2012 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing

A New Fault Injection Approach for Testing Network-on-Chips

Luca Sterpone, Davide Sabena, Matteo Sonza Reorda
CAD Group – Dipartimento di Automatica e Informatica
luca.sterpone@polito.it

Abstract

Packet-based on-chip interconnection networks, or Network-on-Chips (NoCs) are progressively replacing global on-chip interconnections in Multi-processor System-on-Chips (MP-SoCs) thanks to better performances and lower power consumption. However, modern generations of MP-SoCs have an increasing sensitivity to faults due to the progressive shrinking technology. Consequently, in order to evaluate the fault sensitivity in NoC architectures, there is the need of accurate test solution which allows to evaluate the fault tolerance capability of NoCs. This paper presents an innovative test architecture based on a dual-processor system which is able to extensively test mesh based NoCs. The proposed solution improves previously developed methods since it is based on a NoC physical implementation which allows to investigate the effects induced by several kind of faults thanks to the execution of on-line fault injection within all the network interface and router resources during NoC run-time operations. The solution has been physically implemented on an FPGA platform using a NoC emulation model adopting standard communication protocols. The obtained results demonstrated the effectiveness of the developed solution in term of testability and diagnostic capabilities and make our solutions suitable for testing large scale NoC design.

1. Introduction

Today, the incessant reduction of the feature size of digital Very Large Scale Integration (VLSI) circuits allows integrating several hundreds of processing elements such as computational cores on a single chip. These resources are increasing demanding high performance communication; therefore traditional solution provided by on-chip buses can no longer sustain this demand.

In order to improve the performances and structural weaknesses of traditional buses, Networks on Chip (NoCs) are being adopted.

Nevertheless, NoCs are characterized by high performances and low power consumption; one of the open problems that afflict research activities on NoCs is the evaluation of their fault tolerance capabilities. Indeed, as specified in [1], faults are increasing happening and a very large set of effects must be considered in new generation of NoCs. The larger occurrence of fault is mostly due to the technology scaling of the new chip generations that result more susceptible to fault appearance induced by different phenomena such as single-event upsets, cross-talk, age-related degradation or process variability.

Generally, test method of NoC infrastructures address two issues: testing the switch blocks and testing the interconnection segments layout including the logic routers logic resources. Different testing techniques have been proposed in order to evaluate fault effects on NoC. Several NoC test methods have been focused on testing of the functional IP cores using Test Access Mechanism (TAM) [3]. Other authors assumed specific fault model for NoC fabric and subsequently adopting it to test the data transportation and the functional blocks [4]. Vice versa, dedicated TAM based on specific on-chip network is adopted by functional test solutions on SoCs multi-cores [5].

The solution of our work improves previously proposed method by developing a flexible and accurate fault injection environment, which can be adopted in order to evaluate the fault tolerance capability of different NoC architectures. The main advantage of the proposed solution rely on the possibility to apply different fault models that can emulate the effective faults affecting NoC architectures, besides the proposed solutions has a full controllability and observability of the NoC under test, since interconnections values and routers functional behavior can be directly observed during the test operations, feature which is extremely reduced for test solutions applied directly to the manufactured chip, since NoC interconnections are deeply embedded and spread

978-0-7695-4633-9/12 $26.00 © 2012 IEEE
DOI 10.1109/PDP.2012.82

across the chip, therefore adding of probe interconnections results inapplicable.

The implementation of our solution relies on the main idea illustrated in Figure 1. The fault injection method we developed is innovative since it is based on single reconfigurable chip, such a Static RAM-based Field Programmable Gate Array (SRAM-based FPGAs) where thanks to a suitable architecture, faults can be injected and evaluated. As illustrated in Figure 1, the architecture consists of two processors: the processor 1 is devoted to the application of the test pattern to the NoC under test while the processor 2 performs the injection of the faults. The execution of the fault injection does not require the insertion of intrusive module into the NoC architecture, since modifying the configuration memory bit of the FPGA device thus physically inserting the desired fault performs the injection. This operation is performed thanks to the availability of the Configuration Access Port which is located internally to the device and can be controlled by a logic core; in our case by the processor 2 [6]. Thanks to our solution the fault injection into the NoC architecture can be performed without intrusive modules affecting the real behavior and with optimal performances, since the working frequency of the NoC is not drastically degraded by the fault insertion.

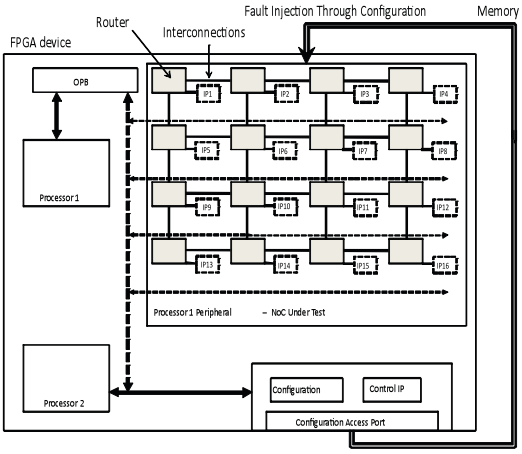

Figure 1. The main architectural scheme of the proposed approach.

The rest of this paper is organized as follows. In Section II we present the main background concepts behind the implementation and testing of NoC. The description of the fault injection method, the developed architecture and the NoC fault models are presented in Section III. Section IV describes the testing routines developed for testing the router logic blocks and the NoC switches and interconnections. Fault injection results performed on a real NoC case study are presented in Section V. Finally, conclusions and future works are drawn in Section VI.

2. Background

A Network on Chip is an interconnection architecture consisting of a 2-D mesh of routers each of which connected to a set of interconnection resources. An example of NoC structure is illustrated by the NoC Under Test box in Figure 1. Each router may be connected to its four neighboring routers and to a set of other routers not located on its same row and column. Various NoC architectures have been proposed basing on the 2-D matrix of resources [7]. All of them are characterized by several aspects including topology, routing algorithms, switching and flow control mechanisms. Several NoC architectures have been proposed in the past. The typical NoC design is based on the data exchange between the functional IP cores in the form of data packets. Depending on the adopted communication scheme, data packets are transmitted through routers and interconnections from the source to the destination IP port. Different schemes have been proposed for NoC communication such as communication switching, virtual cut-through and wormhole switching [8]. While communication switching is more applicable to small-size networks, virtual cut-through and wormhole switching are appropriate for large MPSoCs since data have to traverse large distances and packets are buffered by halfway routers on the routing path from the source to the destination. NoC architecture embeds two main resources: interconnections and routers. Interconnections are wire segments that link the various routers inside to the NoC array architecture, while routers are the most complex part of the NoC architecture. The details of a generic router are illustrated in the following section.

2.1. Overview on the router architecture

The architecture of a NoC router, which an example is illustrated in Figure 2, consists of a set of FIFO buffers forming the input and output data ports, a crossbar switch controlled by a logic circuitry that allow to implement the transportation methodology.

A router generally consists of multiple buffer stages connected through the input and output ports to the routers placed on the same row (SR), column (SC) or on other locations (OL) of the considered router [9][10]. Each input/output buffer stage form an input/output queue which is internally connected to a crossbar switch. The crossbar switch is the core of the router since it allows to form the links between the input and output queues in all the possible combinations. The connection between the input and

output queues are managed by an arbitration circuitry, which on the basis of the adopted communication protocol routes the data packets.

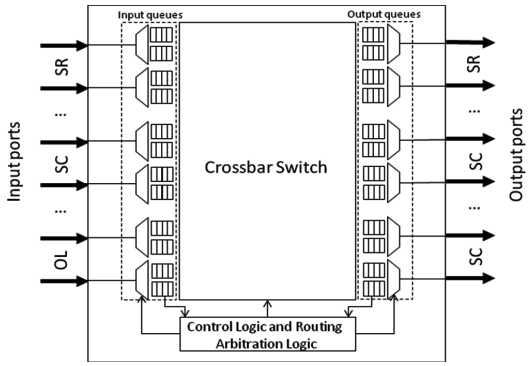

Figure 2. NoC router scheme including input and output registers, crossbar switch and routing and arbitration logic.

3. The proposed fault injection method

The environment of the proposed method consists of an host-PC which communicate to the FPGA board through a JTAG connection and a RS-232 serial cable. The method consists of the preliminary generation of the fault list performed on the host-PC by parsing the NoC netlist and creating all the possible fault locations according to the used synthesis model. The fault locations are consequently transferred through the JTAG cable to the processor 2 dedicated memory. The fault injection execution flow is illustrated in Figure 3. It consists on the following principal modules:

1. *Fault Application*: Fault Application: the fault application module is executed by the processor 2 and consists on all the operations dedicated to the insertion of a fault into the NoC architecture. At first a fault location is read from the memory module, secondly, the fault location is converted into the correspondent FPGA's configuration memory coordinates that control the selected NoC resource. Finally, the processor 2 accesses to the FPGA's internal configuration access port facility and activates the reconfiguration of the configuration memory bits related to the selected coordinates. During the modification of the FPGA's configuration memory, the NoC functionality is temporary freeze. This has not any impact on performance degradation of the NoC, since the freezing is synchronized with the microprocessor control clock.
2. *Configuration memory coordinates*: The configuration memory coordinates are the physical identification of the FPGA's configuration memory bits. Each configuration memory coordinate is associated to a correspondent sequence of FPGA's configuration memory bits.
3. *FPGA configuration memory*: The FPGA's configuration memory contains million of SRAM cells that configure the behavior of the circuit mapped on the FPGA device. The main idea of the present work consists in acting on the configuration memory cells related to the behavior of the NoC architecture in order to insert a fault. Therefore, in order to perform the right fault injection it is necessary to know the exact correspondence between each configuration memory bit and the controlled FPGA's resource [15]. The modification of the configuration memory is performed by the Internal Configuration Access Port (ICAP) controlled by the software running on the Processor 2. Through the ICAP port it is possible to modify selectively a single FPGA configuration frame in a small amount of time (dependent from the kind of FPGA device adopted).
4. *Patterns generator*: The patterns generator aims at generating the test stimuli applied to the NoC. The stimuli are generated in the form of data packets transmitted from a source to a destination IP considering a NoC using a Wormhole transport methodology. The patterns are generated by the software running on the Processor 1. Each pattern consists of a signature composed by the identification of the source and destination IPs and by the complete data packet. The application of the test patterns is executed by the Processor 2 by executing the following steps: at first the signature is decoded individuating the source IP; secondly the Processor 2 access through the On-Chip Peripheral Bus (OPB) to the source IP and send the data packet; third, the signature is decoded individuating the destination IP and finally, the Processor 2 reads through the OPB bus the data packet received by all the IP (except to the source one).
5. *Fault classification*: The fault classification is executed by the Processor 2. The first action consists on capturing all the outputs of the IP destination ports. Secondly, the pattern signature is decoded individuating the destination IP.
6. *Network-on-Chip Under Test*: It consists on the architecture of the NoC under evaluation. The NoC model should be described in HDL language in order to facilitate the integration in our environment. The integration into the testing environment is performed by linking each NoC's port to the platform IP port.

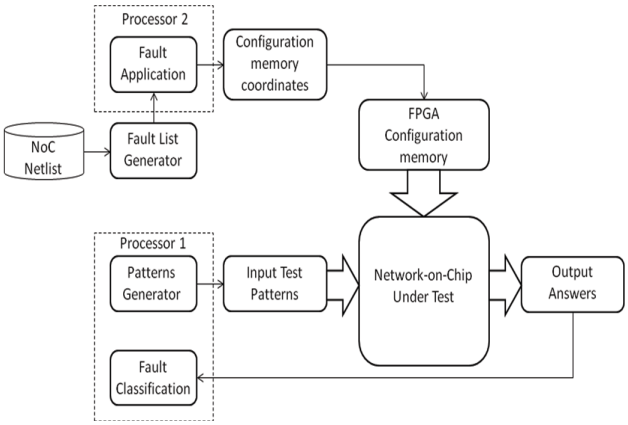

Figure 3. The fault injection execution flow of the proposed method.

4. Routing and logic fault model

The developed environment allows to inject a faults into a NoC architecture by acting through the FPGA's configuration memory. Principally, when a fault occurs into a NoC architecture, it may affect routing or logic resources. A fault into routing resources may interest the hardwired interconnections between routers or the programmable wires internally to each router's crossbar switch. Conversely, a fault into logic resource may essentially affect logic within router elements such as Flip-Flops (FFs) belonging to input / output queues or the circuitry of the control and arbitration logic modules.

4.1. The routing fault model

When a fault occurs into interconnection router resources, it has a probability to affect one or more wires for a duration of some times units (typically clock or transfer cycles) with different possible effects.

When a faults affects an interconnection resource it may create two possible effects: a transient modification of the logic value physically transferred by the wire, a permanent modification of the wire behavior. Since transient effects, such as pulses (i.e. 0-1-0 or 1-0-1) becomes critical only if the propagation of the pulse is captured by FF after the wire, we modeled such effect into the logic fault model. Vice versa, the permanent modification of the routing may introduce a physical change on the routing behavior.

In order to model, all the possible effects into the routing resources, either belonging to interconnections or internal router wires, we defined five different effects depicted in Table 1, where each effect description is associated to the correspondent modification of the FPGA resources necessary to physically implement the injection of the fault within the NoC model.

Type	Effect Description	FPGA modeling
-	Original wire segments	
A	Logic value on wires is inverted	Insertion of a inverter gate through the original wire segments sequence
B	Wire forced to logic 0	Insertion of a ground connection through the original wire segments sequence
C	Wire forced to logic 1	VCC Insertion of a VCC connection through the original wire segments sequence
D	Wires A forced to value of wire B	A B Two wire segments of the nets A and B are opened. A new wire segments is added between A and B.
E	Wire forced to a propagation Delay D	A sequence of wire segments are added to the original wire. The number of wire segments is proportional to the delay D.

Table 1. Routing Fault Effect Description and the Corresponding FPGA Fault Injection Model.

Considering the original wire segment configuration illustrated in Table I and implemented by the FPGA routing resources using a sequence of Programmable Interconnection Points (PIPs), a routing fault may provoke:

A. The inversion of the propagated logic value in case a fault modifies the electrical behavior of the affected routing wires. Typically this effects appear into interconnections between NoC routers;

B/C. The physical rupture of a routing segment is modeled as an open effect on routing segment. Depending on the position of the rupture, a break into a routing segment may create a permanent propagation of logic value 0/1. This effect may principally affect routing resources within the router crossbar switch due to the permanent rupture of a given configuration.

D. The modification of multiple demultiplexer configurations is modeled by a bridge effect. This effect may principally affect routing resources within the router crossbar switch due to permanent rupture of two demultiplexer configurations where a configuration interferers to the other one.

E. The modification of the propagation delay of a routing resource is modeled by the physical modification of the number of routing PIPs in order to insert the selected delay fault.

The physical modification of the FPGA resource in order to insert the previously described fault model is performed through the partial configuration of the FPGA configuration memory which is described in the injection process section.

4.3. The injection process

The injection process is performed by a synchronized cooperation between the injection of a fault performed by the processor 2 and the application of the test patterns executed by the processor 1 this operation is explicated in the following six steps:

1. The processor 2 reads the fault location from the previously generated fault list.
2. The processor 2 configures the internal register of the FPGA's ICAP module in order to apply the re-configuration of the configuration memory cells according to the location and type of fault. The selection of the configuration memory cells is read from an appropriate DB containing the configuration memory bit and resource correspondence and the physical locations of NoC routing and logic resources.
3. Once the ICAP module is configured, the processor 2 starts the modification of the configuration memory.
4. In parallel to the steps 2 and 3, the processor 1 generates the test patterns. We adopted a random generation either for the source/destination locations of the packets and for the content of the data packet. The whole test patterns are stored into a on-board DDRAM memory available on the FPGA carry board.
5. When the processor 2 has terminated the reconfiguration process, the processor 1 starts the application of the test patterns. IP output ports are read at each clock cycle.
6. At the end of the application of all the test patterns, the processor 2 classify the faults according to the previously described fault classification.

These steps are repeated for all the faults available in the fault list. During the execution of the fault injection process the processor 2 communicates to the host-PC by means of the serial connection providing information about the status of the test, and finally providing the final result of the test.

5. Experimental results

We implemented the described approach on a Xilinx Development Board equipped with a Spartan 6 XC6SLX45T SRAM-based FPGA. The environment has mapped using two Microblaze processors communicating by means of on-chip processor local bus and connected to the NoC under test by means of on-chip peripheral bus. Each IP input / output port has been implemented with software accessible registers, by this way the software mapped on the processor 1 may rapidly access to the NoC ports.

5.1. Case study: NoCem

In this work, we rely on the Network on Chip emulator (NoCem) which consists of an open source HDL model of a Network on Chip [17]. NoCem allows different parametric regulations in terms of network configurations, buffering schemes and arbitration logic. The NoCem model is targeted Xilinx-based SRAM-FPGAs and it uses different generics and generate statement in order to create a complete NoC design. For the purpose of this work we configured the NoCem model using NoC with 32 bits dataword size, packet length of 16 datawords, router FIFO buffer length of 8, a square grid configuration with 16 IPs which correspond to a 4x4 2D mesh. The FPGA resource utilization in terms of LUT and FF of the NoC under test and of the developed environment are reported in the Table II.

	Routing Slices [#]	FF Slices [#]	LUT Slices [#]
NoCem	2,612	4,236	7,369
Test Environment	1,790	12,484	13,210

Table 2. NoC emulator implementation details

5.2. Fault Injection Results

We run four different fault injection campaign, in order to test inverted wire faults, stuck-at 0/1 faults, bridge faults and delay faults. The obtained results are illustrated in Table III where we reported the number of Injected and Detected Faults and the fault injection time required by the developed platform.

The average time required by the developed platform in order to perform the physical modification of the FPGA configuration memory in order to inject a wire, stuck-at or bridge faults is of 12 µs. While it increases to 18 µs in case of the delay faults due to the more configuration steps required for adding extra PIPs. The great part of the fault injection time is due to the random patterns application which on the average requires around 73 ms for each injected fault.

	Injected Faults [#]	Detected Faults [#]	Injection Time [min]
Wire Faults	2,834	2,780	3.45
Stuck-at	278,430	261,278	338.8
Bridge	6,829	6,740	8.31
Delay	279,834	143,439	340.6

Table 3. Fault injection results.

In order to compare the effectiveness of the proposed approach with consolidated techniques, we compared the fault injection experiments with the

results obtained by means of fault simulation of stuck-at faults using the Tetramax® fault simulation tool applying the same test patterns. The results reports exactly the same data obtained with the developed Stuck-at fault injection but the time required by the simulation has been of around 13 hours with respect to the 5.6 hours required by the developed approach. Please note that the detected delay faults are extremely low due to the very low coverage of the random test patterns.

6. Conclusions and future works

In this paper we present a novel test approach based on a dual-processor system implemented on SRAM-based FPGA which is able to test mesh-based Network On Chips. The approach has several advantages: flexibility, observability and test speed. The approach has been implemented on an FPGA platform and a real NoC architecture using standard communication protocol has been tested. The experimental results demonstrated the effectiveness of the developed approach and makes our approach suitable for testing large scale NoC design. As future activities, we plan to extend our emulation platform to analyze the effects of delay faults and to improve the delay fault model.

References

[1] S. Borkar, "Designing reliable systems from unreliable components: the challenges of transistor variability and degradation", IEEE Micro, Vol. 25, No. 6, pp. 10-16, November 2005.

[2] L. Chunsheng, V. Iyengar, S. Jiangfan, E. Cota, "Power-Aware Test Scheduling in Network-on-Chip Using Variable-Rate On Chip Clocking", IEEE International Very Large Scale of Integration Test Symposium , 2005, pp. 349 – 354

[3] B. Vermeulen, J. Dielissen, K. Goossens, C. Ciordas, "Bringing communications networks on a chip: test and verification implications", IEEE Communications Magazine, Volume 41, Septmber, 2003, pp. 74 – 81.

[4] A. Kohler, G. Schley, M. Radetzki, "Fault Tolerant Network on Chip Switching With Graceful Performance Degradation", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 29, No. 6, June 2010, pp. 883 – 896.

[5] 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.

[6] Xilinx Product Specification, "LogiCORE IP XPS HIWCAP", Product Spec., DS586, June 22, 2011.

[7] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip", ACM Computing Surveys, Vol. 38, pp. 1-51, March, 2006.

[8] J. Duato, S. Yalamanchili, L. Ni, "Interconnection Networks – An Engineering Approach", Morgan Kaufmann, 2002.

[9] C. Grecu, P. Pande, A. Ivanov, R. Saleh, "Timing Analysis of Network-on-chip Architectures for MP-SoC Platforms", Elsevier Microelectronics Journa, XX

[10] 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.

[11] S. Pasricha, Y. Zou, D. Connors, H. J. Siegel, "OE+IOE: A novel turn model based fault tolerant routing scheme for networks-on-chip", IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, 2010, pp. 85 – 93

[12] H. Zimmer, A. Jantsch, "A fault model notation and error-control scheme for switch-to-switch buses in a network-on-chip", IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, 2003, pp. 188 – 193.

[13] E. Cota,"Power aware NoC Reuse on the Testing of Core-Based Systems", proceedings of the IEEE International Test Conference 2003, pp. 612 – 621.

[14] Yung-Chang Chang, Ching-Te Chiu, Shih-Yin Lin, Chung-Kai Liu, "On the Design and Analysis of Fault Tolerant NoC Architecture Usin Spare Routers", International DAC Conference, 2011, pp. 431 – 436.

[15] L.Sterpone, M. Violante, "A new analytical approach to estimate the effects of SEUs in TMR architectures implemented through SRAM-based FPGAs", IEEE Transactions on Nuclear Science, Vol. 52, Issue 6, 2005, pp. 2217 - 2223.

[16] A. Perreira, J. P. Teixeir and M. Santos, "A Novel Approach to FPGA-based Hardware Fault Modelling and Simulation", IEEE International Workshop on Design and Diagnostic of Electronic Circuits and Systems, 2003, pp. 17 – 24.

[17] Reference to the Network-on-Chip emulator on www.opencores.org