (ra)\}$ where, a is the data background and b is the complement of the data background. $\uparrow$ and $\downarrow$ are increasing and decreasing addressing order of memory, respectively. $\updownarrow$ means memory addressing can be increasing or decreasing. Application of SOA-MATS++ test to the FIFO involves writing patterns into the FIFO memory and reading them back. As a result, the memory contents are destroyed. However, online memory test techniques require the restoration of the memory contents after test. Thus, researchers have modified the March tests to transparent March test [15] so that tests can be performed without the requirement of external data background and the memory contents can be restored after test. We have thus transformed the SOA-MATS++ test to transparent SOA-MATS++ (TSOA-MATS++) test that can be applied for online test of FIFO buffers. The transparent SOA-MATS++ test generated is represented as $\{\uparrow (rx, w\bar{x}, r\bar{x}, wx, rx)\}$.

The operations performed during the test represent three phases of the test, namely, *invert* phase, *restore* phase, and *read* phase. The first two operations form a read write pair (rx, wx) representing the invert phase where the initial content (content before start of test) of the FIFO buffer location under test (*lut*) is read and its complement is written back to the same location. The invert phase is followed by restore phase involving the operations ($r\bar{x}$, wx), where the content of *lut* are read and reinverted. At this point of the test, the contents of *lut* have been flipped twice to get back the original content. The last phase, (rx) involves reading the content of *lut* without any write operation to follow.

A. Test Algorithm

The algorithmic interpretation of the transparent SOA-MATS++ test is presented in Algorithm 1. It describes the step-by-step procedure to perform the three phases of the transparent SOA-MATS++ test for each location of the FIFO memory. The target location for test is given by the loop index i that varies from 0 to $N-1$, where N is the number of locations in the FIFO memory. In other words, i represents the address of the FIFO memory location presently under test. For each location, the three test runs are performed during three iterations of the loop index j.

Algorithm 1 Transparent SOA-MATS++ Test Algorithm

```
Require: N = number of rows of the FIFO memory
1:  i ← 0;                                  /* memory address pointer */
2:  while (i ≤ N − 1) do
3:     j ← 0;                               /* test cycle counter */
4:     while (j ≤ 2) do
5:        temp ← read(i);
6:        if (j = 0) then
7:           original ← temp;
8:           write(i, !temp);
9:        else
10:          if (j = 1) then
11:             result ← compare(temp, original);
12:             write(i, !temp);
13:          end if
14:       else
15:          result ← compare(temp, original);
16:       end if
17:       j ← j + 1;
18:    end while
19:    i ← i + 1;
20: end while
```

For a particular FIFO memory location (present value of i), the first iteration of j (address run1) performs the invert phase, where the content of the FIFO location is inverted. The invert test phase involves reading the content of *lut* into a temporary variable *temp* and then backing it up in *original*. Then, the inverted content of *temp* is written back to *lut*. At this point, the content of *lut* is inversion of content of *original*.

In the next iteration of j (address run2), the restore phase is performed. The content of *lut* is reread into *temp* and compared with the content of *original*. The comparison should result in all 1's pattern. However, deviation from the all 1's pattern at any bit position indicates fault at that particular bit position. Next, the inverted content of *temp* is written back to *lut*. Thus, the content of *lut*, which were inverted after the first iteration get restored after the second.

The third iteration of j performs only a read operation of *lut*, where the content of *lut* is read into *temp* and compared with the contents of *original*. At this stage of the test, all 0's pattern in the result signifies fault free location, while deviation at any bit position from all 0's pattern means fault at that particular bit position. The last read operation ensures the detection of faults, which remained undetected during the earlier two test runs. At the end of the three test runs (iterations of j), the loop index i is incremented by one to mark the start of test for the next location.

B. Fault Coverage of the Proposed Algorithm

The transparent SOA-MATS++ algorithm is intended for test of stuck-at fault, transition fault, and read disturb fault fault tests developed during field operation of FIFO memories. The fault coverage of the algorithm is shown in Fig. 1. In both the figures, the word size of FIFO memory is assumed to be of 4 bits. The text in italics against the arrows indicates the operation performed, while the text in bold font corresponds to the variables used in Algorithm 1.

As shown in Fig. 1, assume the data word present in *lut* be 1010. The test cycles begin with the invert phase (memory address pointer j with 0 value) during which the content of location addressed is read into *temp* and then backed up in the *original*. The data written back to

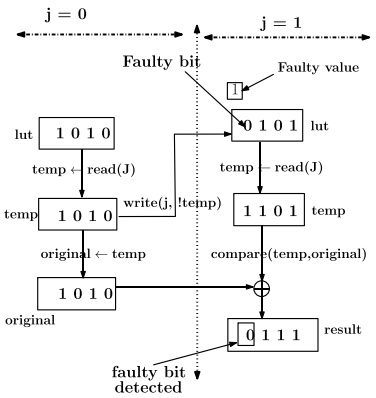

Fig. 1. Fault detection during invert phase and restore phase of the transparent SOA-MATS++ test.

lut is the complement of content of *temp*. Thus, at the end of the cycle, the data present in *temp* and *original* is 1010, while *lut* contains 0101. Assume a stuck-at-1 fault at the most significant bit (MSB) position of the word stored in *lut*. Thus, instead of storing 0101, it actually stores 1101 and as a result, the stuck-at-fault at the MSB gets excited.

During the second iteration of *j*, when *lut* is readdressed, the data read into *temp* is 1101. At this point, the data present in *temp* and *original* are compared (bitwise XORed). An all 1's pattern is expected as result. Any 0 within the pattern would mean a stuck-at fault at that bit position. This situation is shown in Fig. 1, where the XOR of 1010 and 1101 yields a 0 at the MSB position of the *result* indicating a stuck-at-fault at the MSB position. However, for cases where the initial data for a bit position is different from the faulty bit value, the stuck-at-fault cannot be detected for the bit position after the restore phase of the test. It thus requires one more test cycle to excite such faults.

IV. Implementation of the Test on FIFO Buffers of NoC Routers

In this section, we present the technique used for implementing the proposed transparent SOA-MATS++ test on a mesh-type NoC. Data packets are divided into flow control units (*flits*) and are transmitted in pipeline fashion [1]. The flit movement in a mesh-type NoC infrastructure considered for this work is assumed to require buffering only at the input channels of routers. Thus, for a data traffic movement from one core to another, the online test is performed only on the input channel FIFO buffers, which lie along the path. The buffers operate in two modes, the *normal* mode and the *test* mode. The normal mode and test mode of operation of a FIFO buffer are synchronized with two different clocks. The clock used for test purpose (referred as *test_clk* in this brief) is a faster clock compared with the clock required for normal mode (*router clock*).

The FIFO buffers are allowed to be operative in normal mode for sufficient amount of time before initiating their test process. This delay in test initiation provides sufficient time for run-time intermittent faults developed in FIFO buffers to transform into permanent faults. The test process of a targeted FIFO buffer is initiated by a counter, which switches the FIFO buffer from normal mode to test mode. The switching of FIFO buffers from normal mode to test mode occurs after a certain period of time without caring about the present state of the FIFO buffer. It may be argued that at the instant of switching, the buffer may not be full, and as a result not all locations would be tested during the test cycle. However, test initiation after the buffer gets full would cause the following problems. First, wait for the buffer to get full would unnecessarily delay the test initiation process and would allow faults to get accumulated. Second, test of the entire buffer would prolong the test time and would negatively affect the normal mode of operation.

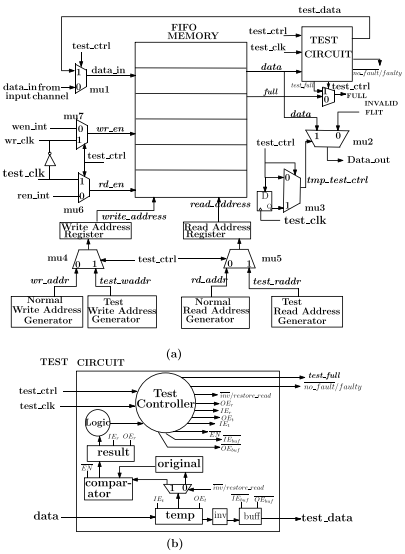

Fig. 2. (a) Hardware implementation of the test process for the FIFO buffers. (b) Implementation of test circuit.

A test burst involves series of test read and write cycles. It requires three read and two write cycles, or in other words three cycles of the faster test clock to perform a transparent SOA-MATS++ test on a single location of a FIFO buffer. It may be argued that during a test burst, not all FIFO buffer locations are tested or a test of a location can get interrupted. These two problems can be avoided by periodically testing the FIFO buffers. Periodic testing of a FIFO buffer allows test of a different set of locations of the FIFO buffer in each test burst. Every time the buffer is switched to test mode, the normal process gets interrupted. The FIFO memory location currently addressed in normal mode, at the instant of switching, becomes the target location for test. Since normal operation is interrupted at different instants in different test bursts, the locations tested in each burst would be different. Thus, repeating the test bursts for a number times on a FIFO buffer would cover the test of each location as the number of locations in a FIFO buffer is few. Moreover, periodic testing prevents accumulation of fault in the buffer.

A. Test Architecture

The FIFO buffer present in each input channel of an NoC router consists of a SRAM-based FIFO memory of certain depth. During normal operation, data flits arrive through a *data_in* line of the buffer and are subsequently stored in different locations of the FIFO memory. On request by the neighboring router, the data flits stored are passed on to the output port through the *data_out* line. Fig. 2(a) shows the FIFO memory with *data_in* and *data_out* line. To perform the transparent SOA-MATS++ test on the FIFO buffer, we added a test circuit, few multiplexers and logic gates to the existing hardware, as shown in Fig. 2(a). The read and write operations on the FIFO buffer are controlled by the read enable and write enable lines, respectively. The multiplexers *mu*6 and *mu*7 select the read and write enable during the normal and test process. During normal operation when the *test_ctrl* is asserted low, the internal write and read enable lines, *wen_int* and *ren_int*, synchronized with the router clock, provide the write and the read enable, respectively. However, during test process, the write enable and read enable are synchronized with the test clock. As mentioned earlier, the read and write operations during test are

performed at alternate edges of a test clock. The read operations are synchronized with the positive edges, while the *write_clk* is obtained by inverting the test clock. In test mode (*test_ctrl* high), the test read and write addresses are generated by test address generators implemented using gray code counters similar to the normal address generation. Muxes *m*4 and *m*5 are used to select between normal addresses and test addresses.

Consider the situation when the FIFO buffer is in normal mode with flits being transferred from the memory to the *data_out* line. After a few normal cycles, the *test_ctrl* is asserted high, switching the buffer to test mode. As long as the buffer is in test mode, no external data is allowed to be written to the buffer, or in other words, the buffer is locked for the test period. As a result, the input data line for the FIFO memory is switched from the external *data_in* line to *test_data* line available from the test circuit. At the switching instant, the flit which was in the process of being transferred to the *data_out* line is simultaneously read into the *Test Circuit*. However, a one clock cycle delay is created for the flit to move to the *data_out* line. This delay ensures that the flit is not lost during the switching instant and is properly received by the router, which requests for it. The single cycle delay in the path of the traveling flit is created by the D-type flip-flop and the multiplexer *m*3, as shown in Fig. 2(a). The flit, which is read in the test circuit, is stored in a temporary register *temp* and the test process begins with this flit.

To avoid packet loss during testing, the *FULL* signal of the FIFO is asserted *high* so that neighboring routers can be prevented from transferring packets to the corresponding router. However, applying such technique increases the network latency as reflected in the results shown in Section V.

V. Experimental Results

A prototype implementation of the proposed test circuit has been integrated into the router-channel interface and online transparent SOA-MATS++ test is performed with synthetic self-similar data traffic. The router design considered in this brief has been taken from [16].

A. Area Overhead Estimation of the Test Hardware

The proposed hardware for the test circuit has been described in Verilog HDL and synthesized using *Synopsys Design Vision* supporting 90-nm CMOS technology. The total area estimate of different modules in the test circuit after synthesis has been estimated to be $\sim$1720 μm^2, while the area of the FIFO buffer (having depth = 6) has been estimated as 4074 μm^2. Thus, a significant amount of 42% area overhead results due to inclusion of the test circuit. However, the overhead when estimated with respect to the router area reduces to 8% and still further to 2% when estimated with respect to the area of the entire NoC.

B. Throughput Estimation

For evaluating the performance of an NoC-based network, a system C-based cycle-accurate NoC simulator [16] has been utilized. Synthetic self-similar traffic has been used during simulation, guided by the communication requirement of cores in the application. The simulator has been utilized to compute the throughput of the network with and without the test circuit. Each simulation has been run for 200 000 clock cycles. In this brief, the definition of throughput and network latency considered is same as in [16]. For a mesh-type NoC of size 4 $\times$ 8, the throughput for a FIFO buffer (depth = 6) without including the test circuit has been estimated to be 0.281. Then, we tried to investigate the effect on overall throughput by including the test circuit within the routers and performing tests at periodic intervals. When the periodicity of test is 20 000 ms, the throughput has been estimated to be 0.280, while it drops by 5.3% and the network latency increases by 4.8% in case the FIFO memory is tested after every 5000-ms period. It may be concluded from the results that if online transparent March tests are frequently performed on FIFO memory, the overall throughput of the NoC decreases, while the network latency increases due to interruption of packet transfer. However, delaying the periodicity of test results in throughput value comparable with result obtained when no tests were performed.

VI. Proposal for Test of Routing Logic

The other part of the router, besides the buffers, vulnerable to run-time permanent faults is the routing logic. In this section, we propose an online test proposal for the routing logic that utilizes the data packets for testing and thus overcomes the need for test access mechanism. Since the router design considered for this brief is taken from [16], both flit size and link width equal to 32-bit as used in [16]. For the header flit, after allocation of address bits (source and destination) and bits for virtual channel selection, some fields in the header flit remain unused. Our proposal is to utilize these unused fields for test pattern encoding. An automatic test pattern generation tool generates deterministic offline test patterns for the routing logic. Once the set of test patterns are available, each pattern can be placed in the unused fields of the header flit. If the size of the test pattern does not fit the size of the available field size in a single header, the test pattern is adjusted in two header flits. In such a situation, it requires two test cycles before the test pattern reaches the routing logic. The test patterns are carried to the routing logic by the NoC infrastructure during normal operation and are applied for testing during the test mode. The test of the routing logic is simultaneously performed with test of the FIFO buffers during the test mode when the normal operation of the router remains suspended.

To validate our proposal, the router has been synthesized using *Design Vision* supporting 90-nm technology and then *Tetramax* has been used to generate deterministic test patterns for the synthesized netlist. A total of 39 test patterns have been generated covering 242 faults for 100% fault coverage. Each pattern is of size 41 bits, which required two test cycles for transporting the test patterns. Thus, in total, 78 test cycles have been required for test of routing logic. We have also experimented with an alternate proposal of using pseudorandom patterns for test. Instead of using deterministic test patterns, we utilize the pseudorandom synthetic data traffic used during normal operation. Similar to the earlier proposal, the pseudorandom bits in each header flit have been treated as test patterns and have been applied to the routing logic. Fault simulation performed on the routing logic using the pseudorandom patterns utilizing a standard fault simulator provides 60% fault coverage.

VII. Conclusion

In this brief, we have proposed a transparent SOA-MATS++ test generation algorithm that can detect run-time permanent faults developed in SRAM-based FIFO memories. The proposed transparent test is utilized to perform online and periodic test of FIFO memory present within the routers of the NoC. Periodic testing of buffers prevents accumulation of faults and also allows test of each location of the buffer. Simulation results show that periodic testing of FIFO buffers do not have much effect on the overall throughput of the NoC except when buffers are tested too frequently. We have also proposed an online test technique for the routing logic that is performed simultaneously with the test of buffers and involves utilization of the unused fields of the header flits of the incoming data packets

for test pattern encoding. As future work, we would like to modify the proposed FIFO testing technique that will allow incoming data packets to the router under test without interrupting the test.

REFERENCES

[1] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Proc. 38th Annu. Design Autom. Conf.*, 2001, pp. 684–689.

[2] A. Bondavalli, S. Chiaradonna, F. Di Giandomenico, and F. Grandoni, "Threshold-based mechanisms to discriminate transient from intermittent faults," *IEEE Trans. Comput.*, vol. 49, no. 3, pp. 230–245, Mar. 2000.

[3] M. Radetzki, C. Feng, X. Zhao, and A. Jantsch, "Methods for fault tolerance in networks-on-chip," *ACM Comput. Surv.*, vol. 46, no. 1, pp. 1–38, Jul. 2013, Art. ID 8.

[4] S. Ghosh and K. Roy, "Parameter variation tolerance and error resiliency: New design paradigm for the nanoscale era," *Proc. IEEE*, vol. 98, no. 10, pp. 1718–1751, Oct. 2010.

[5] S. Borri, M. Hage-Hassan, L. Dilillo, P. Girard, S. Pravossoudovitch, and A. Virazel, "Analysis of dynamic faults in embedded-SRAMs: Implications for memory test," *J. Electron. Test.*, vol. 21, no. 2, pp. 169–179, Apr. 2005.

[6] M. Bushnell and V. Agrawal, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits* (Frontiers in Electronic Testing). New York, NY, USA: Springer-Verlag, 2000.

[7] D. Xiang and Y. Zhang, "Cost-effective power-aware core testing in NoCs based on a new unicast-based multicast scheme," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 1, pp. 135–147, Jan. 2011.

[8] K. Petersen and J. Oberg, "Toward a scalable test methodology for 2D-mesh network-on-chips," in *Proc. Design, Autom., Test Eur. Conf. Exhibit.*, Apr. 2007, pp. 1–6.

[9] D. Xiang, "A cost-effective scheme for network-on-chip router and interconnect testing," in *Proc. 22nd Asian Test Symp. (ATS)*, Nov. 2013, pp. 207–212.

[10] M. Ebrahimi, M. Daneshtalab, J. Plosila, and H. Tenhunen, "Minimal-path fault-tolerant approach using connection-retaining structure in networks-on-chip," in *Proc. 7th IEEE/ACM Int. Symp. Netw. Chip (NoCS)*, Apr. 2013, pp. 1–8.

[11] C. Grecu, P. Pande, B. Wang, A. Ivanov, and R. Saleh, "Methodologies and algorithms for testing switch-based NoC interconnects," in *Proc. 20th IEEE Int. Symp. Defect Fault Tolerance VLSI Syst.*, Oct. 2005, pp. 238–246.

[12] M. R. Kakoee, V. Bertacco, and L. Benini, "A distributed and topology-agnostic approach for on-line NoC testing," in *Proc. 5th ACM/IEEE Int. Symp. Netw. Chip*, May 2011, pp. 113–120.

[13] S. Barbagallo *et al.*, "A parametric design of a built-in self-test FIFO embedded memory," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Syst.*, Nov. 1996, pp. 221–229.

[14] A. J. van de Goor and Y. Zorian, "Functional tests for arbitration SRAM-type FIFOs," in *Proc. 1st Asian Test Symp. (ATS)*, Nov. 1992, pp. 96–101.

[15] M. Nicolaidis, "Theory of transparent BIST for RAMs," *IEEE Trans. Comput.*, vol. 45, no. 10, pp. 1141–1156, Oct. 1996.

[16] S. Kundu, J. Soumya, and S. Chattopadhyay, "Design and evaluation of mesh-of-tree based network-on-chip using virtual channel router," *Microprocess. Microsyst.*, vol. 36, no. 6, pp. 471–488, Aug. 2012.