CROSSTALK FAULT TOLERANT NOC - DESIGN AND EVALUATION

Alzemiro H. Lucas, Alexandre M. Amory, Fernando G. Moraes
Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Av. Ipiranga, 6681 - prédio 32 - Porto Alegre - Brazil - CEP 90619-900
{alzemiro.silva, alexandre.amory, fernando.moraes}@pucrs.br

Abstract. *The innovations on integrated circuit fabrics are continuously reducing components size, which increases the logic density of systems-on-chip (SoC), but also affect the reliability of these components. Chip-level global buses are especially subject to crosstalk faults, which can lead to increased delay and glitches. This paper evaluates different crosstalk fault tolerant approaches for Networks-on-chip (NoCs) links such that the network can maintain the original network performance even in the presence of errors. Three different approaches are presented and evaluated in terms of area overhead, packet latency, power consumption, and residual fault coverage. Results demonstrate that the use of CRC coding at each link is preferred, when minimal area and power overhead are the main goals. However, each one of the methods presented here has its own advantages and can be applied depending on the application*.

Key-words: Networks-on-Chip (NoCs); fault tolerance; reliability; error correction and detection.

1 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 SoC design challenge 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 induced 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.

Compared to buses, NoCs provide more opportunities to implement fault tolerance techniques for intra-chip communication. For instance, a NoC has multiple paths for any pair of modules, which can be exploited to improve the fault tolerance of the communication by using adaptive routing algorithms. Techniques based on codification for error detection/correction can also be applied for NoCs. Other approaches include place and route techniques to avoid routing of bus lines in parallel, changes in the geometrical shape of bus lines and addition of shielding lines 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 [3], due to strict performance and power constraints of NoC architectures.

This paper evaluates error recovery mechanisms to increase the reliability of NoC links, making it resilient against crosstalk faults. As a design constraint, the evaluated mechanisms must be able to keep the NoC performance (latency, throughput, and bandwidth) in case of errors. Additional design constraints include low area overhead, high fault coverage, and minimum delay.

This paper is organized as follows. Section 2 presents related work in fault tolerance for NoCs. Section 3 presents the design of the crosstalk fault tolerant NoCs. Section 4 reports the fault modeling used to simulate and validate the network. The fault tolerant NoCs are evaluated and compared in Section 5. Finally, Section 6 concludes this paper.

2 RELATED WORK

Several types of faults and fault tolerance techniques can be applied to NoCs. Faults can be classified, according to the fault *duration*, as a transient or permanent fault, or according to the *moment* the fault is detected, as an on-line (self-checking) or off-line approach. The types of fault tolerant solutions for NoCs range from adaptive routing algorithms (used to find alternative paths for communication), coding techniques (use redundancy codes to detect and/or correct bit flips), retransmission in case of faults detected, and a combination of those techniques. The retransmission approach can be further classified according to the place where the retransmission takes place: router-to-router or end-to-end.

This paper focuses on on-line self-checking fault-tolerance techniques for transient faults on NoC links, employing coding techniques and router-to-router retransmission. Our goal is to evaluate different implementations, comparing metrics as silicon area, latency, and power consumption. Although one of the presented architectures can detect some faults in the router, our primary goal is to protect NoC links. Other fault-tolerance techniques that deviate from our focus, as adaptive routing, are not mentioned.

Table 1 summarizes the related work. Zimmer [4] 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 buses 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 [5] 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 [6] addresses a NoC architecture with Quality of Service (QoS) and error control techniques. This paper presents the evaluation of the proposed architecture, considering latency and power dissipation metrics. The two QoS methods addressed in this work are guaranteed throughput and best effort, and the error control techniques

Table 1 – Comparison of related works.

Reference	Method	Implementation/ Evaluation	Metrics	Types of Faults
ZIM03	- CRC/Hamming on links - Fault Model - QoS	Implementation	- Residual error rate	Transient
BER04	- Source CRC - Switch-to-switch retransmission	Implementation	Not presented	Transient
VEL04	- Parity/Hamming on links - QoS	Implementation	- Latency - Power	Transient
MUR05	- End-to-end retransmission - Switch-to-switch retransmission: -Flit level -Packet level - Correction + Detection	Evaluation	- Latency - Power - Residual error rate	Transient
GRE07	- End-to-end retransmission - Switch-to-switch retransmission: -Flit level -Packet level - Retransmission with and without priority	Evaluation	- Message arrival probability - Average detection time - Average correction time	Transient
Proposed work	*- CRC/Hamming on links* *- Source CRC* *- Switch-to-switch retransmission/ correction*	*Implementation/ Evaluation*	*- Latency* *- Area Overhead* *- Power* *- Residual error rate*	*Transient*

include single error detection and retransmission, and single error correction.

Murali [7] explores different error recovery methods for NoCs, evaluating for each one the energy, error protection, and latency overhead. The methods include end-to-end error detection, router-to-router error detection, at either at the flit-level and at the packet-level, and a hybrid scheme with correction of single errors and detection of multiple errors. This paper shown that packet storage is responsible for the major part of the 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.

Grecu [8] 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 error detection and recovery techniques for NoCs. Compared to [5] and [7] this work presents 3 different solutions to apply fault tolerance on NoCs, considering the implementation feasibility of each technique, concerning area overhead, power consumption and residual fault analysis.

3 CROSSTALK FAULT TOLERANT NoC ARCHITECTURES

This paper presents three different strategies for fault tolerance on NoC links. All methods use as reference design the Hermes NoC [9]. Hermes is a configurable infrastructure, specified in VHDL at RT level, aiming to implement low area overhead packet switching NoCs. Routers have up to five bi-directional ports and each input port stores received data on a FIFO buffer. It uses the XY routing algorithm and a centralized round-robin arbitration grants access to incoming packets. The Hermes NoC architecture has been configured as an 8*x*8 2-D mesh network, flit size equal to 16 bits, 8-flit buffer depth, without virtual channels.

Two coding techniques are used for error detection/correction: CRC and Hamming.

CRC codification is adopted for error detection, due its small complexity on logic implementation, and small parity delay calculation when using a parallel architecture to generate N parity bits per clock cycle. The designed CRC circuit encodes 16 bits (flit width) per clock cycle and generates four parity bits. Using this approach, 93.75% of all possible 16-bit error patterns can be detected [10]. Sequential and combinational designs may be used to implement the CRC circuitry. The combinational design has been chosen because it does not add latency to the network, and the extra delay of the combinational logic is considered small. Considering the polynomial generator is $g = 1+X+X^4$, the resulting combinational circuit for the CRC encoder is presented below:

$$p_0 = r_{15} \oplus r_{11} \oplus r_8 \oplus r_7 \oplus r_5 \oplus r_3 \oplus r_2 \oplus r_1 \oplus r_0$$
$$p_1 = r_{12} \oplus r_9 \oplus r_8 \oplus r_6 \oplus r_4 \oplus r_3 \oplus r_2 \oplus r_1$$
$$p_2 = r_{13} \oplus r_{10} \oplus r_9 \oplus r_7 \oplus r_5 \oplus r_4 \oplus r_3 \oplus r_2$$
$$p_3 = r_{15} \oplus r_{14} \oplus r_{10} \oplus r_7 \oplus r_6 \oplus r_4 \oplus r_2 \oplus r_1 \oplus r_0$$

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.

The second coding technique used in this work is the Hamming code, which enables the correction of flits with error in one bit at most. The Hamming coder and decoder are based on the function *hamgen*() provided by Matlab. The following equations define the Hamming encoder:

$$p_0 = r_{15} \oplus r_{12} \oplus r_{10} \oplus r_9 \oplus r_6 \oplus r_5 \oplus r_4 \oplus r_3 \oplus r_2$$
$$p_1 = r_{14} \oplus r_{11} \oplus r_9 \oplus r_8 \oplus r_5 \oplus r_4 \oplus r_3 \oplus r_2 \oplus r_1$$
$$p_2 = r_{15} \oplus r_{13} \oplus r_{12} \oplus r_9 \oplus r_8 \oplus r_7 \oplus r_6 \oplus r_5 \oplus r_1 \oplus r_0$$
$$p_3 = r_{14} \oplus r_{12} \oplus r_{11} \oplus r_8 \oplus r_7 \oplus r_6 \oplus r_5 \oplus r_4 \oplus r_0$$
$$p_4 = r_{13} \oplus r_{11} \oplus r_{10} \oplus r_7 \oplus r_6 \oplus r_5 \oplus r_4 \oplus r_3$$

The Hamming decoder has a similar design plus an additional logic to correct the erroneous bit indicated by the syndrome.

3.1 NoC with link CRC

Figure 1 illustrates the first fault tolerant architecture implemented, which protects only the NoC links. This strategy provides router-to-router flit-level error detection and retransmission. This figure represents two adjacent routers, the sender and the receiver routers, and a link between them. Modifications compared to the non-fault tolerant design are in grey. It includes a CRC encoder at the sender, a CRC decoder and an error flip-flop at the receiver, additional signals *crc_in* and *error_out* in the link, and slight modifications in the input buffers.

When the sender sends a flit to the receiver, it encodes the CRC in parallel due to its combinational logic. The CRC is sent through the *crc_in* signal to the receiver, which decodes it and test for faults. If there is no fault, the flit (not the CRC) is stored in the receiver buffer. If some fault arrives, the flit is not stored into the buffer, and an error is signaled, through the *error_out* signal, back to the sender, which

retransmits the last flit. This approach enables error recovery in one clock cycle.

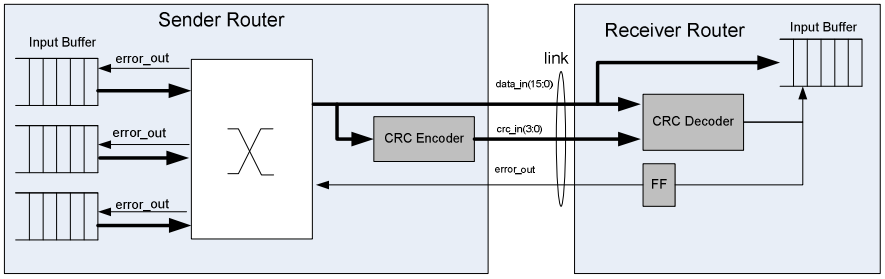

Figure 1 - Block diagram of the fault tolerant NoC design based on CRC for links.

The benefit of this approach is that the buffers and the buses width inside the router remain unchanged, saving silicon area. Only the external router interface receives new signals for error detection and recovery. The impact in silicon area, power, and delay is smaller. There is no impact on the latency when the network has no faults. Under a faulty condition the latency is incremented by one clock cycle only, which is an advantage compared to an approach based on end-to-end retransmission. This approach also provides the following additional 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 sender routers buffers, inducing smaller area overhead.

3.2 NoC with source CRC

This section presents the second fault tolerant NoC architecture, which is illustrated in Figure 2. This figure illustrates a path from the source router to a given router located in the path to the destination router. Modifications compared to the non-fault tolerant design are in grey. It includes a CRC encoder at the sender node, a CRC decoder

and an error flip-flop at the receivers (intermediate routers and destination node), additional signals *crc_in* and *error_out* in the links, and a slight modification in the input buffers.

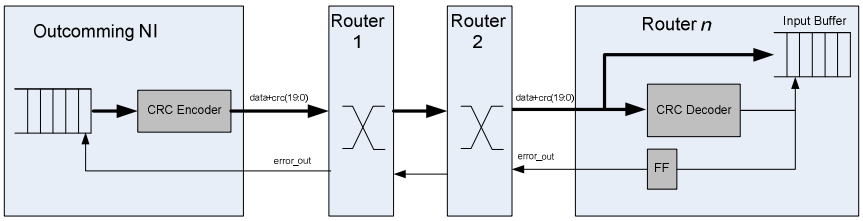

Figure 2 - Block diagram of the fault tolerant NoC design based on source CRC coder.

The main modification compared to the design in Section 3.1is that the CRC encoder is located only at the Network Interface (NI) of the module connected on the router local port. The CRC bits became part of the flit, increasing the width of all buffers of the network, as well as the internal buses of the routers, so that the CRC bits are carried through the network as part of the packet. As can be seen, the *data_out* signal have now 20 bit instead of 16 as the previous network, because this signal includes the *crc_out* signal, presented before in Figure 1.

This network has the advantage of using four less CRC modules at each router (local port does not have CRC). On the other hand, it increases the buffer width, which increases the silicon area, the power consumption, and the delay of routers. This approach uses the same mechanism to recover the corrupted data from the input buffers at the previous router, thus, this network presents the same latency as the previous network to retransmit corrupted flits (one clock cycle).

Another advantage of this approach is that it can not only protect the links, but also protect certain internal logic of the router. It is possible to detect transient or even permanent faults in certain internal modules of the routers like the buffers and the crossbar, however, it cannot detect faults in most of the router control logic. Another limitation is that, if the fault is a bit-flip in a buffer, the error cannot be recovered. Therefore, other techniques for fault tolerance should be adopted.

3.3 NoC with Hamming on Links

Figure 3 presents a simplified structure of the network with Hamming code on links. This figure represents two adjacent routers

(sender and receiver) where the link has been changed to carry Hamming parity bits. The sender has a combinational Hamming encoder at the output ports and the receiver has a combinational Hamming decoder at the input ports. Unlike the previous networks, none of the internal modules of the routers were changed.

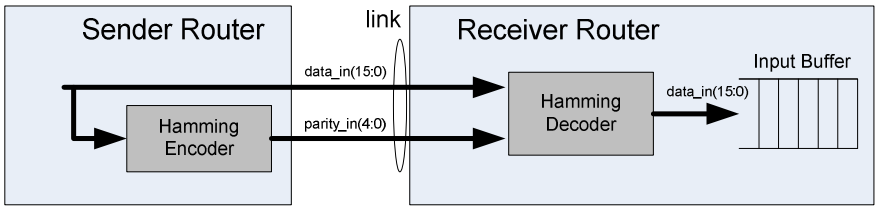

Figure 3 - Block diagram of the fault tolerant NoC design based on Hamming code.

The incoming flits of the receiver are first decoded and then saved in to the buffer. If there is a single fault the Hamming decoder corrects it transparently, without need of flit retransmission, thus, unlike the previous networks, this approach does not add latency to the network in a faulty situation. Note that there is no error signal from the receiver to the sender like the previous networks.

4 FAULT MODELING AND FAULT INJECTION

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

4.1 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 2 depicts the delay ratio factor g for a bus wire as a function of simultaneous transitions on neighboring lines [11].

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 transit 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 [11].

Table 2 - 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*

4.2 Maximal Aggressor Fault (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 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 4 shows the necessary transitions to excite four types of fault for a victim wire Y_i according the MAF model.

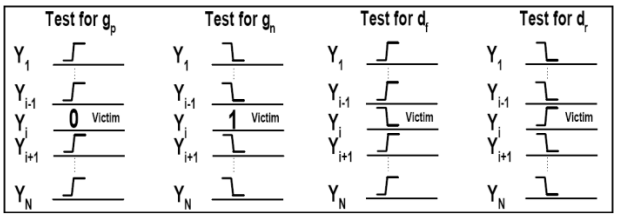

Figure 4 - Required transitions for MAF model [1].

The fault model has four 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.

4.3 Saboteur Module

A module named *saboteur* is 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 Maximal Aggressor Fault (MAF) model, described in [1], is used as a reference for the implementation of the saboteur module (see Section 4.2).

Each NoC link is connected to a saboteur module as illustrated in Figure 5. 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.

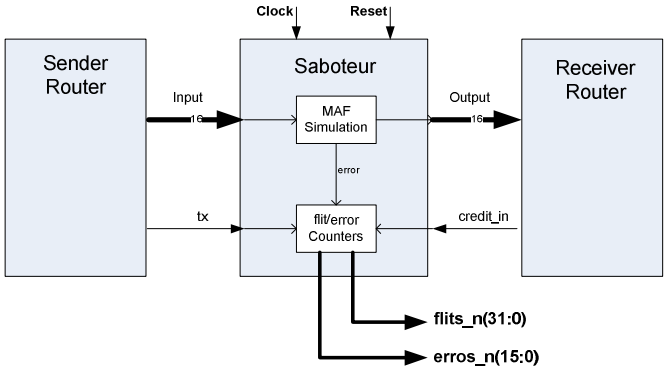

Figure 5 – Saboteur located between two routers.

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. This module can also be parameterized to check individual conditions of MAF model as shown in Section 4.2. The amount of errors injected into the network allows the validation of the architecture. Note that the fault injection considers worst-case situations, and in real circuits, the rate of errors due to crosstalk is smaller, since not all true MAF conditions generate faults.

5 RESULTS

This section presents area, power, latency and residual fault analysis, comparing the reference NoC to fault tolerant NoCs developed in this work.

5.1 Area Overhead

Four networks have been implemented and synthesized using Cadence Encounter RTL Compiler (0.35um standard cells library) to evaluate the silicon area. Table 3 shows the results for a router with 5 ports and for an 8x8 network.

Table 3 - Area results for an 8x8 network (FT means Fault Tolerant).

	Original network	FT NoC Link CRC		FT NoC Source CRC		FT NoC Hamming	
	# of Cells	# of Cells	%	# of Cells	%	# of Cells	%
Router w/ 5 ports	3537	4025	13.8	4145	17.2	4149	17.3
- Buffer	547	590	7.8	630	15.3	547	0
- FT Logic	0	256	-	144	-	612	-
Total NoC cells	208864	236672	13.3	243968	16.8	243136	16.4

The total standard cell area increased 13.3% for the FT NOC with CRC in the links. This area overhead is due to the addition of CRC encoders and decoders at each router port.

The area overhead of the NoC with CRC computed at the source router is 16.8%. The area overhead for this NoC comes from the increased buffer size (*flit* + CRC bits). As previously mentioned, this network can also provide some protection to transient faults on internal modules of the routers, justifying its use in designs where router fault tolerance is required.

The network with Hamming on links presented an area overhead of 16.4% compared to the original network. This overhead is due to the higher complexity of the Hamming decoding circuitry. The advantage of this technique is the error correction without retransmission, not interfering in the network latency in the presence of faults.

In conclusion, these results point out that a more complex code would probably not be an affordable fault tolerance technique for a NoC similar to Hermes [9]. Perhaps, more complex NoCs could afford complex codes.

5.2 Latency Impact

Latency is evaluated using the following test scenario: (*i*) spatial traffic distribution: random destination; (*ii*) temporal traffic distribution: normal distribution, with an average injection rate of 20% and 10% of the available link bandwidth.

Table 4 presents results for the first scenario, with an average injection rate equal to 20% of the available link bandwidth. 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). As expected, both CRC architectures do not add extra latency in the absence of faults, and present the same average latency. The average latency increases 1.8% and 13% for a 1,181 and 33,323 injected faults respectively. The network with Hamming code does not add extra latency for error protection, however with higher error injection rates some faults are not corrected (residual faults).

Table 4 - Average latency, in clock cycles, for an injection rate equal to 20%.

Network	Transmitted Packets	Error Injection Rate (%)	Injected Errors	Average Packet Latency (clock cycles)	Latency Variation (%)
Original	6,400	0	0	839.39	0
Link CRC	6,400	0	0	839.30	0
		0.0717	1,181	854.77	1.8
		2.0300	33,323	948.47	13.0
Source CRC	6,400	0	0	839.30	0
		0.0717	1,181	854.77	1.8
		2.0300	33,323	948.47	13.0
Hamming	6.400	0.0717	1,181	839.39	0
		2.0300	33,323	839.39	0

The previous scenario with 20% of injection rate corresponds to a worst-case scenario where the network is congested. An injection rate of 10% of the available bandwidth, the second simulation scenario, corresponds to a more realistic NoC traffic behavior. In this simulation, 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 5 shows smaller latency values, due to the smaller congestion inside the network (such average value is near to the minimal latency value, 70 clock cycles). The average latency is in practice the same, with or without error injection at lower injection rates.

Table 5 - Average latency, in clock cycles, for an injection rate equal to 10%.

Network	Transmitted Packets	Error Injection Rates (%)	Injected Errors	Average Packet Latency (clock cycles)	Latency Variation (%)
Original	12,800	0	0	101.08	0
Link CRC	12,800	0	0	101.08	0
	12,800	0.113	3,689	101.11	0
	12.800	2.230	72.392	101.77	0

Table 6 shows the latency for some individual packets (worst, best and typical cases), in clock cycles, for packets with retransmitted flits (simulation with injection rate equal to 10%). The worst-case latency overhead is 17.4%. It is important to note that this is a worst-case scenario, where the same crosstalk fault is repeated and recovered at each link in the path between the source and target router.

Table 6 - Latency of packets with and without fault recovery.

Packet ID	Latency without Fault Recovery (clock cycles)	Latency with Fault Recovery (clock cycles)	Latency Variation (%)
11767	299	351	17.4
6800	297	298	0.3
968	278	291	4.7

5.3 Residual Fault Analysis

This section shows the effectiveness of both methods to protect the network against crosstalk faults. To analyze residual faults using CRC and Hamming codes, the saboteur module is parameterized to inject faults varying the number of MAF model conditions to increase the error injection rate. The simulated scenario considers a 5x5 network, with each IP sending 200 packets to a random destination, using 15% of the link available bandwidth. Table 7 shows the results of these simulations.

Table 7 – Residual fault analysis.

MAF Conditions	Transmitted flits	Injected errors	CRC Residual Faults		Hamming Residual Faults	
d_r	806,021	341	0	0%	0	0%
d_r, d_f	808,716	444	0	0%	0	0%
d_r, d_f, g_n	794,816	896	0	0%	0	0%
d_r, d_f, g_n, g_p	809,575	16,727	14	0.08%	389	2.32%

As expected, the CRC coding presents a lower residual fault rate, since its fault coverage is higher than the Hamming code. When all conditions of the MAF model are verified, a 2.32% and 0.08% residual fault rate is observed for the Hamming and CRC codes, respectively.

5.4 Power Consumption

The power consumption is measured using VCD analysis, with Synopsys Prime Time tool. The network is synthesized with the TSMC25 library, and simulated to generate the VCD files. A 5x5 network is used to generate the VCD files, with the same traffic scenario used to evaluate residual faults.

Table 8 shows the average power consumption for routers at different locations of the network (routers at the borders of the network may have 3 or 4 ports, and the central router has 5 ports). The power of the entire NoC is roughly the sum of the power of each router. The power consumption overhead of the network with CRC coding on links is small (1.9%), compared to the original network. On the other hand, the power consumption overhead of the network with source CRC reaches 18.9% since the buffer size increases (more 4 bits to store CRC values), confirming that most of the NoC power consumption is due to the buffers. The network with Hamming on links presented 5.4% power consumption overhead due to increased logic and the additional parity bit, but it can be considered an acceptable cost for this implementation.

Table 8 – Power consumption overhead for the proposed architectures.

	Original network (mW)	Link CRC (mW)	%	Source CRC (mW)	%	Hamming (mW)	%
3-port router	4.80	4.88	1.6	5.90	18.6	5.02	4.4
4-port router	6.34	6.46	1.9	7.82	18.9	6.70	5.4
5-port router	7.88	8.04	2.0	9.74	19.1	8.31	5.2
5x5 NoC	166.20	169.37	1.9	205.01	18.9	175.72	5.4

6 CONCLUSIONS AND FUTURE WORK

The goal of this paper was to evaluate different crosstalk fault tolerant methods for network links such that the network can maintain the original network performance even in the presence of errors. Among the three evaluated architectures, the CRC applied at each link is the recommend method to protect the network against crosstalk effects. The *source CRC* penalizes area and power. On the other hand, the *source CRC* enables to protect some internal router components, since data transmitted through the router is protected. The assumed advantage of the Hamming codification, no retransmission required, presented a smaller fault coverage and higher area overhead compared to CRC, however it can be an interesting alternative for some applications, where a small number of data errors can be tolerated and the latency needs to be minimal.

It is possible to enumerate the following future works: (*i*) explore a *source Hamming* architecture, verifying the feasibility to use it for internal router protection; (*ii*) develop new methods to protect the router, minimizing the use of classical redundant approaches (TMR); (*iii*) evaluate and propose adaptive routing algorithms for faulty routers.

7 ACKNOWLEDGMENTS

This research is supported partially by CNPq (*Brazilian Research Agency*), projects 300774/2006-0, 471134/2007-4 and by CAPES PNPD project 02388/09-0.

8 REFERENCES

[1] 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.

[2] H. H. K. Tang, K. P. Rodbell, “Single-event upsets in microelectronics fundamental physics and issues”. In: *Materials Research Society Bulletin*, vol. 28, pp. 111–116, 2003.

[3] Bertozzi D.; “The Data-Link Layer in NoC Design”. In: Micheli G., Benini L.; Networks on chips: Technology and Tools. Ed. Morgan Kaufmann, 408 p, 2006.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] Koopman P., Chakravarty T.; “Cyclic redundancy code (CRC) polynomial selection for ebedded systems”, In: *The International Conference on Dependable Systems and Networks*, pp. 1-10, 2004.

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