A Concurrent Testing Method for NoC Switches

Mohammad Hosseinabady, Abbas Banaiyan, Mahdi Nazm Bojnordi, Zainalabedin Navabi
Electrical and Computer Engineering, University of Tehran14399 Tehran, Iran
mohammad@cad.ece.ut.ac.ir, {a.banaiyan, m.bojnordi}@ece.ut.ac.ir, navabi@ece.neu.edu

Abstract

This paper proposes reuse of on-chip networks for testing switches in Network on Chips (NoCs). The proposed algorithm broadcasts test vectors of switches through the on-chip networks and detects faults by comparing output responses of switches with each other. This algorithm alleviates the need for: (1) external comparison of the output response of the circuit-under-test with the response of a fault free circuit stored on a tester (2) on-chip signature analysis (3) a dedicated test-bus to reach test vectors and collect their responses. Experimental results on a few test benches compare the proposed algorithm with traditional System on Chip (SoC) test methods.

1. Introduction

RECENT advances in IC design methods and manufacturing technologies allow designers to integrate the complete system on a single chip. This so-called SoC product class is a yet-evolving design style that integrates technology and design elements from other system driver classes into a wide range of high-complexity, high-value semiconductor products [1]. Even though commercial products currently exhibit only a few integrated cores [2], in the next few years, technology will allow the integration of thousands of cores, making a large computational power available.

Today's System on Chip (SoC) technology can achieve unprecedented computing speed that is shifting the IC design bottleneck from computation capacity to communication bandwidth and flexibility[3]. Also, since communication buses between the cores are not sufficiently scalable, bus-based SoCs (the common type of SoCs) cannot handle this high volume of communication between the cores in the SoCs. In addition, these SoCs cannot be used for high speed serial communications. Moreover, as volume of the data communication on the chip increases, the power consumption increases. Therefore, a scalable communication infrastructure that better supports the trend of state-of-the-art SoC integrations is required. Thus, recent researchers use packet-switched micro-network on a chip, so called NoC, as a scalable communication media. The basic idea is borrowed from traditional large-scale multi-processors and the wide-area network domains.

It is important that SoC designers consider a test methodology for their new SoC architectures. Like other SoCs, an NoC has to be tested for manufacturing defects. One of the main problems for testing an SoC is the access to the cores during the test process [4]. A number of solutions have been presented in the literature [5] [6] to solve this problem while minimizing test costs, mainly pin count and test time. Most methods rely on scalable and easy-to-design test access mechanisms (TAMs) to reduce design time. For those methods, bus-based TAMs are usually chosen. Since the NoCs consist of functional cores, switch cores and interfaces, test methodologies should be performed on each of these three parts.

This paper proposes a novel concurrent test methodology for testing the switches of an NoC. The rest of the paper is organized as follows. Related works on testing the SoC and NoC are reviewed in Section 2. Section 3 introduces some preliminaries and definitions. Our proposed algorithm is discussed in Section 4. In our proposed algorithm, a wrapper architecture is proposed for the switches and this is discussed in Section 5. In Section 6 test time calculation will be described. The proposed switch architecture will be discussed in Section 7. Finally the paper ends with experimental results and conclusions.

2. Related Works

A packet switching communication-based TAM for an SoC has been proposed in [7]. The proposed TAM model is called NIMA, and it is defined for the test task. Thus, routing and addressing strategies are defined considering only the test requirements of each system. Test and verification challenges for system chips that utilize on-chip network has been presented in [8]. In [8] an NoC has been exemplified by Philips' ÆTHEREAL NoC architecture. It shows the particular advantages of using an NoC for both testing and verifying the network components, and testing and verifying the other components of the SoC. The reuse of functional connections during test has been suggested in the literature [9][10]-[13] to reduce test costs in terms of area and pin overhead. For those methods, a core-to-core connection model is assumed. The impact of the reuse of an on-chip network for the test of core-based systems is described in [14], [15]. These references formalize a reuse strategy aiming at minimizing the system test costs. A network-based embedded core testing architecture using the star-connected OCN has been proposed in [16]. This architecture provides the scalability and configurability for system integration and core-based test approach. All of these test architectures need the following methods of fault detection: (1) external comparison of the output response of the circuit-under-test with the response of a fault free circuit stored on a tester, (2) on-chip signature analysis and (3) a dedicated test-bus to reach test vectors and collect their responses. Each of these methods has certain shortcomings that are alleviated by using output comparison for identical circuits. The external comparison method has the memory overhead of storing fault free responses. Signature analysis may suffer from aliasing, or error masking, that may cause a fault to go undetected even if it is propagated to an output of the circuit-under-test. Dedicated test bus which currently adds to the SoC has area and power overhead.

3. Preliminaries

NoCs can be defined as a set of structured routers and point-to-point channels interconnecting the processing cores of an SoC, in order to support communication among them. An NoC consists of two parts: switches and core interfaces. A network of the switches provides interconnection between cores. NoCs typically use the message-passing communication model, and the processing cores, attached to the network, communicate by sending and receiving request and response messages. These messages are called packets. Each packet is composed of a header, a payload and a trailer.

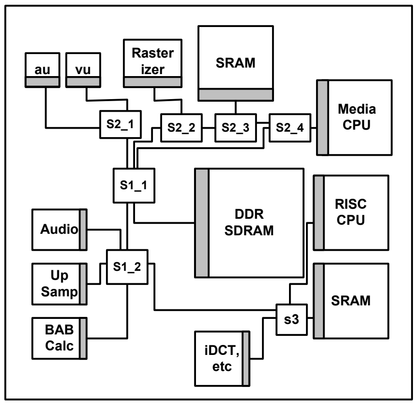

Figure 1. MPEG Decoder

A *topology* can be represented as an undirected connected graph $G(V,E)$, where $V=\{v_1,v_2,...v_n\}$ is the set of vertices or nodes and $E=\{e_1,e_2, ...e_m\}$ is the set of edges or links in the corresponding network. We use $N(v)$ to denote the set of neighbors of node $v\in V$. Also, we denote by $\delta(v) = |N(v)|$ the number of such neighbors. Some examples of topologies include regular tile-based topologies: grid, torus, hypercube, ring, multi-stage and fat-tree which are suitable for interconnecting homogeneous cores in a multiprocessor chip, and irregular topologies which are suitable for heterogeneous cores on the SoC having varied functionality [2].

A *topological* centre of graph $G(V,E)$ is defined as node $v\in V$ that is closest to any other node in the network graph, i.e., node v which minimizes $max_{u\in V}\, d(v,u)$, where $d(v,u)$ is the distance between nodes v and u.

Figure 1 shows an SoC implementation of an MPEG4 decoder, using an NoC for its on-chip communication. In the MPEG4 design, many of the cores communicate with each other through the shared SDRAM. So a large switch (S1_1) is used to connect the SDRAM, while smaller switches are instantiated for the other cores [2]. Large switches can be built by wiring smaller switches which means that we can have a fixed type switch and use it for all switch sizes. Therefore each NoC can be restructured as an NoC with identical switches. Even though, reconstructing the NoC to an identical-switch-based NoC may cause more area overhead for the NoC, but it may distribute network traffic and alleviate communication load over the switches. Reconstructing the NoC by the identical switches should be performed according to the design requirements and constraints.

4. Proposed Work

We propose a novel concurrent test methodology for testing the switches of an NoC. As mentioned above, we assume that all switches in the NoC are identical. So these identical blocks can reuse the same set of test vectors. Our approach broadcasts these test vectors through the network to be applied to these identical blocks. The proposed scheme helps obtain good reliability of switches, since switches are tested in a real integrated platform. Fault detection based on comparison of output responses of identical circuits reusing existing connections alleviates the need for each of the traditional inefficient methods of fault detection mentioned in the section on related works.

Our test algorithm follows the scan-based methodology for testing the switches in the NoC. Since in our NoC architecture, all switches are identical, broadcasting the test vectors, in the form of packets, through the network is sufficient for testing them. The most immediate switch (es) connected to the source of test data is (are) called *Test Access Switch(es), (TAS)*. From TAS all test data are broadcasted.

We assume a test source, connected to TAS, generates the test vectors serially and provides them to the TAS. TAS not only delivers the obtained test vector to its neighboring switches but also feeds them into its own scan-chain, simultaneously. After shifting a test vector into the scan-chain of the switch is finished, the switch captures the results of its combinational part in the scan-chain in one clock cycle (*capture cycle*). Because all switches should generate the same result for a specific test vector, the Test Access Switch broadcasts its result to its neighbors for comparing them with the corresponding results of their switches. This way, we are able to detect possible faults.

Other switches, in addition to this comparison, perform the same function as the TAS in sending the received test vector and their captured results to their neighbors. This excludes the boundary switches. A boundary switch is a switch in the NoC that has at most one switch in its neighborhood. Such a switch does not send test vectors and corresponding results to its neighbors. When a switch detects a discrepancy between the received and its captured results, it notifies the existence of a fault through a signal to its feeding switch. This signal is passed through the network until it reaches the TAS. Then TAS reports the existence of the fault in the NoC to the outside of SoC.

For example, Figure 2 shows a network of switches and communication links among the switches of an NoC. This network consists of four nodes (switches) and three edges (channels). As depicted here, Switch 2 (shaded switch) is chosen as the Test Access Switch and receives test vectors from a test source.

Suppose the length of each test vector is four; so, four clock cycles are needed to shift the test vector into the scan chain of a switch. Also, assume the length of the result of the switch to the test vector is five; therefore, five clock cycles are needed to shift out the result.

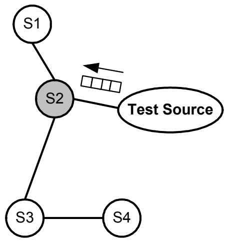

Figure 2. Network Graph of Figure 1

Figure 3 illustrates the timing details of broadcasting test vectors through the switches of Figure 2. In this figure, Row 1 shows shifting a test vector from the test source into the Test Access Switch (TAS) (S2) and its capture cycle (CapCycle). Row 2 shows broadcasting the test vector to Switches 1 and 3. Furthermore, this row shows that TAS broadcasts its result to its neighbors after an idle clock cycle, which is needed to synchronize TAS with its neighbors.

Row 3 and 4 show shifting a test vector from the source into Switches 1 and 3 (S1, S3), respectively and their capture cycles. After the capture cycles, these switches compare the received result from the source point switch with their own results. If a switch detects a discrepancy during this comparison, it stops its test process and issues a fail signal to its predecessor switch to notify the existence of a fault in the NoC. Rows 5 and 6 are similar to Rows 2 and 3, respectively. Switches 1 and 4 are the boundary switches, so they do not broadcast test vectors.

In this example, 12 clock cycles are required to test all switches in the NoC by applying a test vector. Using our method, as shown in this example; the switches are tested concurrently in the test mode.

Finding the TAS and the broadcasting tree impacts the test application time and power consumption in the NoC during the test session. In broadcasting test data, the main objective is to minimize usage of the resources by sending test data from one or more source points to multiple destinations. Examples of resources which are desired to be minimized include bandwidth, time and connection costs. Minimizing usage of the resources decreases the test application time for NoC elements.

Our method broadcasts test vectors through the NoC in the form of packets, called *test packets*. The most basic way of sending test vectors through a network graph is using *flooding* [17]. With this technique, a node that receives a test vector sends packets to its adjacent nodes through all its adjacent links. If node v receives packet p from node u for which it is not the destination, then v first checks if p was received before. If this is true, the packet does not need to be sent again. Otherwise, node v just re-sends the packet to all other adjacent nodes (excluding u). It is clear that after at most n such steps (where n is the number of nodes in the network), a test packet will reach all nodes, including the destinations. Thus, the algorithm is correct. The number of messages sent by each node is at most n. The number of messages received by v is at most $n\delta_G(V)$.

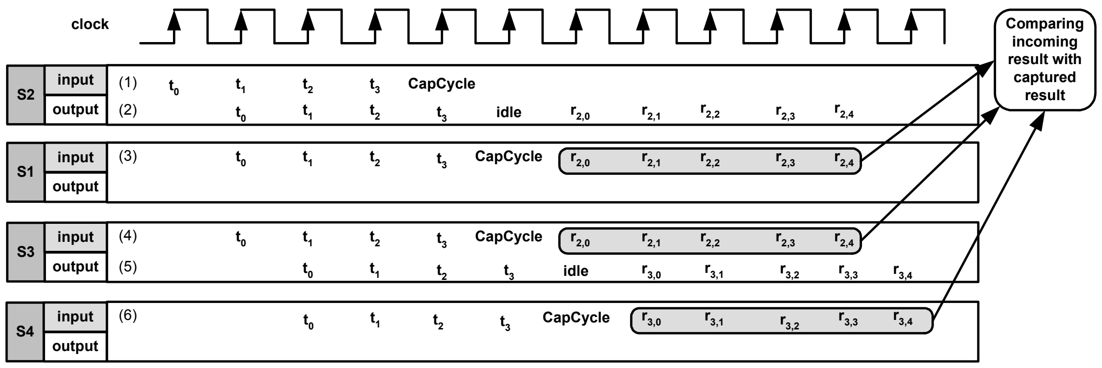
clock
Comparing incoming result with captured result
S2
input
output
(1)
(2)
t0 t1 t2 t3 CapCycle
t0 t1 t2 t3 idle r2,0 r2,1 r2,2 r2,3 r2,4
S1
input
output
(3)
t0 t1 t2 t3 CapCycle r2,0 r2,1 r2,2 r2,3 r2,4
S3
input
output
(4)
(5)
t0 t1 t2 t3 CapCycle r2,0 r2,1 r2,2 r2,3 r2,4
t0 t1 t2 t3 idle r3,0 r3,1 r3,2 r3,3 r3,4
S4
input
output
(6)
t0 t1 t2 t3 CapCycle r3,0 r3,1 r3,2 r3,3 r3,4

Figure 3. Timing diagram of broadcasting test vectors through NoC of Figure 2

This method of packet routing (flooding) is simple, but very inefficient. The first reason is that it uses more bandwidth than required, since many nodes that are not in the path to the destination will end up receiving a sent packet. Second, each node in the network must keep a list of all packets that it sends, in order to avoid loops. This makes the use of flooding prohibitive for all but very small networks.

For solving the problem of the flooding algorithm, test vectors are inserted at the topological center of the network. This is a switch with a minimum distance from other switches. The inserted test vector is multicast from this point through a minimum multicast tree. In this tree a node that receives the test vector forwards it to its neighbors in the tree. For minimizing the test application time, we find the topological center of the network graph and then find the minimum spanning tree rooted at this topological center of graph.

The topological center of a graph can be computed easily if all shortest paths among nodes are known. Using Dijkstra's algorithm [17], the shortest paths between nodes is obtained. To find the optimum broadcast routing (*minimum broadcast tree*), we use the Reverse Path Forwarding algorithm [18] statically.

5. Proposed Wrapper for Switches

Test vectors are broadcast in the form of packets. There are two types of packets: *data packet*, and *test packet*. One bit in the packet distinguishes between these types. Data packets are used during normal mode of the NoC to perform the communication among the NoC. Test packets are used during test mode of the NoC to test the switches. The test packet consists of two parts: *test vector* and *response* (Figure 3).

In order to apply our proposed method to an NoC, the NoC switches must be equipped with certain features. As mentioned above, each switch in the test mode should receive test vectors from one of its neighbors, shift the test vector into its scan chain and send this test vector and its response to the next switch. Furthermore, each switch of the network, except the switch at the access point, should be able to compare the received result from its neighbor with its own.

A wrapper circuit surrounding the switch performs the above said task. The wrapper has no effect on the switch functionality during the NoC normal mode of operation. Figure 4 illustrates the structure of our proposed wrapper. This wrapper comprises a controller, two scan chains for inputs and outputs of the switch, an XOR to compare the results, and some multiplexers to direct the output stream.

The wrapper controller checks the first bit of all newly arrived packets. Once, a test packet is detected at an input port, the controller sets the test mode for the switch. In this mode, the switch scan becomes active. The two scan chains of the wrapper concatenate the inputs and outputs of the switch to the internal scan chain of the switch. At the same time that a switch scan is active its controller sends the same test data to some of its neighbors. The neighbors to which test is sent are decided by the Minimum Broadcasting Tree algorithm. The implementation of this algorithm requires extra hardware in the switch that will be discussed in Section 7. After shifting a test vector into the scan chain is completed, the controller captures the response of the switch in the scan chain (capture cycle). This is an idle cycle.

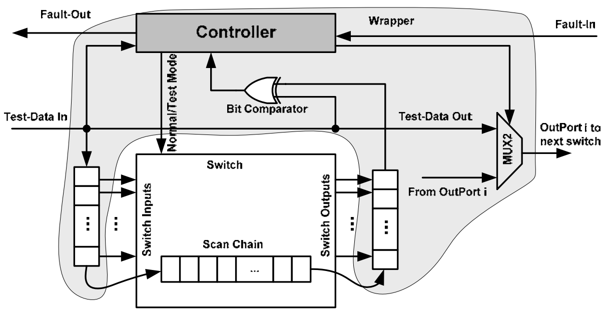

Figure 4. Wrapper Structure

In each test packet, the test vector is followed by the response of the Test Access Switch (response part). After the capture cycle, the response part of the input packet is compared with the responses in scan chain, serially. Once, the controller detects a discrepancy between these two streams, a fault signal is returned back to the predecessor switch to report the fault. Each switch that is notified of the existence of a fault also notifies its predecessor switch of the fault. Finally, the test access switch indicates that the chip is faulty.

6. Test Time Calculation

Let N be the number of test vectors for a switch and n_l, n_i, and n_o be the length of internal scan chain, number of inputs, and number of outputs, respectively. Furthermore, let d be the depth of a minimum broadcast tree.

Switches at the same distance to Test Access Switch are tested simultaneously. Each switch under test gives the test packet to its successor neighbor switches in the minimum broadcasting tree only by one clock delay. Thus, the test application time of an NoC, for one test vector, is equal to the test application time of a switch, for that test vector, plus the depth of the minimum broadcasting tree.

Therefore, the total test application time for an NoC can be written as:

$$TestTime = \left(n_l + \max\left(n_i, n_o\right) + d + 1\right) * N$$

7. A Proposed Switch

The same as switches in wide-area networks, switches in the NoC consist of I/O ports, buffers and a switch core. In order to implement our test scheme, a simple switch has been designed. Figure 5 shows part of this switch that routes an incoming packet to one of its output ports. Switch buffers are placed at the output ports and a distributed crossbar routing architecture is used in the input ports as the switch core.

Packet routing in our proposed NoC is performed by source routing. When a packet arrives at an input port of the switch, beginning *k(=2)* bits of the packet (*header*) are removed and used as the address for determining the outgoing port to the destination core on the SoC. A controller module in the switch controls operation of its different parts. A 4x4 switch based on this architecture is shown in Figure.6. For testability, two flags MTF (*Master Test Flag*) and STF (*Slave Test Flag*) are added to each IO port of the switch that must be configured before testing. Using these flags, the desired spanning tree is specified in the on-chip network structure. Using these two flags, the switch recognizes broadcasting test packets (incoming packets from the port whose MTF is ON) and sends them to the ports that their STF flags are ON. If the packet is received at a port whose STF is ON, the packet contains the result of a test and should be routed to the TAS.

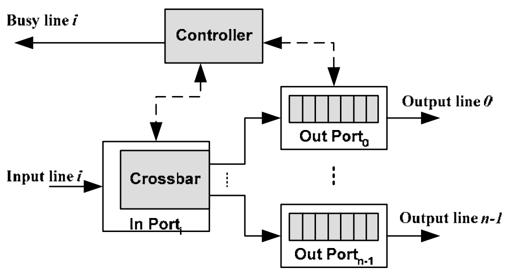

Figure 5. In Port and Out Port modules in the switch

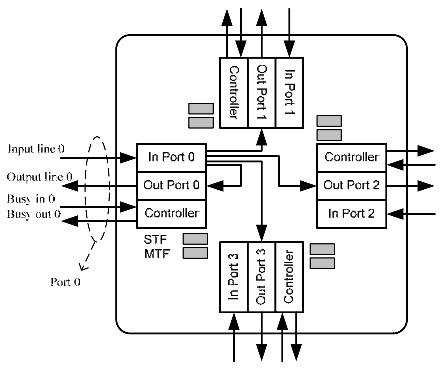

Figure.6. The 4x4 multi-purpose switch

8. Experimental Results

For experimental results, five switches with 3, 4, 5, and 6 ports are designed, synthesized, and their test vectors are generated using ATALANTA[19] (a public domain combinational test generator tool). Table 1 contains information about these switches. The first and second columns show the buffer size and the number of switch ports, respectively. The number of gates for each switch after synthesis is shown in the third column. The fourth column shows area overhead of the wrapper and the required modification for testability on each switch. Other columns contain test statistics of switches. Test power and test coverage are calculated by simulation, based on the number of transitions that take place while testing the gate level model of the circuit. In this table, experimental results are summarized for switches of different packet sizes. For all of these switches port size of 4 is considered in testing. Also other port sizes are considered just for the switch with 16-bit packet size.

Table 1. Switches statistics and their test results

Switch Statistics				Test Results			
# of Bits	# of Ports	# of Gates	Overhead	Coverage	Total vectors	Scan Length	Test Power
8	4	6419	5.84%	93.91%	841	353	1
16	3	6637	6.91%	93.97%	1193	364	1.55
	4	8849	5.16%	93.97%	1481	485	2.60
	5	14466	3.90%	93.87%	2167	766	6.33
	6	21601	2.88%	93.86%	3071	1129	13.61
32	4	13565	6.08%	95.20%	2723	713	6.89

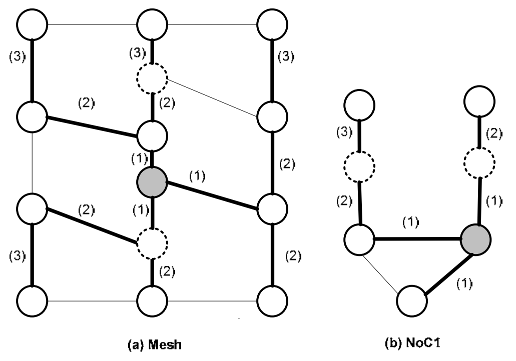

Figure 7. Different topologies for Video Object Plane Decoder

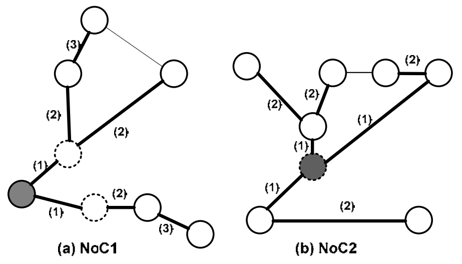

Figure 8. Different topologies for MPEG4 Decoder

To qualify efficiency of our method in an sample NoC, we have considered four different topologies of NoCs that have been proposed in [20]. Figure 7 shows two different topologies for the Video Object Plane Decoder; also Figure 8 shows two different topologies for the MPEG4 Decoder. The dashed switches are added to the original topologies in order to reconstruct them only by identical 4x4 switches. The bold links in these figures indicate the minimum broadcasting tree of each topology. The test access switches are shown by shaded nodes.

For comparing our method with a traditional SoC test method, a bus-based test architecture for the NoCs in Figure 7 and Figure 8 is implemented. For each SoC design, a bus is considered to feed the test vectors to the switches. In order to reduce extra hardware overhead in this method, we have also considered comparing output responses of the identical switches for the test result.

Table 2 compares test time (number of clock cycles needed for testing) and power consumption (number of transitions) of our method with the results obtained by applying the bus-based SoC test method. As shown in this table, our proposed algorithm has improved test application time and power consumption of up to 37.9% and 4.8%, respectively. This table also shows the area overhead as compared to the original NoC architecture. In some cases the modified architecture has less area rather than the original architecture. This is due to reconstruction of the original architecture using the identity switches.

Table 2. Comparison of our method with bus based testing

		Bus-based Testing		Proposed Method			Improvment	
SoC Structure		Test Time	Power	Test Time	Area Overhead	Power	Test time	Power
Video Object Plane Decoder	Mesh	651100	7.62522e9	404892	5.5%	7.26211e9	37.8%	4.8%
	NoC1	651100	4.17412e9	404892	11.6%	4.14510e9	37.8%	0.7%
MPEG 4 Decoder	Mesh	651100	7.62522e9	404892	5.5%	7.26211e9	37.8%	4.8%
	NoC1	651100	4.17896e9	404892	-3.2%	4.14579e9	37.8%	0.8%
	NoC2	651100	4.17389e9	404064	-3.2%	4.14488e9	37.9%	0.7%

Our proposed algorithm has some advantages over the bus-based testing methods. The algorithm uses existing connections between switches while bus-based testing algorithms need to add buses to the original design. In contrast with the proposed algorithm, the bus-based algorithm imposes more area and delay overhead to the original design. Also, our proposed method is more flexible and extendible to large NoCs.

9. Conclusions

To meet the requirements of high speed testing for network on chip components, a new algorithm for testing NoC switches has been developed. High speed testing has been achieved by broadcasting test packets to switches through the existing on-chip networks. To detect faults, output responses of identity switches are considered to be compared with each other. The algorithm eliminates the main drawbacks of the previous bus-based NoC testing methods. According to the experimental results, by use of the proposed testing method, about 38% improvement in test time and up to 5% reduction in power consumption is gained.

References

[1] D. Edenfeld, A.B. Kahng, M. Rodgers and Y. Zorian, "2003 technology roadmap for semiconductors," *Computer*, vol. 37, no.1, pp. 47-56, 2004.

[2] D. Bertozzi and L. Benini, "Xpipes: a network-on-chip architecture for gigascale systems-on-chip," *IEEE Circuits and Systems Magazine*, vol. 4, no. 2, pp. 18–31, 2004.

[3] A. Bona, V. Zaccaria, and R, Zafalon, "System level power modeling and simulation of high-end industrial network-on-chip," *Design, Automation and Test in Europe*, pp. 318-323, 2004.

[4] Y. Zorian, "Test requirements for embedded core-based systems and IEEE P1500," *Proc. of International Test Conference*, pp. 191–199, 1997.

[5] T.J. Chakraborty, S. Bhawmik and C.-H. Chiang, "Test access methodology for system-on-chip testing," *Proc. of International TECSWorkshop*, pp. 111–117, 2000.

[6] V. Lyengary, K. Chakrabarty, M.D. Krasniewski and G.N. Kumar, "Design and optimization of multi-level TAM architectures for hierarchical SOCs," *Proc. of 21st IEEE VLSI Test Symposium*, pp. 299-304, 2003.

[7] M. Nahvi and A. Ivanov, "Indirect test architecture for SoC testing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 23, vo. 7, pp. 1128-1142, 2004.

[8] B. Vermeulen, J. Dielissen, K. Goossens, and C. Ciordas, "Bringing communication networks on a chip: test and verification implications," *IEEE Communications Magazine*, vol. 41, no.9, pp. 74-81, 2003.

[9] E. Cota, L. Carro, F. Wagner, and M. Lubaszewski, "Power-aware NoC reuse on the testing of core-based systems," *Proc. of International Test Conference*, pp. 612-621, 2003.

[10] E. Cota, L. Carro, and M. Lubaszewski, "Reusing an on-chip network for the test of core-based systems," *ACM Transactions on Design Automation of Electronic Systems*, vol. 9, no. 4, pp.471–499, Oct. 2004.

[11] I. Ghosh, N. Jha, and S. Dey, "A low overhead design for testability and test generation technique for core-based systems," *Proc. of International Test Conference*, pp. 50–59, Nov. 1997.

[12] M. Nourani and C. Papachristou "Structural fault testing of embedded cores using pipelining," *IEEE JETTA*, vol. 15, no. 1-2, pp. 129–144, Aug.-Oct. 1999.

[13] E. Cota, L. Carro, A. Orailoglu, and M. Lubaszewski, "Test planning and design space exploration in core-based environment," *IEEE, Design, Automation and Test Conference in Europe*, pp.483–490, Mar. 2002.

[14] E. Cota *et al.*, "The impact of NoC reuse on the testing of core-based systems," *Proc. of 21st IEEE VLSI Test Symposium*, pp.128-133, 2003.

[15] E. Cota *et al.*, "Reusing an on-chip network for the test of core-based systems," *ACM Transactions Design Automation of Electronic Systems*, pp. 471-499, 2004.

[16] K. Jong-Sun *et al.*, "On-chip network based embedded core testing," *Proc. of IEEE International SOC Conference*, pp. 223-226, 2004.

[17] T. H. Cormen, C. E. Leiserson, and C. Stein, *Introduction to algorithms*. McGraw-Hill, 1990.

[18] Y. K. Dalal and R.M. Metcalfe, "Reverse path forwarding of broadcast packets," *Communications of the ACM*, vol. 21, no. 12, pp.1040–1048, 1978.

[19] H.K. Lee and D.S. Ha, "On the generation of test patterns for combinational circuits," *Tech. Rep.* 12-93, Dept. of Electrical Eng., Virginia Polytechnic Institute and State University, 1993.

[20] A. Jalabert, S. Murali, L, Benini. G. De Micheli, "xpipes compiler: A tool for instantiating application specific Networks on Chip," *Proc. of Design Automation and Test in Europe Conference*, pp. 884-889, 2004.