An exhaustive test strategy based on flooding routing for NoC switch testing

Elnaz Koopahi
University of Isfahan
6 PUBLICATIONS **29** CITATIONS
SEE PROFILE

Mahmood Fathy
Iran University of Science and Technology
354 PUBLICATIONS **4,072** CITATIONS
SEE PROFILE

Zainalabedin Navabi
Worcester Polytechnic Institute
340 PUBLICATIONS **2,054** CITATIONS
SEE PROFILE

An Exhaustive Test Strategy Based on Flooding Routing for NoC Switch Testing

Mahshid Sedghi[1,2], Elnaz Koopahi[2], Armin Alaghi[2], Mahmood Fathy[1] and Zainalabedin Navabi[2]

[1]*Computer Engineering Department*
Iran University of Science and Technology
16846_13114, Tehran, Iran

[2]*Electrical and Computer Engineering Department*
Faculty of Engineering, Campus #2
University of Tehran, 14399 Tehran, IRAN

[1]*mahfathy@iust.ac.ir;* [2]*{mahshid, koopahi, armin, navabi}@cad.ece.ut.ac.ir*

Abstract

An exhaustive test strategy based on flooding routing for NoC switches is presented in this paper. Some test-mode hardware is added to the switches to enable them to multicast their incoming packets thus examining all existing routes in the network. A test environment based on high-level switch faults is used which injects high level routing faults into the NoC-Under-Test and considers the total number of the packets received at the destination for fault detection. Experimental results show that that 100% fault coverage can be achieved with a small amount of test hardware overhead within an acceptable test time.

1. Introduction

Network-on-a-Chip (NoC) has emerged as a new design paradigm to replace traditional bus architectures and to provide higher bandwidth, lower latency and a scalable communication infrastructure for modern systems-on-a-chip [1]. An NoC architecture is composed of a set of structured switches and point to point channels interconnecting the processing cores of a SoC in order to support communication among them [2].

As with other hardware structures, this new trend in hardware, requires testing. On the one hand, NoC testing uses conventional hardware testing methods such as scan and BISTs, and on the other hand, network switching methods can be used for NoC testing.

While scan testing NoC switches and processing elements targets stuck-at faults, higher level testing using data packets target higher level routing and switching faults. Analysis of test methods targeting stuck-at faults requires exact hardware model of the circuit under test, while higher level fault analysis requires hardware models that can be injected by such switch level or routing faults.

The work described in this paper focuses on high level testing of NoC structures, and for these structures, the switches are the primary focus. The paper discusses a high-level fault model and a high-level circuit representation in which our faults can be injected. This platform enables us to evaluate various test strategies that can better find high level NoC switch faults.

The test strategy discussed in this paper is based on flooding routing, that is similar to that used in packet switching networks. Using an NoC source, our exhaustive test method sends packets in all directions to NoC switches for testing them for direction faults. Analysis of the packets received at the sink reveals direction faults in the switches. Our exhaustive method tests all switches for all direction faults.

The remainder of this paper begins by reviewing some related works on NoC switch testing methods in Section 2. Section 3 presents some description of the NoC architecture we will use in this paper, the test platform we will use for our simulations, our proposed high level fault model and some previously proposed test strategies. We then describe our proposed exhaustive test strategy and the operation of the NoC in test mode applying this strategy in Section 4. Disadvantages of this strategy will also be mentioned in this section. Finally, the results of comparison of test time, total number of the packets, power consumption and hardware overhead added to switches between different test strategies will be given in Section 5 followed by conclusions in Section 6.

2. Related Works

Most of the work done on NoC switch testing is based on conventional testing methods such as scan and BISTs. Consider works done in [3, 4, 5] as examples of scan-based test method.

The work done in [3] describes a scan-based concurrent testing method for NoC switches. There is a switch in the network called TAS (Test Access Switch) to which a test source is connected. A test source generates serial test vectors and provides them to TAS. TAS delivers the obtained test vectors to its neighboring switches as well as shifting them into its own scan chain. TAS also broadcasts its results to its ports according to the destination field of the neighbors for comparing them with the corresponding results of their switches.

Another test methodology for NoC switch testing is proposed in [4]. This methodology makes use of the intra-switch regularity among ports of a switch and inter-switch regularity among routers of switches for reducing the test application time and test data volume of NoC testing.

In [5], a test data transportation method with multiple data flits and a novel scan chain configuration method are presented which maximize the utilization of the NoC channel without adding too much hardware overhead.

Some of the proposed methodologies for NoC testing use different types of BISTs, e.g. the work described in [6]. They have presented a novel built-in self-test methodology for testing the inter-switch links of NoCs. They have used a high-level fault model which accounts for crosstalk effects due to inter-wire coupling are used.

However, use of standard DfT solutions for NoC results in a prohibitively large area overhead [7]. A non-scan test method for NoC switches is proposed in [8]. This method tests NoC switches for high level direction faults and needs much less test-mode hardware than what is required for most scan methods. A brief description of this method is presented in the following section.

3. Preliminaries

A brief description of the NoC architecture we have used in this paper, our proposed high level fault model, our test platform and some of the previously proposed test strategies are presented here.

3.1. Our NoC Architecture

In this paper we focus on regular 2-D NoC topologies using XY routing algorithm where a packet is first routed in the X direction and then in the Y direction before reaching the destination. Also, we assume that there are only two externally accessible switches in the mesh which we call Test Access Switches (TASs). One of the TASs is located at the lower left of the mesh (TAS1) and the other is located at the upper right of the mesh (TAS2). Processing elements connected to TASs can act as test packet generators and output packet analyzers in addition to their specific operations. Hence, they will be the test source and sink of the circuit. Switching is based on packet switching approach where a packet consists of several fields like source address, destination address, and payload.

Figure 1 shows the structure of the NoC switch architecture used in this paper. As shown in this figure each switch has five I/O ports, 4 ports for connecting each switch to adjacent switches and 1 port to connect the switch to its related processor.

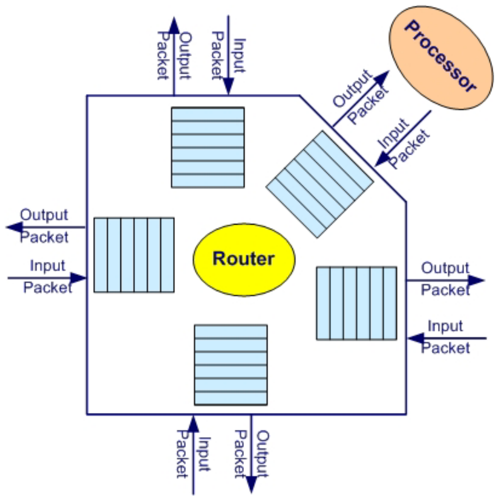

Figure 1. NoC Switch Architecture

3.2. High level Fault Model

In general, there are two categories of data faults and control faults in NoC switches. The data faults have to do with data contents of the packets (payload), and the control faults deal with faults in routing data in switches and between processing elements. We mostly focus on this kind of faults in our NoC switches.

One control fault could be that all packets in the input ports are sent to a specific output port; we call such a fault "stuck at port fault". Accordingly, 5 stuck-at direction faults can be identified:

1. Stuck-at East, *SaE*
2. Stuck-at South, *SaS*
3. Stuck-at West, *SaW*
4. Stuck-at North, *SaN*
5. Stuck-at Processor, *SaP*

When a Stuck-at East fault occurs all packets are sent to the East output port (Port 1) of the switch regardless of their destination fields. Similarly, other faults cause packets to be sent to their stuck-at ports. A stuck-at processor fault in a switch makes all the packets be sent to the related processor.

Other switch control faults such as packet dropping, lost-destination and misrouting can also be modeled with our suggested set of faults. However, these faults are beyond the scope of this paper. For example, consider the case when switch *i* corrupts the destination field of packets that it receives and sets this field to switch *j*. This fault in switch *i* can be modeled with a SaP fault in switch *j* [8].

3.3. Test Platform

In this paper, we use a high level VHDL based platform we have already developed [8] for fault simulation, fault coverage calculation and test time measurement. Our test platform consists of models of actual NoC switches and processing elements in the test mode.

Some modification has been done on the platform in order for it to support our proposed exhaustive test strategy. They are as follow:

- For the exhaustive test strategy, switches are capable of multicasting incoming packets in the test mode.
- Switches decrement hop count field of their outgoing packets in order for them to be discarded after some appropriate time to live (TTL); they will discard packets with hop count equal to zero.

3.4. Previously Proposed Test Methodologies

The previous related methods were based on processing element calculations on a received packet. A fault can be detected as a dropped packet or a packet with a wrong switch count. Different packet sending strategies with different roadmaps can be used to gain different fault coverages [8].

4. Exhaustive Test Strategy

The main idea of exhaustive test strategy is taken from a routing technique used in packet-switched networks called flooding. A brief description of flooding routing is presented here.

4.1. Flooding Routing Technique

Flooding routing requires no network information whatsoever, and works as follows. A packet is sent by a source node to every one of its neighbors. At each node, an incoming packet is retransmitted on all outgoing links except for the link on which it arrived. Unless something is done to stop the incessant retransmission of packets, the number of packets in circulation just from a single source packet grows without bound; a simple technique is to include a hop count field with each packet. The count can originally be set to some maximum value, such as the diameter (length of the longest minimum-hop path through the network) of the network. Each time a node passes on a packet, it decrements the count by one. When the count reaches zero, the packet is discarded.

The flooding technique has three remarkable properties:

1. All possible routes between source and destination are tried.
2. All nodes that are directly or indirectly connected to the source node are visited.
3. Because all routes are tried, at least one copy of the packet to arrive at the destination will have used a minimum-hop route.

Because of the first property, the flooding technique is highly robust [9].

4.2. Flooding as a Fault Tolerant Algorithm for NoC Interconnects

Flooding is an effective fault tolerant technique. If a path exists to the destination, a message will almost certainly arrive. Consequently, some variants on the simple flooding protocol have been proposed as fault tolerant communication algorithms. Dumitras et al. [10] propose a probabilistic flooding scheme based upon well known gossip techniques as a possible fault tolerant solution [11, 12]. In this algorithm, every time a message is generated it will be passed to all of its neighbors with probability *p*, and will be dropped with probability *1- p*. *p* must be carefully chosen to obtain near flooding performance without sending as many redundant copies of the message[13].

The probabilistic flooding algorithm is destination ignorant in that packets get routed irrespective of where they are in the network. This results in significant network resources being used to transmit the packet to areas of the chip where it will not be needed. The directed flooding algorithm makes use of an NoC's highly regular structure to direct a flood towards the destination. In this algorithm, the probability *p* of passing a message to each outgoing link is not fixed and varies based on the destination of the packet [13].

Although flood-based fault tolerant algorithms are highly efficient, it has shown that they have an exceedingly high communication overhead [13]. And, in practice, this level of fault tolerance may not be necessary. So, we assume that the NoC we are designing routes packets in xy fashion in its normal mode and we use it floods only in its test mode for fault detection.

4.3. Manhattan Distance

The Manhattan distance (also known as taxicab distance) between two points in a Euclidean space with fixed Cartesian coordinate system is the sum of the lengths of the projections of the line segment between the points onto the coordinate axes. For example, in the plane, the Manhattan distance between the point P_1 with coordinates (x_1, y_1) and the point P_2 at (x_2, y_2) is $|x_1 - x_2| + |y_1 - y_2|$.

For example, consider the mesh shown in Figure 2 Manhattan distance between the two black points in the mesh is $|7 - 1| + |7 - 1| = 12$. The red, blue, and yellow lines representing the Manhattan distance all have the same length, whereas the green line represents the Euclidian distance [14].

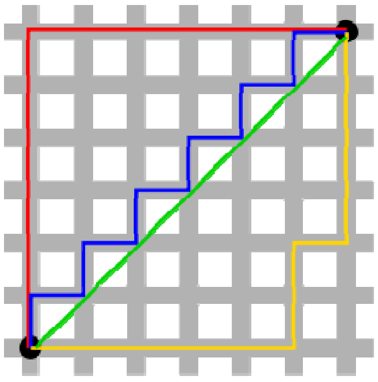

Figure 2. Manhattan vs. Euclidean Distance[14]

From Combinational Analysis we know that the total number of the paths with Manhattan distance between the two black points can be computed from the below formula:

$$K(m\,,\,n) = \frac{(m\text{-}1 + n\text{-}1)!}{(m\text{-}1)!(n\text{-}1)!}$$

where m and n are the number of the rows and columns of the mesh respectively [15]. It is obvious that if a packet in this mesh travels from a black point to another in xy or yx fashion, the total number of the hops it traverses is equal to the Manhattan distance between the two black points plus 1.

4.4. Test Mode Operation

Our proposed exhaustive test strategy works as follows: the processing element connected to TAS1 generates a packet and sets its destination field value to TAS2 address and sets its hop count field value to Manhattan distance between TAS1 and TAS2 and then floods it as discussed in Subsection 4.1. The same sequence of actions will take place in TAS2, but this time the generated packet will have a destination field value equal to TAS1 address.

Then, every switch in the mesh decrements the hop count field value of its incoming packets by 1 and floods them to its output ports. Packets arrived at each switch with hop count filed value equal to zero will be discarded. Also, packets arrived at each TAS with destination field value other than its address will be discarded.

This way, if no stuck at fault exists in the network, the total number of the packets arrived at each TAS will be equal to the total number of paths with Manhattan distance between the two TASs, i.e., *K(m,n)*. But if there is a stuck-at-fault in the network, the total number of the packets received at each TAS will be smaller than *K(m,n)*. For example, consider the situation in which switch TAS1 is stuck at East; this way, it will send the first generated packet only to its Eastern output port and consequently, some of the packets like those that travel in the red path shown in Figure 2 will be discarded in the middle of the path, because of traveling more hop counts than the Manhattan distance. Also, simulation results show that this method provides full fault coverage for the stuck at port faults described in previous sections.

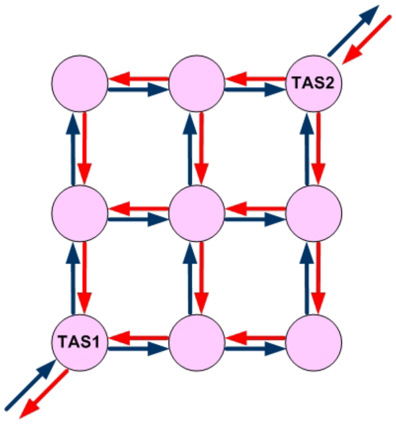

Figure 3. Propagation of Packets in NoC Test Mode

It should be mentioned that a test-mode controller is responsible for the operation of the network in the test mode. The controller configures the switches so that they will be able to do flooding and will also set a counter in the processing elements connected to TASs to count the total number of packets received at each TAS for the purpose of fault detection. Figure 3 shows the TASs and the propagation of packets in the NoC test mode. Note that each edge may be traveled by many packets.

The main advantage of the exhaustive test strategy is its simplicity in implementation which accordingly results in a small amount of test-mode hardware added to switches as well as its full fault coverage.

The principal disadvantage of exhaustive test strategy is the high traffic load that it generates as well as its high power consumption, which is directly proportional to the connectivity of the network. In fact, the main problem with this strategy is that some of the packets generated while flooding do not travel toward the test sink switch and accordingly their time to live will get zero before arriving at the sink and they will be discarded.

5. Experimental Results

The following subsections present statistics and formulas for the test time, total number of packets, power consumption, and hardware overhead added to NoC switches in test mode for exhaustive test strategy. We also make a comparison with the statistics collected for the previously presented strategies.

5.1. Fault Coverage

As mentioned before, the proposed method of this paper has 100% fault coverage. The other related works we presented in [8] were the *slow-train* and *fast-train* methods that have an average of 82% fault coverage. Depending on the road map selection and train selection, the fault coverages may differ [8].

5.2. Test Time

Test time of the exhaustive strategy is when the last test packet arrives at each TAS switch. Because of many packets that this scheme generates and switch wait-times, the timing of this method is undeterministic. However, the timing for a modified version of this method that we refer to as pseudo-exhaustive can be calculated as shown below.

$$T(m\,,\,n) = K(m,n) - 1 + (m + n - 1)$$
$$= \frac{(m-1+n-1)!}{(m-1)!(n-1)!} + m + n - 2$$

The graph shown in Figure 4 represents the test time of the exhaustive strategy for different NoC mesh sizes. As shown, the test time of the proposed method is relatively high compared to the previous method of packet trains. The worst road map of the packet trains method has a test time of 192 packet transitions for an 8x8 NoC. [8]

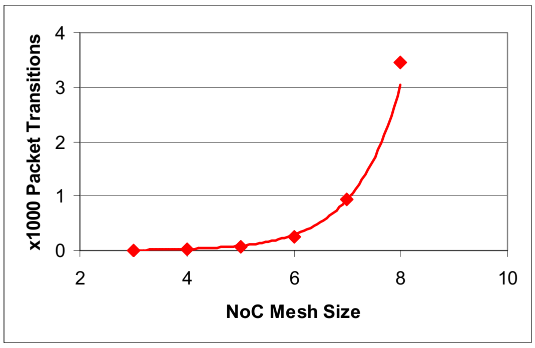

Figure 4: Test Time of the exhaustive Strategy

5.3. Number of Packets and Power Dissipation

The heavy traffic caused by multicasting of the packets in the switches leads to a huge number of packet switchings. Each packet is duplicated three times in each switch and, for large NoC structures, produce an astronomic number of packets. These dense packet switchings lead to high power dissipation in the test mode of the NoC. The packet trains method on the other hand have a relatively low number of packet switchings (an average of 100 packet switchings for an 8x8 NoC) and thus have a low power dissipation compared to the proposed method.

5.4. Hardware Overhead

The proposed method has a small amount of hardware overhead due to its simplicity. In the normal mode, the routing algorithm has to check a received packet's destination and make a decision and assign a proper output multiplexer to it. In the test mode, on the other hand, there is no decision to be made. Each packet is simply sent to all other ports except the incoming port. Upon receiving a packet, the three other multiplexers duplicate the incoming packet to enable multicasting. However, the amount of the hardware overhead remains the same for larger NoCs, thus making less overhead percentage for large NoCs.

6. Conclusions

This paper discussed a method of NoC switch testing that is based on network switching algorithms. In this method a small hardware is added to each switch. During the test time, this small hardware enables the switch to multicast the received packets. The packets propagate through the entire NoC and the fault is detected by counting the number of packets received in the TASs.

The proposed test method was used in a high-level VHDL test platform for NoCs of different sizes using a high-level fault model. We achieved 100% fault coverage with a negligible hardware overhead and a reasonable test time.

References

[1] J. Raik, V. Govind and R. Ubar, "An External Test Approach for Network-on-a-Chip Switches," *Proc. Asian Test Symposium (ATS)*, pp. 437-442 , 2006.

[2] E. Cota, C. Zeferino, M. Kreutz, L. Carro, M. Lubaszewski, and A. Susin, "The Impact of NoC Reuse on the Testing of Core-based Systems," *Proc. VLSI Test Symposium (VTS),* pp. 128-133, 2003.

[3] M. Hosseinabadi, A. Banaiyan, M. N. Bojnordi, and Z. Navabi, "A Concurrent Testing Method for NoC Switches," *Proc. Design Automation and Test in Europe(DATE)*, pp. 1171-1176, 2006.

[4] M. Hosseinabadi, A. Dalirsani, and Z. Navabi, "Using the Inter- and Intra-Switch Regularity in NoC Switch Testing," *Proc. Design Automation and Test in Europe(DATE),* pp. 361-366, 2007.

[5] Ming Li, Wen-Ben Jone, Qing-An Zeng, "An Efficient Wrapper Scan Chain Configuration Method for Network-on-Chip Testing," *Proc. Emerging VLSI Technologies and Architectures (ISVLSI'06),* 2006.

[6] Cristian Grecu, Partha Pande, André Ivanov, Res Saleh, "BIST for Network-on-Chip Interconnect Infrastructures", *Proc. VLSI Test Symposium (VTS'06)*, 2006.

[7] A. M. Amory, E. Brião, É. Cota, M. ubaszewski, F.G. Moraes, "A Scalable Test Strategie for Networkon-Chip Routers," *Proc. International Test conference (ITC)*, 2005.

[8] M. Sedghi, A. Alaghi, E. Koopahi, and Z. Navabi, "An HDL-Based Platform for High Level NoC Switch Testing," To appear in *Proc. Asian Test Symposium (ATS),* 2007.

[9] W. Stallings, *Data and Computer Communications*, Prentice Hall, 1996.

[10] T. Dumitras, S. Kerner, and R. Marculescu, "Towards onchip fault-tolerant communication," *Proc. Asia and South Pacific Design Automation Conference (ASPDAC)*, pp. 225-232, 2003.

[11] S. M. Hedetniemi, T. Hedetniemi, and A. L. Liestman, A survey of gossiping and broadcasting in communication networks, *NETWORKS*, 18:319–349, 1988.

[12] D.W. Krumme, G. Cybenko, and K. N. Venkataraman, Gossiping in minimal time, *SIAM J. Comput.*, 21(1):111–139, 1992.

[13] M. Pirretti, G. M. Link, R. R. Brooks, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, "Fault tolerant algorithms for network-on-chip interconnect," *Proc. International Symposium on Very Large Scale Integration (ISVLSI),* pp. 46-51, 2004.

[14] http://en.wikipedia.org/wiki/Manhattan_ distance.

[15] R. P. Grimaldi, *Discrete and Combinatorial Mathematics: An Applied Introduction*, Addison Wesley, 1998.