Methodologies and Algorithms for Testing Switch-Based NoC Interconnects

Cristian Grecu, Partha Pande, Baosheng Wang, André Ivanov, Res Saleh
SoC Research Lab
Department of Electrical and Computer Engineering
University of British Columbia
2356 Main Mall Vancouver, BC, V6T 1Z4 Canada

Abstract

In this paper, we present two novel methodologies for testing the interconnect fabrics of network-on-chip (NoC) based chips. Both use the concept of recursive testing, with different degrees of parallelism in each case. Our test methodologies cover the logic switching blocks and the FIFO buffers that are the basic components of NoC fabrics. The paper concludes with test time evaluations for different NoC topologies and sizes.

1. Introduction and motivation

Recent advances in semiconductor technology raise the challenge of managing the difficulties of designing complex chips containing billions of transistors. In this context, the reusability concept is an essential tool that helps when coping with this issue, where large Systems on Chip (SoC) are built using pre-existing Intellectual Property (IP) cores. Networks on Chip (NoC) [1] [2] interconnect infrastructures are emerging as the new paradigm to address the on-chip data communication architecture of large scale SoCs. In general, two trends can be identified with respect to the architectures of the different NoC implementations: (i) regular interconnection structures derived from parallel computing, and (ii) irregular, custom-built NoC fabrics. Kumar [3] has proposed a mesh-based interconnect architecture named CLICHÉ. Dally et al. [4] suggested a torus-based architecture. Both of these architectures consist of an m x n mesh of switches interconnecting computational resources (IPs) placed along with the switches. Guerrier and Greiner [5] proposed the use of a tree-based interconnect (SPIN) and addressed system level design issues. In [6] and [7], the authors describe an interconnect architecture based on the Butterfly Fat Tree (BFT) topology for a networked SoC, as well as the associated design of the required switches and addressing mechanisms. Benini et al. [8] have introduced application specific irregular architectures to build networked SoCs. The common characteristics of all these interconnect fabrics is that the functional IP blocks communicate with each other through a highly structured switched medium, consisting of intelligent switches and inter-switch links.

A necessary requirement for any new design methodology to be widely adopted is to be complemented by efficient test mechanisms. So far, the NoC test methodologies are mainly concerned with the testing of the functional IP cores, using the communication infrastructure as a Test Access Mechanism (TAM) [9]. Our work complements this approach by developing the test strategy for the interconnect infrastructure itself. The test strategy of NoC-based interconnects infrastructures needs to address two problems: (i) testing of the switch blocks; (ii) the testing of the inter-switch interconnect segments. Here we specifically propose and evaluate the testing methodology suitable for the NoC switch blocks.

The controllability/observability of the NoC interconnects is relatively reduced, due to the fact that they are deeply embedded and spread across the chip. Pin-count limitations restrict the use of I/O pins dedicated for the test of the different components of the data-transport medium; therefore, we propose that the NoC infrastructure be progressively used for testing its own components in a recursive manner, i.e., that the good, already tested NoC components be used to transport test patterns to the untested elements. This test strategy minimizes the use of additional mechanisms for transporting data to the NoC elements under test, while allowing the reduction of test time through the use of parallel test paths and test data multicast.

2. Related work

Since the inception of SoC designs, the research community has targeted principally the testing of the IP cores, giving little or no emphasis on the testing of the communication infrastructure. Among the different test access mechanisms, TestRail [10] can be mentioned as one of the first to address core-based test of SoCs. After the emergence of the NoC design paradigm, different research groups proposed the reuse of the communication infrastructure as a TAM [9] [11].

Proceedings of the 2005 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'05)
0-7695-2464-8/05 $20.00 © 2005 **IEEE**

IEEE COMPUTER SOCIETY
Authorized licensed use limited to: Washington State University. Downloaded on June 17,2010 at 16:36:53 UTC from IEEE Xplore. Restrictions apply.

In [11] the authors assumed the NoC fabric as fault-free and subsequently used it to transport test data to the functional blocks; however, for large systems, this is an unrealistic assumption, considering the complexity of the design and communication protocols. In [12], the authors proposed a dedicated TAM based on an on-chip network, where network oriented mechanisms are used to deliver test data to the functional cores of the SoC.

In this work, we are principally focused on the development of test strategies for NoC communication backbones.

3. Switch architecture

Typically, in NoC-based designs, the data exchange between functional IP cores takes place in the form of packets. There are different schemes that have been proposed for on-chip communication, namely *circuit switching*, *virtual cut-through* and *wormhole switching* [17]. The first is more suitable for relatively small-size networks, while the latter two are appropriate for large-scale SoCs, where data have to traverse longer distances and have to be buffered at the intermediate switches on the path from source to destination.

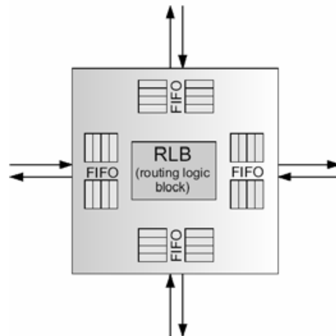

Figure 1: NoC switch architecture

The switch blocks consist of the FIFO buffers and the routing logic blocks (RLB) as shown in Fig. 1. Generally, wormhole routing is adopted as the data transport methodology. In wormhole switching, the packets are divided into fixed length flow control units (*flits*) and the input and output buffers are expected to store only a few flits. As a result, the buffer space requirement in the switches can be small compared to that generally required for other schemes. The first flit, i.e., *header flit,* of a packet contains routing information. Header flit decoding enables the switches to establish the path and subsequent flits simply follow this path in a pipelined fashion. In NoC architectures, the inter-switch wire segments together with the switch blocks constitute a highly-pipelined communication medium. The switches generally consist of multiple pipeline stages [8] [13]. Though the number of intra-switch pipeline stages may vary with design style and features incorporated within the switch blocks, through careful circuit level design and analysis, the delay of each intra-switch stage can be made to be less than the target clock period in a particular technology node [13].

4. Testing of NoC switches

The switch blocks consist of the FIFO buffers and the routing logic. Consequently, the testing of the switch blocks can be sub-divided into two problems: testing of the FIFO buffers and the testing of the routing circuitry. Generally, routing logic consists of a few hundred logic gates and traditional testing methods like scan or BIST can be employed here. However, testing of the FIFO buffers poses a unique challenge as numerous relatively small buffers are distributed all over the chip.

4.1 Testing of Routing Logic Blocks (RLB)

Once the FIFO buffers are tested and known to be operational, they can be reused to transport test data to the combinational blocks of the switches. Our test methodology assumes that the test data patterns used to test the RLBs are injected from an external ATE through one injection port that is connected directly to one of the NoC switches. Moreover, we assume that scan insertion was performed as a DFT strategy for the RLBs. The expected values of the scan-chain outputs (SCO)

IEEE COMPUTER SOCIETY
Authorized licensed use limited to: Washington State University. Downloaded on June 17,2010 at 16:36:53 UTC from IEEE Xplore. Restrictions apply.

and of the primary outputs (PO) of the RLBs are also transported to each switch, such that the expected values of SCOs and POs can be compared locally, similar to [12], thus eliminating the need for an external test sink [14].

In the following two subsections, we describe two different strategies we propose for recursively transporting test data to the RLBs of the NoC switches. The first strategy uses a simple, unicast mechanism to test the RLBs sequentially; the second one involves a combination of sequential and parallel mechanisms, together with a multicasting scheme.

Before presenting the two strategies, we introduce the following notations:

T: total test time for a NoC based infrastructure, consisting of N switches (in clock cycles);
T_r: total test time for testing one stand-alone RLB (= to the product of the number of test patterns and test clock cycles/pattern);
N_p: Number of pipeline stages per switch.

4.1.2 Strategy I: sequential testing – unicast

Under this strategy, the RLBs of the switches are tested in a sequential-recursive manner, one block at a time, using the upstream tested switches to transport test data to the RLB under test. Due to the fact that there is one test data source (the *Test plug* in Fig. 3) and one RLB under test at a time, this test strategy will subsequently be referred to as *unicast test.*

The switches have two modes of operation: the test mode (T), in which test data is scanned-in/captured/scanned-out, and a normal mode (N) in which test patterns are transported toward the next RLB to be tested. The algorithm of the test sequence is described in Fig. 2.

Algorithm 1 – unicast test

```
Start
While Test_result = OK do
        For i =1 to N
        Set Mode(switch[i-1]) = Normal;
              Set Mode(switch[i]) = Test;
              Execute Test(switch[i]);
              Return Test_result;
End
```

Figure 2: Pseudocode for unicast test sequence

For test purposes, the switches are indexed from 1 to N relative to their reachability, i.e., switch[i+1] is reachable through switch[i] if there is a direct connection between switch[i] and switch[i+1].

For clarification, let us consider two distinct cases corresponding to two different NoC topologies: (i) a mesh-connected and (ii) a Butterfly-fat-tree (BFT)-connected network. Fig. 3 shows a snapshot of the state of the NoC for two successive unicast test steps according to *Algorithm 1.* In the first step, the switch under test is marked by T; in the next step, the mode of operation of this switch is changed to *normal* node, marked by N, and the next switch is tested.

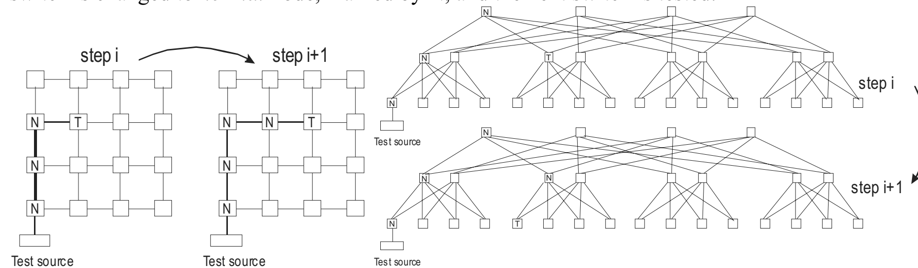

Figure 3:
Unicast test strategy for a 4x4 MESH

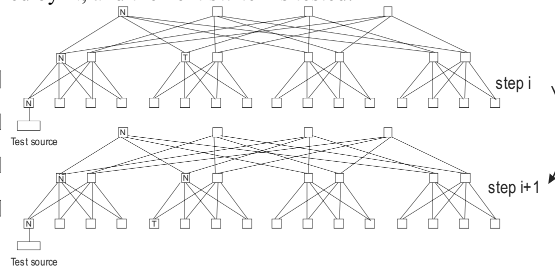

Figure 4:
Unicast test strategy for a –level BFT

0-7695-2464-8/05 $20.00 © 2005 IEEE
IEEE COMPUTER SOCIETY
Authorized licensed use limited to: Washington State University. Downloaded on June 17,2010 at 16:36:53 UTC from IEEE Xplore. Restrictions apply.

To estimate the amount of time required to test the architecture in Fig. 3, observe that the time required to actually test one RLB is equal to the sum of the time required to transport the test data at the location of the RLB and the test time of a stand-alone RLB (T_r); the time needed for test data to traverse a switch and an inter-switch link is equal to (N_p + 1) test clock cycles. With these observations, we can derive the expression for the time required to test the RLBs on the first column as:

$$T_1 = \sum_{i=1}^{m}[(i-1)(N_p+1)+T_r] = mT_r + \frac{m(m-3)}{2}\left(N_p+1\right) \quad (1)$$

Hence, the time needed for testing the switches on column j is:

$$T_j = (j-1)(N_p+1)+T_1 \quad (2)$$

And finally, the total unicast test time for the m x $m = N$ switches mesh is:

$$T^u_{MESH} = \sum_{j=1}^{m} T_j = m^2T_r + \frac{m(m+1)(m-3)}{2}\left(N_p+1\right) \quad (3)$$

The unicast approach for delivering the test patterns yields a test time that depends linearly on the number of switches present in the NoC infrastructure (since m^2 equals the number of switches). For small or medium sized networks, this linear dependence is acceptable and has the advantage of simple implementation, since the only special measure that has to be taken is to provide the two modes of operation (normal/test) for the switches, and the appropriate control signals.

The unicast test strategy for the BFT architecture is shown in Fig. 4. To derive the test time corresponding to the fat-tree architecture in Fig. 4, we start by observing that such a topology is fully characterized by the number of levels and the number of child/parent ports. Moreover, the unicast test time is equal to the sum of distances from the test plug node to all the other vertices (corresponding to the time required to transport the test patterns from the test plug to their destinations), plus a term generated by the effective test procedure per each switch. With these considerations, the total unicast test time for a BFT having l levels can be expressed recursively as:

$$T^u_{BFT,l} = T^u_{BFT,l-1} + 2(l-1)(N_p+1) + 4(N_p+1)\sum_{k=0}^{l-1}2^k(k+4) + T_r\left(2^{2l-3}+2^{2l-2}-2^{l-2}\right) \quad (4)$$

The second term of the expression above accounts for the two parent ports of the BFT switches, and the third term accounts for the four children ports. The last term accounts for the switches that are added when the number of BFT levels increases from *l-1* to *l*.

4.1.3 Strategy II: multicast testing

To improve the test time of the NoC infrastructure switches, we now consider the use of a multicasting scheme to transport test patterns to the RLBs under test in a parallel fashion. In our *multicast testing* scheme, there is one test data source and multiple RLBs under test concurrently. To incorporate multicasting, the addressing mechanism has to be modified, as incoming flits will go to more than one port at a time.

The header of a multidestination message must carry the destination node addresses [17]. To route a multi-destination message, a switch must be equipped with a method for determining the output ports to which a multicast message must be simultaneously forwarded. The multi-destination worm header encodes information that allows the switch to determine destination ports.

One possible way to multicast is simply to unicast multiple times, but this implies a very high latency. The all-destination encoding is another simple scheme in which all destination addresses are carried by the header. This encoding scheme has two advantages. First, the same routing hardware used for unicast messages can be used for multi-destination messages. Second, the message header can be processed on the fly as address flits arrive. The main problem with this scheme is that as the number of switch blocks in the system increases, the header length increases accordingly and thereby results in significant overhead.

Authorized licensed use limited to: Washington State University. Downloaded on June 17,2010 at 16:36:53 UTC from IEEE Xplore. Restrictions apply.

A form of header encoding that accomplishes multicast to arbitrary multicast destination sets in a single communication phase and also limits the size of the header is known as *bit-string encoding* [19]. A multi-destination worm with a bit-string encoded header carries an encoding of the destinations. The encoding consists of N bits where N is the number of switch blocks, with a '1' bit in the ith position indicating that switch i is a multicast destination. To decode a bit-string encoded header, a switch must possess knowledge of the switches reachable through each of its output ports [18].

The reachability information can be encoded using a similar N bit string for each output port with 1's denoting switches reachable via this output port. When a multi-destination worm with a bit-string encoded header arrives at a switch, the switch compares the bit-string in the header with the reachability information associated with each of its output ports.

The algorithm of the multicast test sequence is described in Fig. 5.

```
Algorithm 2 – multicast test
Start
While Test_result = OK do
Set Mode(switch[0]) = Test;
Execute Test(switch[0]);
        Get set ℜ(0) of switches reachable through switch[0];
        Set Mode(switch[0]) = Multicast;
        Set Mode(ℜ(0)) = Test;
        Execute Test(ℜ(0));
        Get set ℜ(1) of switches reachable through ℜ(0);
        Set Mode(ℜ(0)) = Multicast;
        Set Mode(ℜ(1)) = Test;
        Execute Test(ℜ(1));
Return Test_result;
End
```

Figure 5: Pseudocode for multicast test sequence

where $\Re(i)$ denotes the set of switches reachable by the ith switch.

Fig. 6 shows two consecutive steps of the multicast test sequence for a 16 switches MESH interconnect.

Similarly to the case of unicast test, we now determine the test time required by a mesh-connected NoC infrastructure. Considering the test plug placed on one corner of the structure of Fig. 6, the number of multicast steps that have to be carried out in order to execute *Algorithm 2* is *2m-1*.

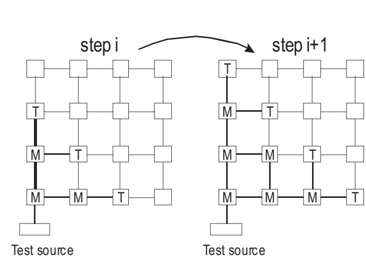

Figure 6:
Multicast test strategy for a 4x4 MESH

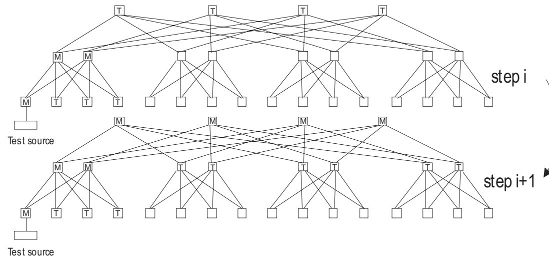

Figure 7:
Multicast test strategy for a 4-level BFT

Correspondingly, the amount of time required to execute step i can be expressed as:

$$T_i = T_r + (i-1)(N_p + 1) \tag{5}$$

IEEE COMPUTER SOCIETY
Authorized licensed use limited to: Washington State University. Downloaded on June 17,2010 at 16:36:53 UTC from IEEE Xplore. Restrictions apply.

Consequently, the total test time can be calculated as:

$$T_{MESH}^{m} = \sum_{1}^{2m-1} T_i = (2m-1)T_r + (2m-1)(m-2)(N_p+1) \tag{6}$$

Note that the test time is significantly reduced when compared to the case of unicast test.

Figure 7 presents two consecutive multicast test steps in the case of the BFT architecture according to *Algorithm 2.* Similarly to the MESH architecture, the total test time for BFT in the case of a multicast test will be given as

$$T_{BFT}^{m} = (\log_2 N - 1)T_r + (\log_2 N - 2)(N_p+1) \tag{7}$$

From Eq. (7), the test time required by the multicast test strategy is significantly shorter than that for a unicast test, since it depends logarithmically on the number of switches under test.

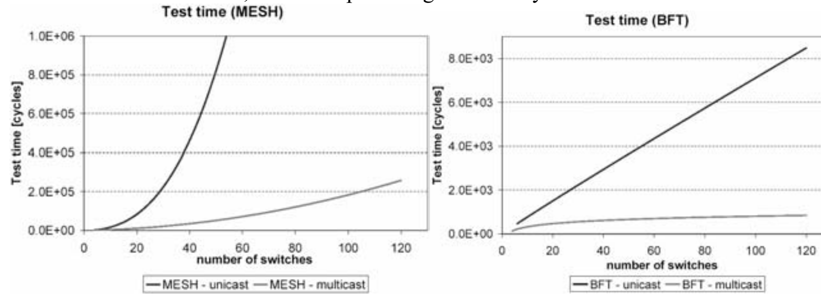

Figure 8: Test time comparison

Fig. 8 summarizes the behaviour of *Algorithm 1* (unicast) and *Algorithm 2* (multicast) with respect to the size of the NoC under test for the two different architectures considered here. In a typical NoC design, the number of functional blocks is quite high and consequently a large number of switch blocks are required. It is evident from Fig. 8 that in such a case multicast testing significantly outperforms the unicast methodology.

4.2 NoC-FIFO test strategy

A critical step in the test methodology of the NoC infrastructure is the test of the FIFO memories that buffer the data at the inputs/outputs of the switches. It has been shown that they account for a considerable amount of silicon area [6] relative to the total area of the NoC data transport medium, and, consequently, potential defects affecting these FIFOs can significantly compromise the yield of the NoC based chips.

In the NoC scenario, the classical BIST approach for testing the FIFO blocks is not suitable, as one dedicated BIST mechanism per FIFO will give rise to an unacceptably large silicon area overhead. Consequently, a distributed BIST methodology as the one depicted in Fig. 9 is more appropriate.

In a distributed BIST scheme, the read/write mechanisms, the control circuitry and the test data source are shared among the multiple FIFO blocks, whereas the local response analyzers (LRA) are distributed, one for each FIFO [15].

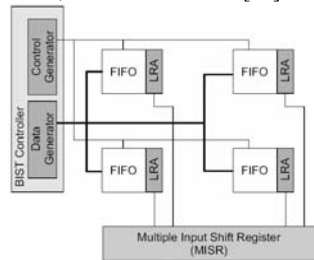

Figure 9:
Distributed mechanism for FIFO test

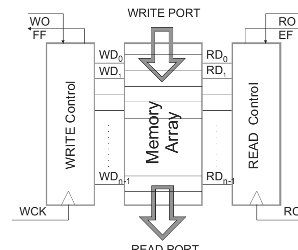

Figure 10:
Dual port (wo-ro) FIFO structure

Proceedings of the 2005 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'05)
0-7695-2464-8/05 $20.00 © 2005 IEEE
IEEE COMPUTER SOCIETY
Authorized licensed use limited to: Washington State University. Downloaded on June 17,2010 at 16:36:53 UTC from IEEE Xplore. Restrictions apply.

From a test point of view, the NoC-specific FIFOs fall under the category of restricted two-port memories, in the sense that, due to the unidirectional nature of the NoC communication links, they have one write-only port and one read-only port, and are referred to as (wo-ro)2P memories. Under the restrictions above, the FIFO function can be divided in three ways: the memory-cells array, the addressing mechanism, and the FIFO-specific functionality [17].

In the following, we describe the fault models associated with each of these components.

Memory array faults

1. Stuck-at fault SAF: for any operation, the memory cell remains at 0 or 1;
2. Transition fault TF: after a transition to 1, the cell remains at 0 and vice versa.
3. DRF (data retention fault): after a time *Del* the memory cell loses its data.
4. Bridging fault BF: a short between the bitlines b_i and b_j ($i \neq j$), that can eventually yield an AND (BFand) or an OR (BFor) behaviour. This fault model accounts also for dual port coupling faults (a write operation executed on cell *i* determines an erroneous read operation executed on adjacent cell *j* and vice-versa).

Addressing faults

Reading/writing data from/into a specific FIFO location is done under the control of the corresponding *RD/WD* (read data/write data) address lines. A SOF (stuck-open-fault) on RD_i/WD_i line behaves like a SAF on RD_i/WD_i. A SA0 (stuck-at-0) fault on RD_i/WD_i line prevents cell *i* from being read/written; a SA1 fault on RD_i/WD_i line has the effect that cell *i* is read every cycle and cause the read pointer to incorrectly point to cell (*i+1*), or that cell *i* is written every cycle regardless of the status of the write pointer. Hence, the addressing faults of interest are:

5. SA0 fault on RD_i/WD_i
6. SA1 fault on RD_i/WD_i

FIFO functionality-specific faults:

These fault models relates to the faults of the control logic block in Fig. 10.

- Faulty reset (RS), write (WO), and read (RO) operations

9. FF, EF lines not initialized after an RS operation.
10. Full flag not set (FF≠1) after *n* successive WOs.
11. Empty flag not set (EF≠1) after *n* successive ROs.

- Faults on FF and EF lines – these are detected by fault models 9 through 11.

Functional test for FIFO buffers

The test for the faults presented above was developed based on [16]. Assuming the FIFOs being B bits wide and having *n* locations, the test uses B-bits background pattern. In order to detect the bridging faults (BFs), four specific background patterns are used: 0101…, 1010…, 0000…, and 1111…, denoted by G_1, $\overline{G}_1$, G_2, and $\overline{G}_2$, respectively. Specifically, to test the dual port coupling faults, the following sequence is used:

$$w\left\{\Uparrow_1^{n-1}\left(wr\right)\right\}r$$

for each of the four background patterns above. The first *write* operation initializes sets the read/write pointers to FIFO cells 0 and 1 respectively; the next (*n-1*) simultaneous *read/write* operations sensitize the coupling faults between adjacent cells, and the last *read* operation empties the FIFO and prepares it for the next background pattern. As such, the number of cycles required to test the dual port coupling faults is $4\left(n+1\right)$.

Hence, the total test time required for testing the FIFO buffers amounts to:

$$T_{FIFO} = \left(2n + 3Del + 4\right) + 4\left(n+1\right) = 6n + 8 + 3Del \qquad (8)$$

where the first term amounts for the test of single port faults [17] and the second for the test of dual port faults.

T_{FIFO} is not affected by the system size; rather, it appears as a constant overhead irrespective of the specific test strategies adopted for testing the combinational RLBs.

IEEE COMPUTER SOCIETY
Authorized licensed use limited to: Washington State University. Downloaded on June 17,2010 at 16:36:53 UTC from IEEE Xplore. Restrictions apply.

5. Results and comparison

In order to evaluate the two test strategies proposed in this work, we applied them to regular NoC architectures of different sizes, and also varied the size of FIFO buffers within limits suggested by [17][20].

In our experiments, we used gate-level netlists of the NoC switches, and performed full-scan insertion with Synopsys™ DFT Compiler; the test patterns were generated with Synopsys Tetramax ATPG software. The number of test patterns was 146 in the case of MESH RLBs and 153 in the case of BFT RLBs.

Table 1 summarizes the results of our experiments. We can observe that there are two components of the test time: one related to the size of the chip, and the second related to the testing of the data retention faults for the FIFO buffers. The first term can be greatly reduced by employing the multicast test strategy, as seen in the last two columns of Table 1; however, the second one (the 3*Del* term) depends only on the nature of the fault tested and process parameters.

Table 1 **– Test time results**

Architecture	System Size (# of switches)	FIFO Size (depth *x* width)	Test Time (#test clock cycles + DRF test time)	
			Unicast Test	Multicast Test
MESH	16	4x16	2512+3Del	1194+3Del
		16x16	2584+3Del	1266+3Del
		64x16	2872+3Del	1554+3Del
	64	4x16	9632+3Del	3002+3Del
		16x16	9704+3Del	3074+3Del
		64x16	9992+3Del	3362+3Del
	256	4x16	52576+3Del	8154+3Del
		16x16	52648+3Del	8206+3Del
		64x16	52936+3Del	8514+3Del
BFT	16	4x16	652+3Del	498+3Del
		16x16	724+3Del	570+3Del
		64x16	1012+3Del	858+3Del
	64	4x16	5944+3Del	814+3Del
		16x16	6016+3Del	886+3Del
		64x16	6304+3Del	1174+3Del
	256	4x16	30112+3Del	1130+3Del
		16x16	30184+3Del	1202+3Del
		64x16	30472+3Del	1490+3Del

6. Conclusions and future work

In this paper we proposed two methodologies for testing the switch blocks of NoC interconnects. We paid special attention to the testing procedure of the FIFO components of these blocks. The first method is based on a single source – single destination test pattern transport strategy, while the second one exploits the inherent parallelism of NoCs and uses a single source – multiple destination test pattern transport strategy. These two methodologies have been evaluated with respect to test time for different system sizes. Our results indicate that the multicast-based test strategy improves the test time significantly, specifically for large NoC

IEEE COMPUTER SOCIETY
Authorized licensed use limited to: Washington State University. Downloaded on June 17,2010 at 16:36:53 UTC from IEEE Xplore. Restrictions apply.

sizes. We are currently working on extending these test strategies to irregular NoCs infrastructures, characterized by irregular topologies and/or switch blocks having different number of ports and channel widths.

7. References

[1] L. Benini, G. de Micheli, "Networks on chips: a new SoC paradigm" Computer, Volume: 35, Issue: 1, Jan. 2002, pp: 70 – 78.
[2] P. Magarshack, P.G. Paulin, "System-on-Chip Beyond the Nanometer Wall", Proceedings of DAC 2003, June 2-6, 2003, Anaheim, USA.
[3] S. Kumar et al, "A Network on Chip Architecture and Design Methodology," Proceedings of ISVLSI, pp. 117-124, 2002.
[4] W. J. Dally, B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks", *Proceedings of DAC 2001*, pp: 683-689, Las Vegas, Nevada, USA, June 18-22, 2001.
[5] P. Guerrier, A. Greiner, "A generic architecture for on-chip packet-switched interconnections", Proceedings of DATE 2000, pp: 250 –256.
[6] P. P. Pande, C. Grecu, A. Ivanov, R. Saleh, "Design of a Switch for Network on Chip Applications", Proceedings of ISCAS, Vol. V pp: 217-220 Bangkok, May 2003.
[7] C. Grecu, P. P. Pande, A. Ivanov, R. Saleh "Structured Interconnect Architecture: A Solution for the Non-Scalability of Bus-Based SoCs", Great Lakes Symposium on VLSI 2004, pp: 192-195, Boston, USA, April 26-28.
[8] L. Benini, D. Bertozzi, "Xpipes: A Network-on-chip architecture for gigascale systems-on-chip", IEEE Circuits and Systems Magazine, Volume 4, Issue 2, 2004, pp. 18-31.
[9] E. Cota et al., "Power aware NoC Reuse on the Testing of Core-Based Systems", Proceedings of ITC 2003, pp. 612-621.
[10] E. J. Marinissen et al., "A Structured and Scalable Mechanism for Test Access to Embedded Reusable Cores", *Proceedings of ITC 1998,* pp. 284-293.
[11] B. Vermeulen, J. Dielissen, K. Goossens, C. Ciordas, "Bringing communications networks on a chip: test and verification implications", IEEE Communications Magazine, Volume 41, Issue 9, sept. 2003, pp. 74-81.
[12] M. Nahvi, A. Ivanov, "Indirect Test Architecture for SoC Testing", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume 23, Issue 7, July 2004, pp. 1128-1142.
[13] C. Grecu, P. Pande, A. Ivanov, R. Saleh, "Timing Analysis of Network-on-chip Architectures for MP-SoC Platforms", to appear in Microelectronics Journal, Elsevier.
[14] Y. Zorian, E. J. Marinissen, S. Dey, "Testing Embedded-core-based System Chips", IEEE Computer, Volume 32, Issue 6, June 1999, pp. 52-60.
[15] B. Wang, Y. Wu, A. Ivanov, "Designs for Reducing Test Time of Distributed Small Embedded SRAMs", IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'04), Oct. 2004, pp. 120-128.
[16] A.J. Van de Goor, I. Schanstra, Y. Zorian, "Functional test for shifting-type FIFOs", Proceedings of ED&TC 1995, March 1995, pp. 133-138.
[17] J. Duato, S. Yalamanchili, L. Ni, Interconnection *Networks – An Engineering Approach*, Morgan Kaufmann, 2002.
[18] Partha Pratim Pande,Cristian Grecu, André Ivanov, Res Saleh, "Switch-Based Interconnect Architecture for Future Systems on Chip", Proceedings of SPIE, VLSI Circuits and Systems, Vol. 5117, pp. 228-237, 2003, Maspalomas, Spain.
[19] C. B. Stunkel, R. Sivaram, D. K. Panda, "Implementing Multidestination Worms in Switch-Based Parallel Systems: Architectural Alternatives and their Impact." Proceedings of the 24th ACM International Symposium on Computer Architecture (ISCA-24), June 1997.
[20] G. Varatkar, R. Marculescu, "On-chip Traffic Modeling and Synthesis for MPEG-2 Video Applications", IEEE Transactions on VLSI, Vol.12, No.1, Jan. 2004.

IEEE COMPUTER SOCIETY
Authorized licensed use limited to: Washington State University. Downloaded on June 17,2010 at 16:36:53 UTC from IEEE Xplore. Restrictions apply.