Toward a Scalable Test Methodology for 2D-mesh Network-on-Chips

Kim Petersén, Johnny Öberg
Dept. of Electronic Computer and Software Systems, Royal Institute of Technology (KTH)
KTH/ICT/ECS, Electrum 229, SE-164 40 Kista, Sweden
+46 (0) 70 859 41 84, +46 (0) 8 790 41 27
kim.petersen@hdc.se, johnny@imit.kth.se

Abstract[1]

This paper presents a BIST strategy for testing the NoC interconnect network, and investigates if the strategy is a suitable approach for the task. All switches and links in the NoC are tested with BIST, running at full clock-speed, and in a functional-like mode. The BIST is carried out as a go/no-go BIST operation at start up, or on command. It is shown that the proposed methodology can be applied for different implementations of deflecting switches, and that the test time is limited to a few thousand-clock cycles with fault coverage close to 100%.

1. Introduction

During the last ten years, state of the art SoC designs have increased dramatically in complexity and the prediction is that this will continue also for the years to come. The technical and economical benefit of this is expected to be enormous, at least if the full potential can be utilised efficiently. Several improvements have been envisioned.

From a functional and implementation point of view, a decrease of implementation and verification time by increased reusability through IPs is expected. Traditional on-chip busses are expected to be replaced by Networks-on-chip (NoC), since future generations of SoC designs require much higher volumes of communication than can be handled efficiently by traditional on chip busses [9].

From a manufacturing test point of view, a change from external test to BIST is envisioned, a change that is expected to solve several problems: [12].

1) Scalability. Usage of external testers is becoming very difficult and expensive, since test data volumes increase at the same time as the number of gates hidden behind each package pin also increases.
2) Deep submicron (DSM) processes must be tested at full clock-speed, which is very expensive [12], if even possible to accomplish with external testers.
3) A change from production test only to life time test is required, since tomorrows designs are foreseen to be used in more reliability demanding applications.

And from a manufacturing yield point of view, a change from “go/no-go-test” to “test and repair” is required, since the increased density of designs increases the probability for physical defects [13] [5].

This paper presents a BIST strategy for testing the NoC interconnect network, and investigates if the strategy is a suitable approach for the task. The test method is inline with the vision of a scalable test methodology. The intention is to use the BIST to detect faults and to be able to pinpoint the location of each defect, and finally autonomously use this information to reconfigure the architecture in such a way that full functionality, from a user point of view, can be remained. The efficiency of the method is evaluated in terms of testability and area overhead for the selected switch type.

The paper is organised as follows. Section 2 reviews related work and motivate the proposed method. Section 3 describes the target NoC architecture. Section 4 presents the functional test strategy. In section 5, test experiments are carried out. Finally, in section 6, conclusions are drawn and future work is discussed.

2. Related work

In [3], a method is proposed based on partial scan together with an IEEE 1500-compliant test wrapper [2]. All routers have an identical number of scan chains. The routers are tested in parallel by providing the same test stimuli for all routers, and using a single comparator per scan chain. Diagnostic is supported by the comparison logic. In [6], a similar strategy is presented to test multiple identical blocks in parallel; the difference is that BIST is used instead of external patterns. The proposals are limited to NoC architectures where multiple identical versions of switches always exist, which is not always true for a 2D-mesh NoC. As an example, a 3 by 3 2D-mesh only contains one centre switch with four links, i.e. there is no neighbour switch to compare and share test vectors with. In [14] there is a similar strategy presented as in [6], with the difference that external patterns are used, supplied directly through a switch.

Different test strategies for a commercial solution of a NoC are discussed in [4], including the possibility to repair them during manufacturing test. It is claimed that the error information must be collected and permanently stored inside the SoC. With this approach, error

[1] The Industrial Research School in Electronic Design in Sweden supports this work.

information is static and thus can not handle situations where the chip slowly degrades. In our approach, error data is updated at each start-up to allow for the possibility of a graceful degradation of the NoC over time.

[7] suggests that a wide variety of standard Design-For-Test (DFT) techniques can be used for NoC based designs, from BIST for FIFOs, to functional testing of wrapped switches. This approach has a high area overhead due to full scan and BIST.

In [8], an implementation of the IEEE 1149.1 boundary scan standard is proposed as a strategy to carry out hierarchical test, and enable diagnostics, of a 2D grid router structure. However, this approach does not enable test at full clock-speed in a NoC. The amount of extra logic added to enable serial shifting through all registers during test is also high.

In [10], it is proposed to add dedicated logic to enable analysis of response from each FIFO in the switch, however no test data is presented.

In general, all related work assumes one core per resource. Our method assumes that any number of cores can exist in a resource. Also, they assume that global synchronization can be achieved during test. With our method only local synchronization is required during test. We also propose to use the Network Interfaces (NIs) as autonomous test masters. In this paper, the NIs are used as BIST engines to test the NoC. However, the future goal is to extend the NIs to also be test masters at test of the resources as well. This type of strategy none of the other papers take into consideration. However, it has been proposed to use the NoC as a TAM during test of resources [15].

3. The NoC architecture used

The NoC architecture used is the Nostrum NoC [1]. The Nostrum backbone consist of a number of nodes organised in a 2D mesh Manhattan-like structure, as shown in figure 1.

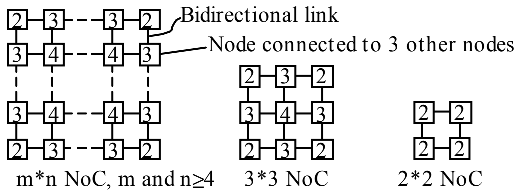


Figure 1. Different sizes of 2D mesh NoCs

Each node is composed of a switch (SW) and a resource (R). The switch and the resource are connected together through a Network Interface (NI), as shown in figure 2.

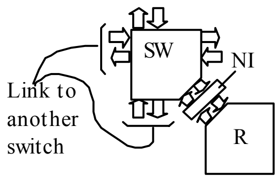


Figure 2. The Switch-Resource Interface.

The resources are logic modules that contain the various functions. A resource can comprise of one or more IPs, which can be connected together through a traditional bus. The signalling between any two switches and between a switch and a resource is packet based.

The functional blocks in each resource communicate directly with the off chip world through the IOs at the package. It is assumed that all on chip communication between resources is carried out through the NoC-structure. The NoC is only intended for on chip communication. It is also assumed that all communication from or to the chip always goes through a resource, never directly through a link. Any number of resources can have interfaces to packet pins.

The NoC itself is administrated by a dedicated operating system (OS) located, for example, in one of the nodes. The OS is considered to be a part of the NoC itself, and the presence of the OS is invisible outside the NoC.

During start-up, the OS configures the NoC and takes part of the analysis phase carried out after switches and links have been tested [11]. It is assumed that some or all resources are flexible enough to carry out more than one function, and that this degree of freedom in flexibility can be used to hide (from a user point of view) the presence of one or a couple of physical defects, either somewhere in the NoC or in a resource.

During functional operation, the main task for the OS is to work as a type of traffic police regarding usage and allocation of the NoC. Requests to open and close channels of communication are carried out by the NIs and granted or denied by the OS. The goal with this approach is to avoid possible and temporary overload of switches and links. The presence or absence of the OS does not affect the test phase of the proposed test methodology, it only affect how to carry out the analysis phase.

3.1 Switch Architecture

The function of the switch is to route packets over the full duplex links connected to each switch. The routing function carried out by each switch can be of any type. We use the Nostrum method of deflective (also called hot-potato) routing [1] in our experiments. Routing decisions are performed locally at each switch, guided by the control logic. A new routing decision is carried out every clock cycle.

A switch can be divided into two main parts, the “datapath” and the “controller logic”. In figure 3, all parts outside the square “control logic” are part of the datapath. The datapath is the part of the design where packets are passing across. The “controller logic” is the local intelligence in the switch, and decides which input to connect to which output.

A switch can be configured to use either an absolute or a relative addressing mode. Figure 3 show a switch configured to use a relative address mode. If an absolute address mode is selected, the blocks “incrementer” (+1) and “decrementer” (-1) can be omitted. The function of HC (Hop Counter) is explained in a later section.

The transmission time across a link is one clock cycle. The transmission time through a switch is also one clock cycle. If the registers at the input of a switch are omitted, the total transmission time across a link plus a switch is reduced to one clock cycle (with a longer critical path).

Deflective switches can be categorized into bouncing and non-bouncing ones, depending on whether feedback routing is allowed or not. For the datapath, the differences between the two types are that the bouncing version requires one more input at the multiplexor for connection of an input to an output. Figure 3 shows the bouncing version of a switch. Our experiments always use the bouncing version.

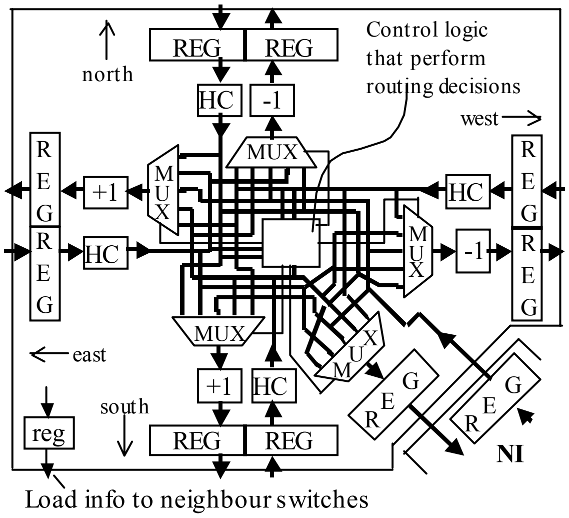


Figure 3. The architecture of a switch with four links

All technical data included in this paper is, if nothing else is stated, based on that 128 bit wide links are used and that registers are present both at input and output of the switches.

3.2 Transmission protocol

A complete message is transferred as one or more equally sized packets (flits). The size of a packet is the same as the width of a link. Each packet consists of a header and a payload, as shown in figure 4.

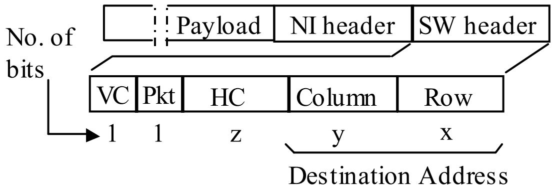


Figure 4. General structure of a packet

The outcome of a routing decision consists of connecting each input to one and only one output in the switch. The contents in the "SW header" together with the load status in the neighbour switches rules the router decision carried out by the control logic. Only the NI uses the "NI header" part.

The bit "Pkt" indicates if the packet is a real packet or just a set of "dummy" bits. VC stands for "Virtual Circuit" and is a flag that enables two levels of priority for a packet, high or low.

The HC is a "Hop Counter", and can be configured as being either a fixed or a dynamic priority flag.

If fixed, a switch never modifies the content in HC. In this configuration, HC is only used as a priority level flag.

If dynamic, the value of the HC is increased by one for each switch the packet arrives to. The higher value the HC has, the higher priority the packet gets. A switch can be configured to kill the packet (bit "pkt" is set to zero) when HC reaches a maximum value. This method allows removing a packet that due to an error in the network can not reach its destination.

The SW-header grows with increasing NoC size as shown in Table 1, which is valid when a relative address and a dynamic HC is used.

Table 1. How the "SW header" increases with NoC size

NoC	Bits in the "SW header"					Tot. No.
n*n	VC	Pkt	HC	y	x	of bits
n=2	1	1	3	2	2	9
3 ≤ n ≤ 4	1	1	4	3	3	12
5 ≤ n ≤ 8	1	1	5	4	4	15
9 ≤ n ≤ 11	1	1	6	5	5	18
12 ≤ n ≤ 16	1	1	7	5	5	19
17 ≤ n ≤ 22	1	1	8	6	6	22
23 ≤ n ≤ 26	1	1	9	6	6	23

The difference, in number of bits used, between relative addressing and absolute addressing is illustrated in figure 5.

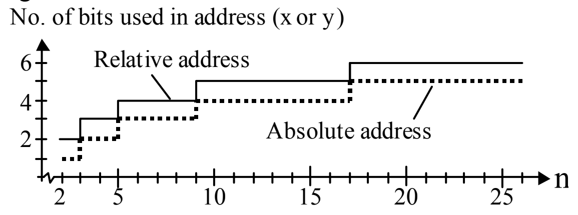


Figure 5. Absolute vs relative address size

4. A scalable NoC test strategy

The test methodology presented here is originally developed for usage with deflective switches in a 2D-mesh NoC. Even so, the test strategy should also be applicable for other types of switches and NoC architectures. The test is carried out automatically at power on or when activated by a broadcast command from the OS.

Our proposal to carry out a complete test strategy is as follows, where each step is carried out serially in order of appearance:

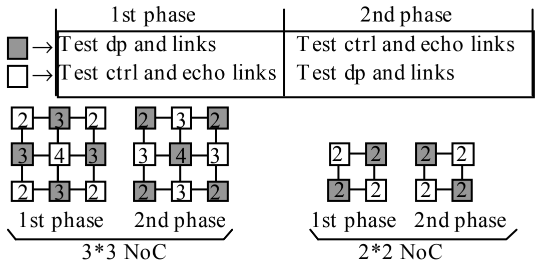
1) All NIs are tested concurrently with the help of BIST engines embedded in each NI.
2) All NIs autonomously and concurrently carry out test of the switches and links, and saves a signature.
3) The OS reads the signatures stored in each NI from step 1 and 2, and analyses the read signatures. If the NI detects one or more faults, but the NI needs further information to be able to point out the position of the fault, go to step 4, otherwise go to step 5.
4) The OS demands a proper set of NIs to carry out further tests of the switches and links. After ready, the OS reads and analyses the new signatures.
5) If needed, the OS orders the disconnection of the faulty part of the NoC.

Only step "2)" is further described and investigated in this paper, due to limited space.

The procedure to carry out step 2 can be divided into two equal main phases, as indicated by figure 6.

In Phase 1: The datapath with surrounding links are tested in the grey marked nodes. At the same time in the white marked nodes, the controller part is tested, and all links are set in an unconditional deflective mode.

In the second phase, the roles of the nodes are reversed.



	1st phase	2nd phase
▨ →	Test dp and links	Test ctrl and echo links
□ →	Test ctrl and echo links	Test dp and links

Figure 6. Checkerboard test strategy at NoC Level.

During the two test phases, all the NIs are working concurrently as local test masters. Each NI tests its own switch and all the links (2, 3 or 4) attached to the switch. Each NI supplies all test vectors needed, and collects test responses. We have not investigated if it is more optimal to distribute some analysis into the NIs instead of concentrating this activity to the OS only.

4.1 Testing the Links and Datapaths

The links connected to a switch is tested at the same time as the datapath in that switch is tested. This implies that links are fully tested twice and from different directions, as shown in figure 7. By doing so, after test no untested logic or nets remain in the boundary between the switches.

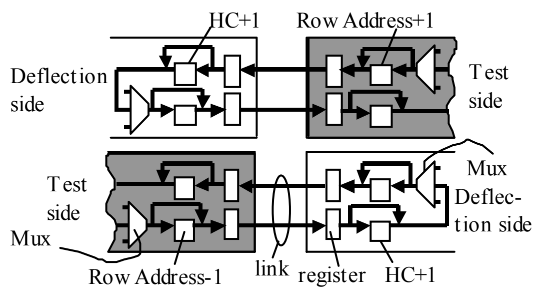


Figure 7. Test of a link between two neighbour switches.

The test is carried out at full clock-speed and in a functional-like mode. No extra logic for test is inserted into the datapath. During test and "deflection", the NI controls the multiplexors in the datapath, and not the controller logic.

4.2 Testing the Controller Logic

The only test logic inserted into the switches or links consists of a number of 2-input multiplexors, as shown in figure 8.

The inserted multiplexors creates a test wrapper to enable NI to carry out a stand alone BIST on the controller, and at full clock-speed. The NI also uses some of the multiplexors inserted during test of the datapath. The controller logic is also partly tested when the datapath is tested, since the NI can observe the outputs from the controller logic at the same time as the datapath (under test) feeds data onto the inputs of the controller logic. The experiments have not taken into consideration that this may improve the fault coverage.

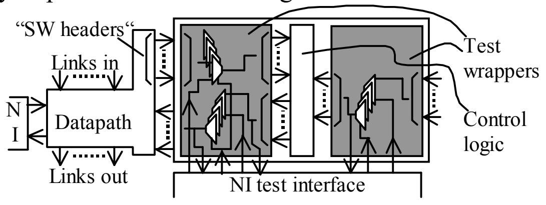


Figure 8. Control logic wrapped for test.

5. Experimental results

A set of experiments has been carried out to get an indication of if the proposed test strategy is good or not.

A commercial test tool has been used to carry out the different experiments, and ATPG generated test patterns have been used as test vectors. ATPG has been used since it is a time effective method to generate high quality test vectors. We believe that ATPG, in this particular case, is a useful approach to get an indication of what data to expect to achieve with a BIST solution. This expectation is based on the assumption that it should be possible to develop an optimal and dedicated BIST solution, since the datapath in a NoC is a highly regular structure, and that such a solution generates test patterns that are less pseudorandom compared to when using standard LFSRs.

The efficiency of the method is evaluated in terms of test time, fault coverage and area overhead achieved for different sizes of NoCs, and for different variants of deflective switches. The results included in the tables are extracted by using ATPG generated test vectors. The only test logic inserted into the switches and links are a limited number of 2-input multiplexors, which creates a test wrapper around the controller logic. All test vectors are applied, and all test responses are collected through the NI interface.

A commercial synthesis tool has been used to translate the logic tested from VHDL to a netlist in Verilog format to be used by the backend ATPG tool. The CMOS library used is tcbn90g (90 nm), revision 110. All design sizes stated in "eq. gates" are extracted by dividing area values, given by the synthesis tool, by 2.4 (the area of a 2-input NAND-gate). In the cases when gate size for BIST logic is stated, it includes logic for both test vector generation and test vector analysis. If nothing else stated, all values presented are valid for switches using relative address and with registers on both inputs and outputs to links.

The datapath is flip-flop dominated and the controller is combinational logic dominated.

5.1 Links and Datapaths

The smallest type of switch in a 2D-mesh is a corner switch, which is the only type of switch used in a NoC of size 2 by 2. Table 2 shows how data for a corner switch scales for different sizes of NoCs.

Table 2. Datapath in a corner switch (2 links).

NoC n*n	Datapath area Eq. Gates	Total test time in clock cycles (patterns)	Fault coverage (%)
n=2	5784	1126 (55)	99.94
3 ≤ n ≤ 4	5807	1177 (58)	99.94
5 ≤ n ≤ 8	5829	979 (47)	99.93
9 ≤ n ≤ 11	5848	1226 (60)	99.90
12 ≤ n ≤ 16	5858	1234 (64)	99.90
17 ≤ n ≤ 22	5876	1322 (68)	99.86
23 ≤ n ≤ 26	5886	1483 (77)	99.86

For NoCs larger than 2 by 2, the largest type of switch has four links connected to it. How such a switch scales for different sizes of NoCs is shown in table 3.

The right most column shows that fault coverage always is almost 100%. The second column from right side shows the number of clock cycles needed to completely test the datapath including the surrounding links for one switch. The total test time for all the switches in a NoC is this value multiplied by two, because of the checkerboard testing strategy described earlier in section 4. Values enclosed in parentheses show the number of patterns used during the test. A pattern is carried out across a number of clock cycles, since it takes a multiple of four clock cycles before a response returns to the NI.

A slight dip in test time is found at row "5 ≤ n ≤ 8" for both table 2 and 3. In table 3 is another dip found at row "23 ≤ n ≤ 26". The reason for this is probably due to that most of the test patterns created by the ATPG tool are created randomly, and that they just happened to fit properly for that particular configuration. This will be investigated in the future.

Table 3. Datapath in a full size switch (4 links).

NoC n*n	Datapath area Eq. gates	Total test time in clock cycles (patterns)	Fault coverage (%)
n=2	12016	2693 (129)	99.90
3 ≤ n ≤ 4	12063	2742 (129)	99.91
5 ≤ n ≤ 8	12102	2737 (134)	99.89
9 ≤ n ≤ 11	12144	2760 (134)	99.85
12 ≤ n ≤ 16	12164	2830 (147)	99.85
17 ≤ n ≤ 22	12202	3091 (153)	99.81
23 ≤ n ≤ 26	12227	2936 (152)	99.80

Table 4 shows how test time is affected when the addressing mode is changed from relative to absolute at the same time as HC is changed to a 3 bit fixed value priority flag. The two upper rows show the case when registers (just as earlier) are used at both inputs from and outputs to links. The difference in the bottom row is that the input registers from links have been removed.

Table 4. Absolute addressing and fixed HC (priority).

No. of links in switch	Datapath area Eq. gates	Total test time clk cycles (patterns)	Fault coverage (%)
2	5738	1000 (50)	100.00
4	11922	2550 (121)	100.00
4 (single reg)	8417	1919 (141)	100.00

We can see that for n ≤ 26, datapath with links can be tested in less than 6000 (2*2936) clock cycles, independently of the version of deflective switch used. Further more we can also see that test time increase slowly with size. When relative addressing is used, the increase for a corner switch is less than 5.5% and for a 4-link switch in average about 1.5% for every increasing value of n that leads to new bits. In the case of absolute addressing, test time is constant 5100 (2*2550) clock cycles for any n ≥ 3.

If registers at input from links are removed, shorter test times are achieved since the number of hops until a packet returns is reduced.

5.2 The Controller Logic

Table 5 shows how area and test time for a corner switch scales for different sizes of NoCs. The number of clk cycles is the same as the number of patterns needed for testing the controller.

The difference between table 5 and 6 is that in table 5 a dynamic HC is used. The HC scales in size according to table 1. In table 6, HC is changed into a 3-bit wide and fixed value priority flag.

Table 5. Control logic in a corner switch (2 links).

NoC n*n	Control area Eq. gates	Total test time (clk cycles)	Fault coverage (%)	BIST area Eq. gates
n=2	209	54	99.3	1275
3 ≤ n ≤ 4	228	72	99.4	1655
5 ≤ n ≤ 8	245	74	99.4	1868
9 ≤ n ≤ 11	273	81	99.5	2083
12 ≤ n ≤ 16	288	89	99.4	2208
17 ≤ n ≤ 22	301	91	99.5	2452
23 ≤ n ≤ 26	316	108	99.5	2663

The implementation of the control logic is hardly affected if the address mode is changed from relative to absolute; this is the reason why no table showing test data when using absolute address is included.

For every increasing value of n: A dynamic HC increase the test time for a corner switch with less than 13% and when HC is changed to always be 3-bits wide (priority coding), the increase is in average about 5%.

Table 6. Corner switch with a fixed HC size (3 bits).

NoC n*n	Control area Eq gates	Total test time (clk cycles)	Fault cov. (%)	BIST area Eq. gates
n=2	209	54	99.3	1275
$3 \leq n \leq 4$	213	61	99.4	1421
$5 \leq n \leq 8$	215	67	99.4	1630
$9 \leq n \leq 16$	228	61	99.4	1582
$17 \leq n \leq 26$	227	65	99.4	1783

We can conclude that for $n \leq 26$ the time to test the control logic is much shorter than the time it takes to test the corresponding datapath with surrounding links, thus the testing of the datapath is the limiting factor.

5.3 Discussions

The size of the BIST logic is much bigger than the control logic itself. Also, the BIST area increases faster with increasing NoC size compared to how the control logic area increases. On the other hand, a linear approximation of the area of the datapath BIST is ~12k gates, which is roughly 100% of the 4-link switch area and ~5% of the area occupied by the links between the switches (link area is about 20 times larger than the switch area in our NoC). However, the BIST area presented can be expected to be unnecessary large. The implemented BIST generates deterministic patterns (extracted from ATPG) and analyses the achieved response once every clock cycle against the expected response (generated by ATPG). So, the BIST area for the control logic can be reduced by using an LFSR based version instead, since the test time for the controller can be extended at least 10 times, without increasing the total test time of the NoC.

The example with the deterministic BIST indicates that it is possible to create dedicated and fast BIST structures. The datapath in a switch is highly regular, compared to the controller logic. Therefore, it should be possible to develop a dedicated BIST for the datapath, and that has an acceptable area overhead.

Another thing to take into consideration for the links is the use of a MAF (Maximum Aggressor Fault) strategy discussed by [16]. Our experience is that faults targeted by the MAF model is unlikely to occur, if a ground line is inserted between the individual lines in a link.

6. Conclusions

A scalable test methodology targeting 2D-mesh type of NoCs has been presented. The test is executed at full clock-speed and in a functional-like mode. It is carried out as a go/no-go BIST at start-up. On the NoC level the proposed method can locate physical defects down to the granularity of a switch, with its surrounding links. At the switch level, the position of a physical defect can be distinguished to be present either in the controller part or in the datapath/links.

We conclude that from test point of view absolute address mode is better than relative address mode, since test time is constant for a NoC greater or equal to 3*3, and up to large values of n*m.

Switches and links in a 2 by 2 NoC can be tested in less than 3000 clock cycles and NoCs up to 26 by 26 switches can be tested within 6000 clock cycles. If the addressing mode is changed from relative to absolute, the test time can be reduced to a constant of 2000 clock cycles for a 2 by 2 NoC and 5100 clock cycles for any 2D NoC with size $n \geq 3$. The test times can possibly be reduced further with a more careful selection of patterns instead of using an ATPG tool to generate them. This will be investigated in the future.

The total BIST area overhead, for a centre switch and links is limited to less than 7%. Optimising the BIST solutions should reduce the area overhead. How to do this will also be investigated in the future.

7. References

[1] M. Millberg, E. Nilsson, R. Thid and A. Jantch. *A Guaranteed bandwidth using looped containers in temporary disjoint networks within the Nostrum network on chip*. Proc. DATE, pp 890-895 Vol. 2, 2004.

[2] Standard IEEE 1500. *Standard Testability Method for Embedded Core-based Integrated Circuits*. IEEE 2005.

[3] A. M. Amory, E. Brião, É Cota, M. Lubaszewski, F. G. Moraes. *A Scalable Test Strategy for Network-on-Chip Routers*. Proc. of ITC 2005.

[4] B. Vermeulen, J. Dielissen, and K. Goossens. *Bringing Communication Networks on a Chip: Test and Verification Implications*. IEEE Communications Magazine, vol. 41-9, 2003, pp. 74-81.

[5] Y. Zorian. Embedded Infrastructure IP for SOC Yield Improvement. DAC 02, pages 709-712.

[6] K. Arabi. *Logic BIST and Scan Test Techniques for Multiple Identical Blocks*. IEEE VLSI Test Symposium, 2002, pp. 60-68.

[7] R.Ubar and J Raik. *Testing Strategies for Network on Chip*. Book: Network on Chip, A. Jantsch and H. Tenhunen, Eds. Kluwer Academic Publisher, 2003, pp. 131-152.

[8] C. Aktouf. *A Complete Strategy for Testing an on-Chip Multiprocessor Architecture*. IEEE Design & test of Computers, vol. 19-1, 2002, pp. 18-28.

[9] C. A. Zeferino, M. E. Kreutz, L. Carro and A. A. Susin. *A Study on Communication Issues for System-on-Chip*. Proceedings of the 15th Symposium on Integrated Circuits and System Design (SBCCI), 2002.

[10] Panda et al. Design, Synthesis, and Test of Networks on Chips. IEEE D&T, vol. 22, issue 8 2005, pp404-413.

[11] K. Petersén, J Öberg. Utilizing NoC Switches as BIST-structures in 2D-Mesh Network-on-Chip. Future Interconnects and Network on Chip Workshop, 2006.

[12] Y. Zorian. Testing the monster chip. IEEE Spectrum, July 1999, pages 54-60.

[13] Y. Zorian. Embedded Memory Test and Repair: Infrastructure IP for SoC Yield. International Test Conference 2002, pages 340-349.

[14] C. Grecu, P. Pande, B. Wang, A. Ivanov, R. Saleh. Methodologies and Algorithm for Testing Switch-Based NoC Interconnects. IEEE DFT 2005, pages 238-246.

[15] E. Cota et al.The Impact of NoC Reuse on the Testing of Core-based Systems. VTS 2003, pages 128-133.

[16] C. Grecu, P. Pande, A. Ivanov, R. Saleh. BIST for Network-on-Chip Interconnect Infrastructures.