# Crosstalk fault tolerant NoC - design and evaluation

Alzemiro H. Lucas, Fernando G. Moraes
Faculdade de Informática - Pontifícia Universidade Católica do Rio Grande do Sul, PUCRS - Porto Alegre, Brazil
{alzemiro.silva, fernando.moraes}@pucrs.br

*Abstract*—**Networks-on-chip (NoCs) interconnection infrastructure was proposed to maximize the integration level of multiple processing elements in the same chip. New applications require a high performance communication infrastructure with low data latency. The innovations on integrated circuit fabrics is continuously reducing components size, which increases the logic density of systems-on-chip (SoC), but also affect the reliability of these components. This work presents an implementation of an error recovery technique for networks-on-chip with the objective to protect network links against crosstalk effects, minimizing the latency of error recovery, with minimal area overhead. Results of the proposed technique are illustrated by simulating the network with crosstalk fault simulation, measuring latency and area overhead.**

***Keywords-Networks-on-Chip (NoCs); fault tolerance; reliability (key words)***

## I. INTRODUCTION

NoCs has emerged as a candidate solution to interconnect IPs in complex SoCs, due to its scalability and parallelism, compared to bus architectures. A NoC can be defined as a set of routers responsible to transmit data on the intra-chip domain, exploiting methods used in general networks, decoupling communication from computation, enabling the creation of protocols to grant reliability and quality of service.

Besides the communication infrastructure, an important challenge on the SoC design is the degradation of the signal integrity on long wires. Coupling capacitances tends to increase with the reduced components size. Faster clocks and lower operation voltage makes the delay caused by crosstalk effects even more critical, being the major source of errors in nanoscale technologies [1]. Another noise sources that can produce data errors [2] are electromagnetic interference, radiation-induced charge injection and source noise.

NoC utilization makes easier the implementation of fault tolerance techniques for intra-chip communication. Those techniques include adaptive routing algorithms and codification for error detection and correction. Other techniques proposed to avoid the coupling capacitance effect includes place and route techniques to avoid routing of the bus lines in parallel, changes in the geometrical shape of bus lines and adding a shielding line between two adjacent signal lines. However, those techniques require advanced knowledge on electric layout design, and they are executed later in the design flow. Considering these issues, bus encoding techniques represents a good tradeoff between implementation costs and design time to minimize crosstalk effects [3], and it is a technology independent mechanism to increase reliability on intra-chip communication. Even though, designers should carefully choose the codification technique, taking into account that the ideal codification should have minimal area overhead while delivering the desired reliability [4], due to strict performance and power constraints of NoC architectures.

In this paper we present the implementation of an error recovery mechanism *at the flit level*, to increase the NoC reliability, making it resilient against crosstalk data faults. The proposed technique has low area overhead, high fault coverage and minimum latency on error recovery.

The rest of this paper is organized as follows. Section II presents related works concerning reliable communication on NoCs. Section III presents the design of the crosstalk fault tolerant NoC. Section IV reports the fault modeling used to simulate and validate the network. The results obtained with the implemented architecture are presented on section V. Finally, Section VI concludes the paper.

## II. RELATED WORK

Marculescu [5] introduced the concept of stochastic communication for NoC architectures. The principle of this method is to use a probabilistic broadcast algorithm to spread a coded message into the network until this message reaches the receiver IP. Therefore, if an error occurs in one of the copies traveling in the network, this copy can simply be discarded, since at least one correct copy will reach the receiver IP. This algorithm ensures the correct delivery of messages, but generates excessive traffic in the network.

Zimmer [6] proposes a fault model notation, where faults can occurs simultaneously in multiple wires, during multiple clock cycles. This fault model is used to evaluate coding techniques on NoC buses. The goal of this work is to present an accurate model to simulate the occurrence of faults in wide data bit busses and to show the reduction of residual faults when the bus is protected with coding techniques using single error correction and a double error detection (two bits coverage).

Bertozzi [7] presents a NoC architecture with pipelined links, pipelined arbitration and switching within the routers. The goal of this architecture is to provide high-speed operation and reliable communication. To achieve

reliable communication, this architecture provides error control using CRC codes on links.

Vellanki [1] addresses a NoC architecture with Quality of Service (QoS) and error control techniques. This paper presents the evaluation of the proposed architecture, taking into account performance and power dissipation. The two QoS methods addressed in this work are guaranteed throughput and best effort, and the error control techniques include single error detection and retransmission, and single error correction.

Murali [4] explores different error recovery methods for NoCs, evaluating for each one the energy, error protection, and performance impact. The methods comprehend end-to-end error detection, router-to-router error detection, at either flit-level and packet-level, and a hybrid scheme with correction of single errors and detection of multiple errors. This paper has shown that the packet storage is the main concern in energy consumption, thus the end-to-end error detection and retransmission has the higher cost in energy efficiency. Using the hybrid scheme, it is possible to reduce the energy overhead at lower error rates. Small errors are corrected and retransmissions are avoided. However, in the presence of multiple errors its efficiency is lower than error detection and retransmission schemes.

Pullini [8] evaluated the impact of fault tolerance in flow control schemes for NoCs, presenting three alternative schemes (STALL/GO, T-Error and ACK/NACK). Pullini concludes that the most adequate flow control protocol for error recovery is ACK/NACK, even considering its higher energy consumption and its smaller latency overhead in absence of faults.

Muralli [9] presents an alternative routing algorithm for NoC architectures with in-order packet delivery and fault tolerance guarantees. This algorithm includes multipath routing without the need of reorder buffers, saving energy. The strategy presented in this paper explores spatial and temporal redundancies to tolerate transients and possible permanent faults on NoC links.

Zhu [10] shows comparisons between fault tolerant adaptive routing algorithms for networks-on-chip. The analyzed metrics are throughput, latency, energy consumption, and successful arrival probability. The algorithms were compared with XY routing to evaluate area and energy overhead. The conclusion shows that the *negative first* algorithm has the smaller overhead compared to XY and can provide tolerance against permanent faults, presenting the best results over the analyzed algorithms.

Grecu [11] proposed new metrics for performance evaluation of fault tolerant NoC architectures. The metrics proposed in this paper includes detection latency, recovery latency and message arrival probability. These metrics were analyzed in a simple simulation scenario taking into account retransmission with and without priority. The author states that these metrics can better estimate the quality of a particular fault tolerant implementation than just performance metrics such as latency, throughput, and power consumption.

The *contribution* of this work is the development, validation, and analysis of an error detection and recovery technique for NoCs. The Hermes NoC [12] is chosen as the reference design, since it is an open-source NoC, validated by simulation and prototyping (FPGA).

## III. CROSSTALK FAULT TOLERANT NOC

Error detection and recovery circuitry should have a minimal impact on the area, power consumption, and latency. A router-to-router flit-level error detection and retransmission method is adopted. Such method has the following advantages:

- It is not necessary to store full packets at each router, enabling the use of wormhole packet switching, reducing area, power and latency compared to store-and-forward or virtual cut- through;
- This method is faster compared to full packet retransmissions, since once an error is detected, the flit can be retransmitted in the next clock cycle;
- Error detection occurs before routing decision, making the network resilient against misrouting due to header flit errors.
- Flits to be retransmitted are available at the routers buffers, inducing smaller area overhead.

CRC codification is adopted for error detection, due its small complexity on logic implementation, and small parity delay calculation when using a parallel architecture [13] to generate *N* parity bits per clock cycle.

### *A. Reference NoC*

HERMES is a parameterizable infrastructure, specified in VHDL at RT level, aiming to implement low area overhead packet switching NoCs. Routers have centralized switching control logic and up to five bi-directional ports. One port is used to establish the communication between a router and its local processing element, while the others are connected to the neighbor routers. Each input port stores received data on a FIFO buffer. The routing algorithm chosen for this work is the XY. A centralized round-robin arbitration grants access to incoming packets. If the incoming packet request is granted by the arbiter, the XY routing algorithm is executed to connect the input port to the correct output port. If the algorithm returns a busy output port, the header flit and all subsequent flits of this packet are blocked. After all flits in a packet are transmitted, the port is released.

The basic NoC architecture to implement the error detection and recovery circuitry is a 8*x*8 2-D mesh network, flit size equal to 16 bits, 8-flit buffer depth, without virtual channels.

*B. CRC Circuit*

The CRC circuit must encode 16 bits per clock cycle, since the flit size is equal to 16 bits. Four parity bits are generated, to ensure error coverage in multiple bits. Using this approach, 93.75% of all possible 16-bit error patterns can be detected, and thus recovered.

The polynomial generator is $g = 1+X+X^4$. The rest of the division between the input word and the polynomial generator generates the four parity bits of each word. The circuit shown on Figure 1. executes this operation in 16 clock cycles, because it can only receive 1 bit per clock cycle. The four parity bits are stored in each flip-flop (FF) at the end of the operation.

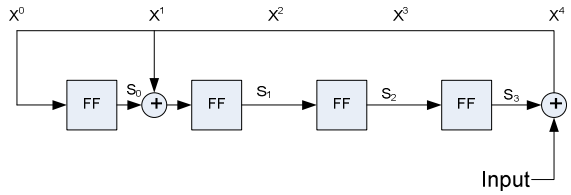


Figure 1. CRC encoder for g = 1+X+X4.

This basic circuit has small area, but it is too slow. To enable the codification of one flit per clock cycle, a parallel encoder is required. The circuit illustrated in Figure 2. enables the parallel CRC computation. This circuit is composed by connecting 16 combinational blocks with 4 inputs and 4 outputs, generating the 4 parity bits in parallel.

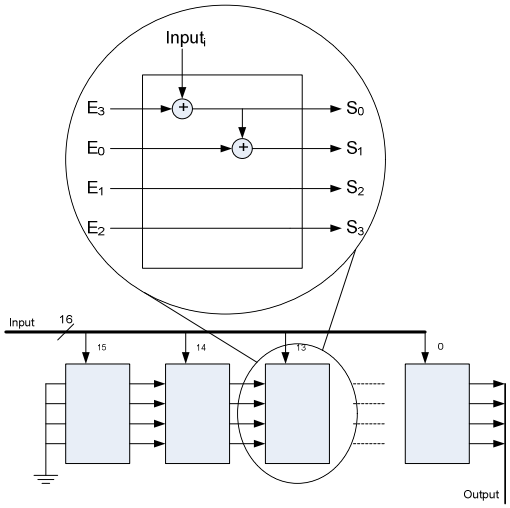


Figure 2. Parallel CRC Encoder.

This second CRC circuit has a long critical path, reducing the final clock frequency. Simplifying the logic equations of the circuit presented in Figure 2. , it is possible to obtain a logic equation for each parity bit, with minimal propagation delay:

$$S_0 = E_{15} \oplus E_{11} \oplus E_8 \oplus E_7 \oplus E_5 \oplus E_3 \oplus E_2 \oplus E_1 \oplus E_0$$
$$S_1 = E_{12} \oplus E_9 \oplus E_8 \oplus E_6 \oplus E_4 \oplus E_3 \oplus E_2 \oplus E_1$$
$$S_2 = E_{13} \oplus E_{10} \oplus E_9 \oplus E_7 \oplus E_5 \oplus E_4 \oplus E_3 \oplus E_2$$
$$S_3 = E_{15} \oplus E_{14} \oplus E_{10} \oplus E_7 \oplus E_6 \oplus E_4 \oplus E_2 \oplus E_1 \oplus E_0$$

In this way, the CRC encoder used in this work contains 4 XOR gates, with 8 or 9 inputs (the critical paths contains two logic levels). The decoder uses the same circuit of the encoder and compares the received parity bits with the bits generated locally by this encoder. If the values of received parity bits are different from the calculated bits, the decoder signals an error.

*C. Router architecture*

The basic router architecture is modified to support error detection and recovery. The first modification concerns the external interface between routers. Figure 3. shows the interface between input and output ports of two routers, highlighting the added signals for error recovery. The interface between two routers is bidirectional, being represented in the Figure only one of such directions.

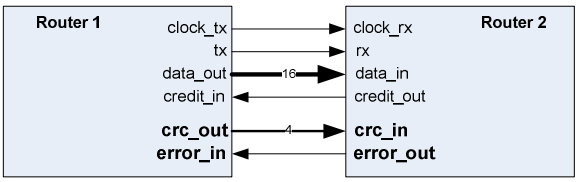


Figure 3. Interface between input and output ports.

The signal *clock_tx* synchronizes the data transmitted in *data_out* bus, and the signal *tx* indicates the validity of data. While the receiving router has free positions in its input buffer to consume new data, the signal *credit_out* remains high. CRC is computed over *data_out* and transmitted through the *crc_out* signal. The receiver router computes the CRC over the incoming data, and checks if the parity received in *crc_in* corresponds to the calculated parity. If an error is detected, the receiver signalizes the presence of an error, setting the signal *error_out*.

As the local port busses are not long enough to be susceptible to crosstalk effects as the network links between routers, the local port does not contain the signals *crc_out* and *error_out*.

To avoid corrupted data to be written on the input port buffer of the receiver router, the CRC decoder generates and *error* signal, disabling the writing of *data_in* into the buffer. In such a way, if an error occurs in the header flit, this process avoids erroneous routing and arbitration. The *error* signal is also stored in a register and sent to the source router, to request the flit retransmission. Figure 4. shows a block diagram of the modified router input port.

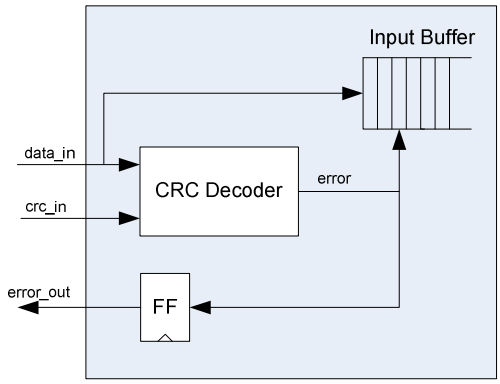


Figure 4. Block diagram of buffer consuming protocol.

The scheme presented in Figure 4. enables error recovery in one clock cycle. The source router has stored the correct flit in some input port, as shown in Figure 5. . The flit is transmitted to the output port through the internal router crossbar. The incoming *error_in* signal is connected to the input buffer controller, and if it is asserted, the *data_out* receives the last transmitted flit, which in practice corresponds to retransmission.

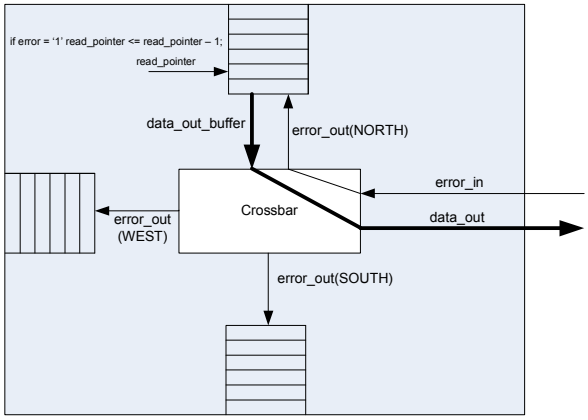


Figure 5. Error signal routing and data read from source buffer.

During a process of data recovery, the source input buffer can continue to receive data in its input port, if there is available space. Therefore, the final impact on latency is very small. The area overhead induced by such approach is basically one encoder/decoder pair, placed at each router port. It is important to mention that buffers, responsible for at least 80% of the consumed router power [14], kept the same sizing.

## IV. FAULT MODELING AND FAULT INJECTION

This section describes the strategy used to simulate the network and the fault injection technique for the evaluation of the implemented error recovery mechanism.

### A. Saboteur Module

A module named *saboteur* was developed to control the fault injection during the simulation process. This module is responsible for monitoring the data transmitted in each channel of the network, and according to its patterns, changes the value of some data bits to simulate the crosstalk effect. The MAF model, described in [15], is used as a reference for the implementation of the saboteur module.

Each NoC link is connected to a saboteur module. Besides fault injection, each saboteur module counts the amount of data transmitted in the link during a full simulation and the amount of errors injected on each link, enabling to generate statistics related to the fault injection on each channel.

### B. Crosstalk Effects

Crosstalk effect is observed at each interaction of two adjacent wires running long distances in parallel. Each wire has its load capacitance and resistance, and each pair of wires has its cross-coupled capacitance that causes interferences during switch activity. TABLE I. depicts the delay ratio factor *g* for a bus wire as a function of simultaneous transitions on neighboring lines [16].

TABLE I. RELATION BETWEEN TRANSITION DIRECTIONS AND THE DELAY FACTOR G.

| bit *k*-1 | bit *k* | bit *k*+1 | Factor *g* |
|---|---|---|---|
| ↑ | ↑ | ↑ | 1 |
| ↑ | ↑ | – | 1+*r* |
| ↑ | ↑ | ↓ | 1+2*r* |
| – | ↑ | – | 1+2*r* |
| – | ↑ | ↓ | 1+3*r* |
| ↓ | ↑ | ↓ | 1+4*r* |

The symbols ↑, ↓ and – represent positive transition, negative transition and no transition respectively. The factor *r* is a relation between inter-wire capacitance and the relative capacitance of the wire and the ground signal. In a usual situation, where these capacitances are the same, the *r* factor is 1. Thus, in the worst case scenario, where both neighbors transits on the opposite direction of the victim wire, the delay factor *g* is 5. Consequently, the wire delay can vary over 500% between the worst and the best case, just as a function of the direction of the transitions on neighboring wires.

### C. MAF model

The MAF model simplifies the creation of test vectors to induce the occurrence of crosstalk effects in integrated circuits. This model reduces the fault set by considering the worst case combinations of coupling capacitances between all possible aggressors. Therefore, the MAF model considers all N-1 aggressors transitioning in the same direction as a fault. This model considers that only one fault is modeled for each error on a victim line $Y_i$, and only one set of transitions can excite that fault. Figure 6. shows the necessary transitions to excite four types of fault for a victim wire $Y_i$ according the MAF model.

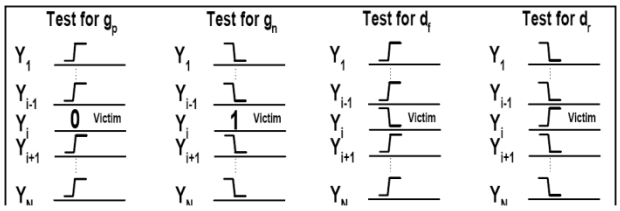


Figure 6. Required transitions on MAF model.

The fault model has 4 error conditions for each N-line wide set of interconnects to be tested: (*i*) $\mathbf{g_p}$: positive glitch error; (*ii*) $\mathbf{g_n}$: negative glitch error; (*iii*) $\mathbf{d_f}$: falling delay error; (*iv*) $\mathbf{d_r}$: rising delay error.

In this work, it is considered every 5 bits in parallel, on each 16 bits data buses, to check MAF model conditions. If one of them is meet, the victim value changes. The amount of errors injected into the network allows the validation of the architecture. Note that the injection fault considers worst-case situations, and in real circuits, the rate of errors due to crosstalk is smaller than the one simulated using the MAF model.

## V. RESULTS

This section presents area and latency results, comparing the reference NoC to fault tolerant NoC developed in this work.

### A. Area Overhead

An 8x8 network with and without the fault tolerant modules was synthesized using the Cadence Encounter RTL Compiler (0.35 µm standard cells library). TABLE II. shows the area results.

TABLE II. AREA RESULTS FOR AN 8X8 NETWORK.

| Module | N# of Cells **Original** | N# of Cells **Fault Tolerant** |
|---|---|---|
| Central Router (5 ports) | 3537 | 4025 |
| -Buffer | 547 | 590 |
| -Error Control Logic | 0 | 256 |
| **Total n# 8x8 NoC cells** | **208864** | **236672** |

From TABLE II, it is possible to note that the fault tolerance mechanism have a moderate impact on area compared to the original network. The total standard cells area increased 13.3%. Besides, none of the related works presents area overhead results. Such overhead has an acceptable cost, since it protects the network against 93.75% of possible crosstalk faults in the links. It is important to observe that the area overhead is mainly due to the addition of CRC encoders and decoders on each port of the routers, and the choice of CRC coding was made due its simplicity and smaller area. In addition, such result points out that more complex codes should probably not be afforded to be employed for fault tolerance in NoCs.

Mapping the same networks to FPGAs, the area overhead was 15.6%, and expected operating frequency is reduced by 8%.

### B. Latency Impact

To analyze the network performance in the presence of crosstalk faults, we define the following test scenario: (*i*) spatial traffic distribution: random destination; (*ii*) temporal traffic distribution: normal distribution, with an average of 20% and 10% of the available link bandwidth.

In the first simulation, with an average injection rate equal to **20%**, each router sends 100 48-flits packets, resulting in 6,400 transmitted packets. Two error injection rates are adopted: 0.0717%. (1,181 injected errors) and 2.03% (33,323 injected errors). The first line of the TABLE III. presents the average latency for the reference NoC (in clock cycles), and the second line the average latency for the fault tolerant NoC, without error injection. As expected, the average latency is the same. The third line of the Table presents the latency overhead with the smaller injection rate: 1.8%. The last line of the Table presents the latency overhead in the presence of a higher injection rate: 13%.

TABLE III. AVERAGE LATENCY VALUES FOR AN INJECTION RATE EQUAL TO 20% OF THE AVAILABLE BANDWIDTH.

| Fault Tolerance Support | Transmitted Packets | Errors Injected | Average Packet Latency |
|---|---|---|---|
| **NO** | 6,400 | **0** | 839.39 |
| **YES** | 6,400 | **0** | 839.30 |
| **YES** | 6,400 | **1,181** | 854.77 |
| **YES** | 6.400 | **33,323** | 948.47 |

In the second simulation scenario, with an average injection rate equal to **10%**, each router sends 200 48-flits packets, resulting in 12,800 transmitted packets. Two error injection rates are adopted: 0.113%. (3,689 injected errors) and 2.23%. (72.392 injected errors). The results presented in TABLE IV. shows smaller latency values, due to the smaller congestion inside the network (such average value is near to the minimal latency values - 70 clock cycles). The average latency is in practice the same, with or without error injection at lower injection rates.

TABLE IV. AVERAGE LATENCY VALUES FOR AN INJECTION RATE EQUAL TO 10% OF THE AVAILABLE BANDWIDTH.

| Fault Tolerance Support | Transmitted Packets | Errors Injected | Average Packet Latency |
|---|---|---|---|
| **NO** | 12,800 | **0** | 101.08 |
| **YES** | 12,800 | **0** | 101.08 |
| **YES** | 12,800 | **3,689** | 101.11 |
| **YES** | 12.800 | **72.392** | 101.77 |

Such results demonstrate the small latency overhead induced by the proposed fault tolerant mechanism. The higher latency overhead is observed only for higher

injection and error rates. In addition, in both scenarios all transmitted packets were received without errors, proving the effectiveness of the CRC circuit.

TABLE V. shows the latency of some individual packets (worst, best and typical case), in clock cycles, for packets with retransmitted flits (simulation with injection rate equal to 10%).

TABLE V. LATENCY OF PACKETS WITH AND WITHOUT FAULT RECOVERY.

| Packet ID | Latency with fault recovery | Latency without fault recovery | Difference (ck cycles) |
|---|---|---|---|
| 11767 | 351 | 299 | 52 |
| 6800 | 298 | 297 | 1 |
| 968 | 291 | 278 | 13 |

As we can observe, the latency overhead on packets with recovered flits is the worst case 17%. It is important to note that this is a worst-case scenario, where the same crosstalk fault is repeated and recovered in each link in the path between the source and target router.

## VI. CONCLUSIONS AND FUTURE WORKS

In this paper, we presented a NoC architecture with crosstalk fault tolerance on network links. The main goal of the architecture presented here is to maintain the original network performance either in presence or in absence of errors. The implemented strategy has high fault coverage due to the use of 4 parity bits on each flit, which covers 93.75% of possible error patterns, thus offering enough reliability for critical applications. The results show that the proposed architecture have small area and latency impact.

As future works we intend to implement other strategies of error recovery in the same network, including error correction and routing algorithms, and evaluate the advantages and disadvantaged over the method mentioned here. We also plan to measure the energy consumption overhead of this approach and analyze the network performance with real applications scenarios, such as multiprocessor applications.

## ACKNOWLEDGMENTS

This research was supported partially by CNPq (*Brazilian Research Agency*), projects 300774/2006-0, 471134/2007-4.

## REFERENCES

[1] Vellanki P.; et al. “Quality-of-Service and Error Control Techniques for Network-on-Chip Architectures”. *In: Great Lakes Symposium on VLSI (GLSVLSI’04)*, pp. 45-50, 2004.

[2] Benini L.; Micheli G. “Networks on chips: a new SoC paradigm”. IEEE computer, vol. 35, no. 1, pp. 70-78, 2002.

[3] Huang P.; et al. “Low Power and Reliable Interconection with Self-Corrected Green Coding Scheme for Network-on-Chip”. In: *Second International Symposium on Networks-on-Chip (NoCS 2008)*, pp. 77-83, 2008.

[4] Murali S.; et. al. “Analysis of Error Recovery Schemes for Networks on Chips”. IEEE Design and test of computers, vol. 22, no.5, pp. 434-442, 2005.

[5] Marculescu R. “Networks-on-Chip: The Quest for On-Chip Fault-Tolerant Communication”. In: *IEEE Computer Society Annual Symposium on VLSI Design (ISVLSI'03)*, pp. 8-12, 2003.

[6] Zimmer H.; Jantsch A. “A Fault Model Notation and Error-Control Scheme for Switch-to-Switch Buses in a Network-on-Chip”. In: *Hardware/Software Codesign and System Synthesis (CODES+ISSS’03)*, pp. 188-193, 2003.

[7] Bertozzi D.; Benini L. “Xpipes: A Network-on-chip Architecture for Gigascale Systems-on-Chip”. IEEE Circuits and Systems Magazine, vol. 4, no. 2, pp. 18-31, 2004.

[8] Pullini A.; et al. “Fault Tolerance Overhead in Network-os-Chip Flow Control Schemes”. In: *18th Symposium on Integrated Circuits and Systems Design (SBCCI’05)*, pp. 224-229, 2005.

[9] Murali S.; Atienza D.; Benini L.; Micheli G. “A Method for Routing Packets Across Multiple Paths in NoCs with In-Order Delivery and Fault-Tolerance Gaurantees”. VLSI-Design Journal, vol. 2007, no. ID 37627, pages 11, 2007.

[10] Zhu H.; Pande P.; Grecu C. “Performance Evaluation of Adaptive Routing for achieving Fault Tolerance in NoC Fabrics”. In: *Application-specific Systems, Architectures and Processos (IEEE ASAP 2007)*, pp. 42-47, 2007.

[11] Grecu, C.; et al. “Essential Fault-Tolerance Metrics for NoC Infraestructures”. In: *IEEE International On-Line Testing Symposium (IOLTS 2007)*, pp 37-42, 2007.

[12] Moraes F.; et al. “HERMES: an Infrastructure for Low Area Overhead Packet-switching Networks on Chip”. Integration, the VLSI Journal, vol. 38, pp. 69-93, 2004.

[13] Sprachmann M. “Automatic Generation of Parallel CRC Circuits”. IEEE Design & Test of Computers, vol. 18, no.3, pp. 109-114, 2001.

[14] Guindani, G.; et al. “NoC Power Estimation at the RTL Abstraction Level”. In: *IEEE Computer Society Annual Symposium on VLSI Design (ISVLSI'08)*, pp. 475-478, 2008.

[15] Cuviello M.; et al. “Fault Modeling and Simulation for Crosstalk in System-on-Chip Interconnects”. In: *IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD’99)*, pp. 297-303, 1999.

[16] Rabaey J. “Digital Integrated Circuts”. Ed. Prentice Hall, 792 p, 2003.