Analyzing the Error Propagation in a Parameterizable Network-on-Chip Router

Douglas Rossi de Melo, Cesar Zeferino, Luigi Dilillo, Eduardo Augusto Bezerra

Title: Analyzing the Error Propagation in a Parameterizable Network-on-Chip Router

Author(s): Douglas Rossi de Melo, Cesar Albenes Zeferino, Luigi Dilillo, and Eduardo Augusto Bezerra

DOI: 10.1109/LATW.2019.8704580

Published: 06 May 2019

Document version: Post-print version (Final draft)

Please cite the original version:
D. R. de Melo, C. Albenes Zeferino, L. Dilillo and E. A. Bezerra, ”Analyzing the Error Propagation in a Parameterizable Network-on-Chip Router,” 2019 IEEE Latin American Test Symposium (LATS), 2019, pp. 1-6, doi: 10.1109/LATW.2019.8704580.

Analyzing the Error Propagation in a Parameterizable Network-on-Chip Router

Douglas Rossi de Melo[*†‡], Cesar Albenes Zeferino[*], Luigi Dilillo[†], and Eduardo Augusto Bezerra[†‡]
[*]Laboratory of Embedded and Distributed Systems, University of Vale do Itajaí, Brazil
[†]Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier, France
[‡]Laboratory of Communications and Embedded Systems, Federal University of Santa Catarina, Brazil
drm@ieee.org

Abstract

The constant reduction in the components size in integrated circuits and the increase of the operating frequency make Systems-on-Chip (SoCs) more vulnerable to noise and other interference phenomena. Such phenomena can lead to faults, which generate errors that may result in a system crash. SoCs with dozens of cores use Networks-on-Chip as their interconnection architecture. In this context, this work presents an analysis of the error propagation in a parameterizable router architecture that allows different combinations of input and output controllers, data width and buffers depth. The router has been described focusing on design flexibility and low logical resource occupation. We elaborated different combinations of the router architecture and evaluated the error propagation by means of Single Event Upset fault injections. The synthesis results presented a reasonable increase in term of logical elements when applying wider data representations, and combinations using Mealy finite state machines in the routing component had the lowest error propagation rate at a price of a small degradation in the maximum operating frequency.

Index Terms

Systems-on-Chip; Networks-on-Chip; Router Architecture; Fault Tolerance.

I. Introduction

Systems-on-Chip (SoCs) with several dozens of processing cores use Networks-on-Chip (NoCs) as interconnection infrastructure to provide communication between cores. This architecture is reusable, scalable, and offers parallelism in communication [1]. A NoC has a set of routers and point-to-point channels interconnecting the routers in a structured path [2].

The continuous reduction in components size and the increase of the operating frequency make NoCs vulnerable to internal (power supply and crosstalk) and external (electromagnetic, thermal, ionizing particles interference) noise sources [3].

According to [4], several types of SoC require fault-tolerant components, depending on the target environment. A fault-tolerant NoC should be able to detect the occurrence of a fault and prevent the resulting error from causing a malfunction in the application. However, providing reliability in a NoC impacts performance, silicon costs, and power consumption because it is generally done through redundancy [3].

Due to the technological development and increasing integration, communication architectures used in computers for aerospace applications are made up of a growing number of processing cores. Some approaches that are proposed in the literature do not meet the communication requirements of future On-Board Computers (OBCs) with multiple cores [5]. NoCs represent a valid alternative to the bus for multicore interconnection, but their costs are not negligible, and fault tolerance techniques are needed for applications with reliability requirements.

Currently, fault tolerance in NoCs is primarily based on spatial redundancy. However, since replication increases power dissipation, in systems that have energy restrictions, such as in embedded and aerospace applications, it is necessary to seek for solutions that allow the provision of fault tolerance with low energetic impact.

In this context, this work aims to evaluate the performance in terms of resilience of a router using combinations of input controllers, data width and buffers depth, and to present the trade-off between hardware resources usage and the susceptibility to error propagation. The results showed that the use of Mealy Finite-State Machines (FSMs) for the routing structures provides a significant decrease in the number of propagated faults, at a price of a lower maximum operating frequency of the router. We found no previous work covering the use of different FSM approaches in routing design to increase the component resilience.

The remainder of this paper is structured as follows. Section II provides a brief account of the background of NoCs. Following this, Section III discusses the types of fault to which a NoC is subject. Section IV presents the characteristics of the router described for this evaluation. In Section V, we present the verification model used to measure the error propagation. Finally, Section VI presents the experimental results and Section VII summarizes the conclusions of the paper.

II. Networks-on-Chip

The interconnection in SoCs with few cores is generally performed by using shared buses. This approach is justified due to the re-usability of this architecture, which reduces design costs and time. However, for SoCs with several dozens of cores, a more complex interconnection structure is required in order to obtain performance scaling. In this direction, in the early 2000s, several studies pointed out that NoCs would be the best solution to solve this problem [2], [6], [7], [8], [9]. NoCs are reusable like the buses and offer parallelism in communication and scalable performance.

NoCs are derived from the interconnection networks that are used in parallel computers [10], [11]. A NoC consists of routers, links and network interfaces [2], as depicted in Fig. 1.

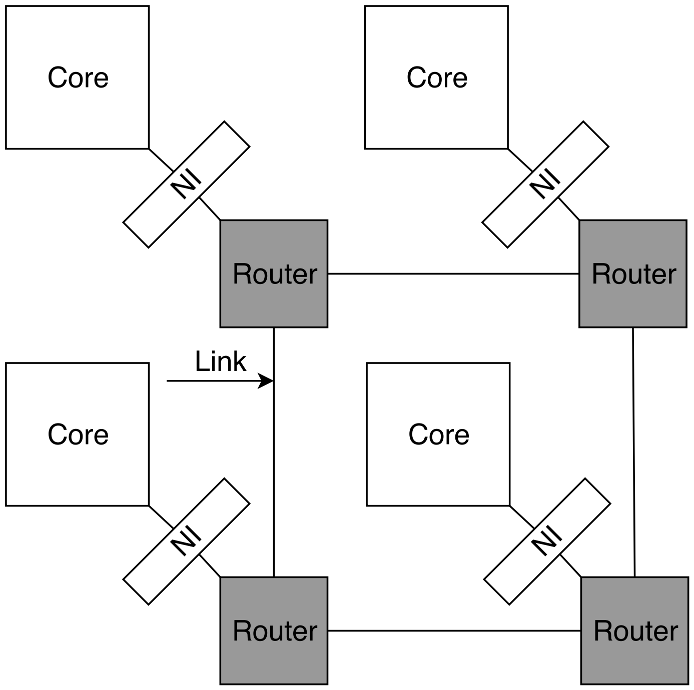


Fig. 1. Components of a NoC-based system

According to [11], a router has registers, multiplexers, routing and flow control circuits, and buffers for temporary storage of packets. It also has input and output ports for communication with other routers and with one of the cores connected to the NoC. Links usually have two unidirectional point-to-point channels, which may be synchronous or asynchronous. Each channel is made up of data, frame and flow control signals.

The Network Interface (NI) is the unit that transparently connects the processing core to the router. It is responsible for adapting the communication protocol that is used by the core to the one of the NoC. Network interfaces are classified according to the nature of the client, being a processor, a shared memory or an external channel [11].

The usual form of communication between cores in a SoC is the exchange of messages. Each message consists of a header (starting the packet), a payload (content) and a trailer (signaling the end of the packet), as shown in Fig. 2.

A. NoC and the OSI model

The data flow over the network is presented in [12]. Fig. 3 identifies the components of a SoC and relates the layers of an NoC to the layers of the Open System Interconnection (OSI) model. Each layer can provide services, including fault tolerance. The System layer corresponds to the processing cores and the application. The Interface layer decouples the cores from the network and controls the sending and receiving of messages. The Network layer is responsible for message routing, while the Link layer addresses coding, synchronization, and reliability issues.

B. NoC Features

A NoC can be characterized according to the following attributes [2]:

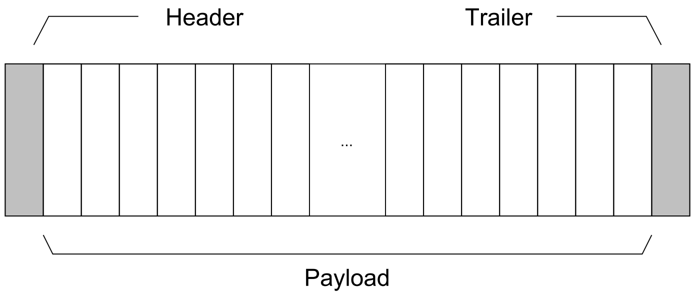


Fig. 2. NoC message structure

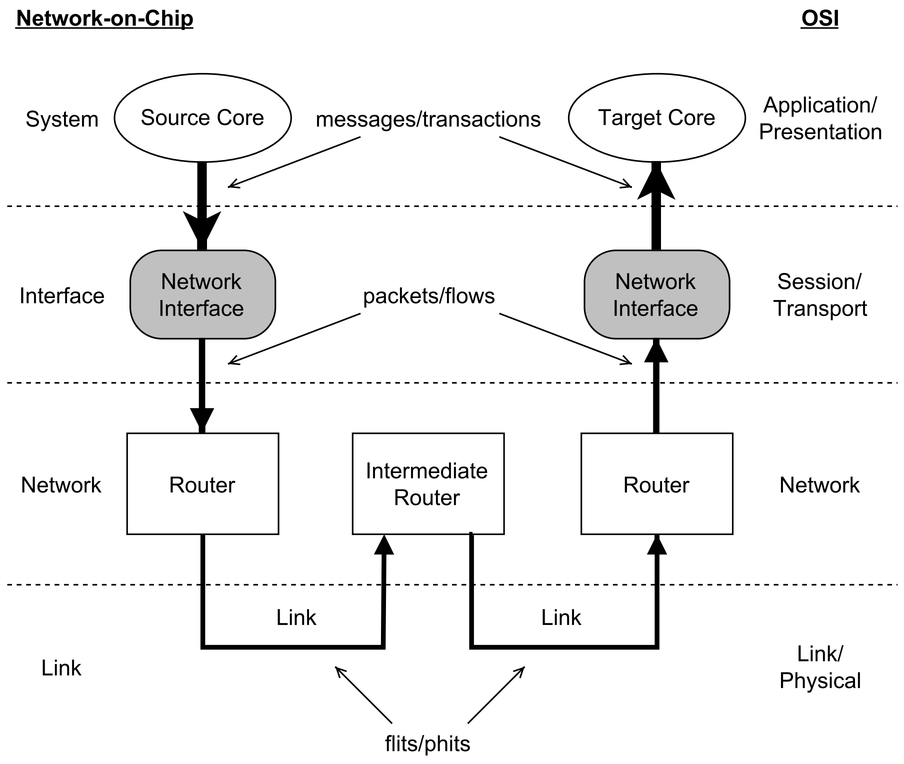


Fig. 3. The relationship between NoC layers and the OSI model [12]

Topology: Defines the arrangement of routers and links in the form of a graph. The most common topologies in NoCs are those of the planar type, like the 2-D mesh. The topology of a NoC defines the physical layout and the connections between the nodes and the channels in the network.

Flow Control: Deals with the allocation of the resources necessary for a message to advance through the network, performing the regulation of traffic in the channels. The data unit on which the flow control is performed is called Flow Control Unit (flit). Typically, a flit corresponds to a word of the physical channel called Physical Unit (phit).

Memorization: Packets destined for channels that are already allocated must wait for their release to be forwarded, what requires the implementation of some memorization scheme to store the blocked packets in queues within the router. Memorization is usually implemented only at the input channels.

Routing: It sets the path to be used to forward a message to its destination. There are various routing strategies, which are typically classified according to the number of destinations (unicast or multicast), the place in which the routing is performed (centralized, source or distributed), the physical implementation type (table-based or algorithm), and the adaptivity (deterministic, adaptive or oblivious). Routing algorithms in NoCs should prevent packets from being blocked in the network (*deadlock*) and avoid packets to move through the network without reaching their destination (*livelock*) [10].

Arbitration: It resolves a conflict that arises when two or more messages compete for the same channel. Round-Robin is the main arbitration scheme used in NoCs because it promotes an equal distribution of the channel.

Switching: It determines how a message is transferred from the input of a router to one of its output channels. The main types are circuit and packet switching.

III. Fault Tolerance

Several trends are leading to increased fault rates in commodity processors, such as smaller devices, hotter chips, more devices per processor, and more complex designs [4].

The relationship between fault, error, and failure is presented in [13]. A fault is active when it produces an error. An error can be propagated and turned into other errors. A system failure occurs when an error is propagated to the service interface and causes a deviation in its regular form of execution. Likewise, [4] considers a fault as a physical defect. A fault may manifest an error, and an error may result in an incorrect behavior, called a failure (Fig. 4).

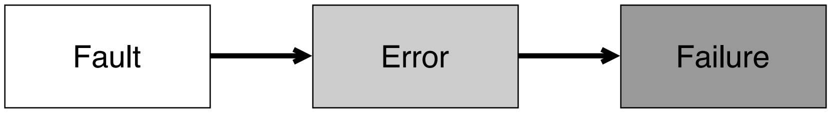


Fig. 4. Fault, error and failure relation

Faults and errors can be masked and not lead to an error or a failure. Masking occurs at several levels and because of several reasons, including:

- Logical masking: the effect of an error may be logically masked;
- Architectural masking: the effect of an error may never propagate to architectural state and never become a user-visible failure; and
- Application masking: even if an error does impact architectural state and thus becomes a user-visible failure, the failure might never be observed by the application software.

A. Fault Classification

Faults and errors can be classified according to their duration [14], as follows:

Transient Faults: Transient faults occur only once and do not persist. An error arising from a transient fault is generally known as Soft Error (SE) or Single Event Upset (SEU). One of the main causes of transient faults is radiation from particles produced when cosmic rays impact the atmosphere. Other sources are alpha particles, produced by the natural decay of radioactive isotopes. Such faults have a high incidence on memory elements of a system [4].

Permanent Faults: Permanent faults occur once and persist from that time. A permanent fault is assumed to be a repetitive error. These faults originate from physical wear-out, fabrication defects, or design failures. They are more likely to occur on the communication links [4].

Intermittent Faults: Intermittent faults occur repeatedly, but not continuously. They manifest themselves as intermittent errors and are generated by wires or devices that establish or not the connection according to the temperature or noise variation [14].

B. Faults in NoCs

Works in fault tolerance in NoCs address mainly transient and permanent faults. For instance, [15], [16], [17], [18], [19], [20] address SEU in NoC components. The works of [17], [18], [19], [20], [21] deal with transient faults due to crosstalk. In [22], [23], [24], the authors handle short and open circuit faults on the links of a NoC. In [25], [26], [27], [28], the authors discuss the yield of vertical links in 3D NoCs. Works dealing with intermittent faults treat them as permanent faults, as done in [22], [29], [30].

In this paper, we are addressing SEUs faults which can manifest in any of the registers present in the architecture of a NoC router.

IV. Router Architecture

The authors propose a parameterizable router architecture that has been implemented to evaluate the error propagation. The router has been described in VHDL focusing on simplicity of design and low area overhead. The overview of the architecture is shown in Fig. 5.

The router has four ports (*North*, *East*, *South*, and *West*) connecting neighbouring routers, and a *Local* port connected to a processing core through a NI. Each port is connected to a crossbar internally to the router, handling grants and requests signals, and to the neighboring structures by means of bi-directional links. A port is structured into input and output channels, and a simplified view is presented in Fig. 6.

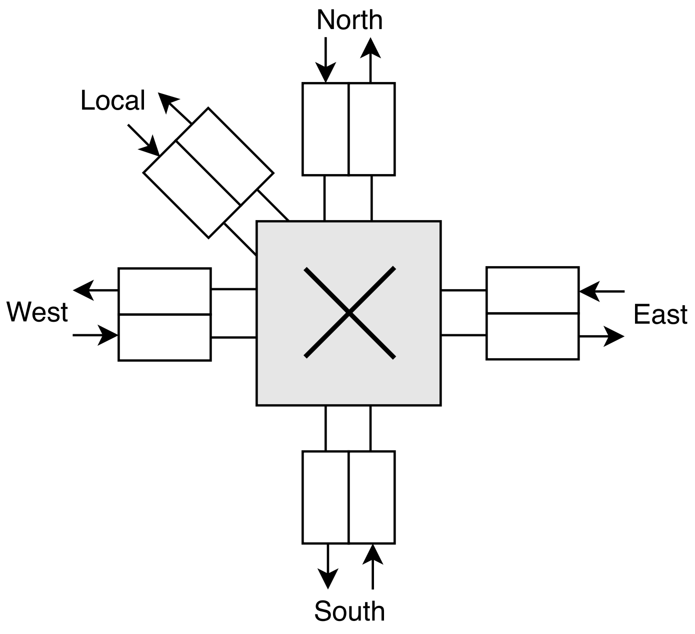


Fig. 5. Router architecture

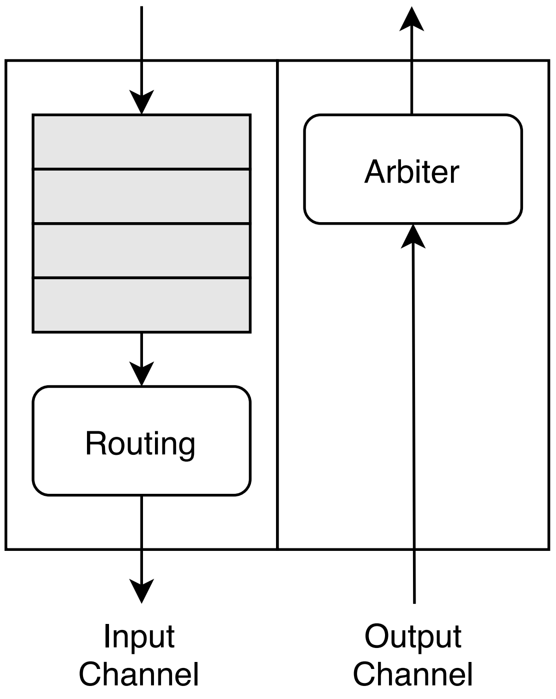


Fig. 6. Port structure

The input channel controls the incoming flow and has a buffer and a routing component. The buffer consists of a ring FIFO able to store *n* flits. The routing performs the XY algorithm in a Finite-State Machine (FSM) to request an output channel. The FSM scheme can be selected by the designer, with a Moore implementation, with less susceptibility to problems related to glitches, or a Mealy one, with lower latency.

The output channel handles the outgoing flow and has the arbiter as the main component. The arbiter is responsible for granting the output channel to a requesting input channel and uses a Round-Robin policy.

The router is intended to be used in 2-D mesh topology networks with the same width for flits and phits. The router performs a 4-stage handshake flow control and utilizes the wormhole switching technique. Fig. 7 shows the proposed packet structure.

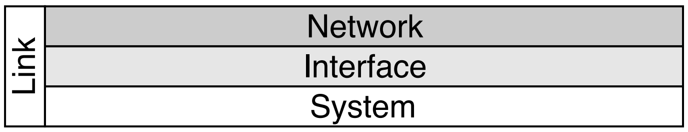


Fig. 7. Proposed packet structure

The proposed packet covers all the layers presented in Fig. 3. The Link field is a frame signal present in every flit and it is used for flow control. The first set of flits is related to the Network layer, containing the destination address optionally followed by the source information. Like the Network, the Interface field is also part of the header flits and has all the information needed by the NI for end-to-end communication. Lastly, the payload to be forwarded to the processing core is present at the System flits.

TABLE I
SYNTHESIS AND SIMULATION RESULTS FOR 4-FLIT BUFFERS

	Flit Width															
	4				**8**				**16**				**32**			
Routing	**LUT**	**FF**	**F_{max} (MHz)**	**Error Rate**	**LUT**	**FF**	**F_{max} (MHz)**	**Error Rate**	**LUT**	**FF**	**F_{max} (MHz)**	**Error Rate**	**LUT**	**FF**	**F_{max} (MHz)**	**Error Rate**
Moore	92	89	368.6	0.75	365	268	249.3	0.74	488	428	236.6	0.77	728	748	212.5	0.77
Mealy	99	89	319.4	0.45	429	268	190.7	0.38	535	428	153.9	0.29	769	748	153.0	0.25

TABLE II
SYNTHESIS AND SIMULATION RESULTS FOR 128-BIT BUFFERS

	Flit Width															
	4				**8**				**16**				**32**			
Routing	**LUT**	**FF**	**F_{max} (MHz)**	**Error Rate**	**LUT**	**FF**	**F_{max} (MHz)**	**Error Rate**	**LUT**	**FF**	**F_{max} (MHz)**	**Error Rate**	**LUT**	**FF**	**F_{max} (MHz)**	**Error Rate**
Moore	284	396	258.3	0.39	681	838	207.7	0.79	713	783	202.6	0.85	728	748	212.5	0.77
Mealy	298	396	206.5	0.13	710	838	148.0	0.13	749	783	146.8	0.16	769	748	153.0	0.25

V. FAULT INJECTION

This section first describes the combinations of the router architecture submitted to verification. Then, it presents the fault injection method and the fault model used for the experiments.

A. Router Verification

We elaborated 16 router architectures to perform the fault injection campaign. The combinations consist in the variation of the following parameters:

- Routing: XY algorithm using a Moore or a Mealy FSM;
- Data Width: flit width of 4, 8, 16, or 32 bits; and
- Buffer Depth: FIFOs storing 4 words or 128 bits.

A workload has been generated for the router evaluation. It was described to intermittently transmit packets to a fixed combination of input and output channels, emulating a 128-bit cache line transfer, commonly used in 32-bit processors. The following combinations were assigned to meet the requirements of the XY routing algorithm: *Local* $\rightarrow$ *East*; *East* $\rightarrow$ *West*; *West* $\rightarrow$ *South*; *South* $\rightarrow$ *North*; *North* $\rightarrow$ *Local*. Fig. 8 illustrates the connections within the router's crossbar.

The packet structure used for verification is presented in Fig. 9. Its structure is composed of a single bit to perform the flow control, a single flit as header and *n* payload flits. The header flit is used exclusively to address the coordinates of the destination router. Both header and the last payload flit (trailer) use '1' as frame bit, while regular payload flits use '0'.

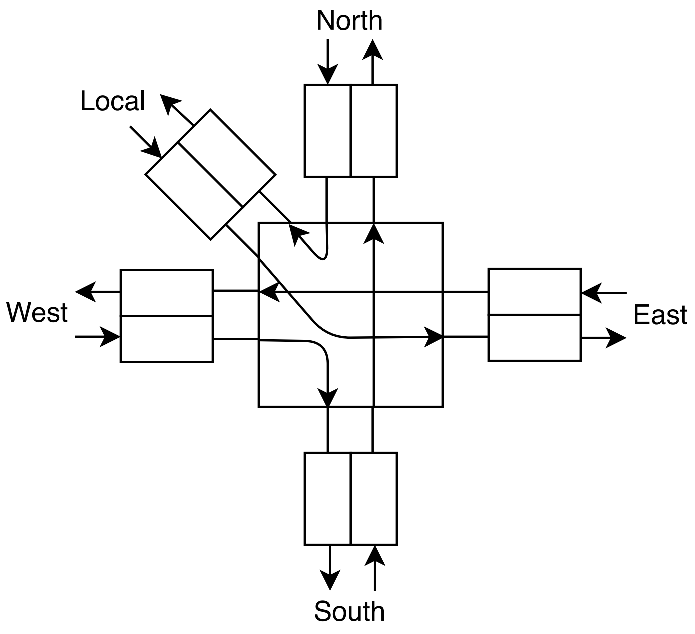
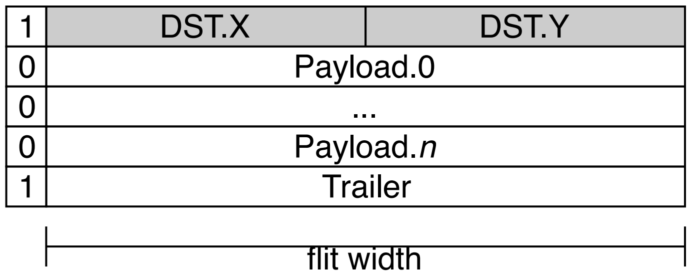


Fig. 8. Channel connections for verification



1	DST.X	DST.Y
0	Payload.0	
0	...	
0	Payload.*n*	
1	Trailer	

Fig. 9. Packet structure for verification

B. Fault Injection Environment

There are several different fault injection strategies proposed in the literature. They can be classified in hardware-based fault injection, software-based fault injection, simulation-based fault injection, emulation-based fault injection, and hybrid fault injection [31].

In this experiment, we are using the approach proposed by [32]. The technique is composed of a simulation-based fault injection relied on the use of built-in commands in the ModelSim-Intel® FPGA edition software. This setup enables interaction with the simulation engine, allowing the manipulation of signals (fault injection), and observation of the error propagation.

The following steps are performed for each interaction of the fault injection scheme:

1) Runs the simulation once without fault injection to obtain a golden run.
2) Lists all the flip-flops in the circuit and randomly picks one of them to inject a fault into it.
3) Randomly chooses when the fault is going to happen during the simulation time.
4) Simulates until the chosen injection instant.
5) Forces a bit flip in the selected flip-flop.
6) Simulates until a predefined timeout is reached.
7) Compares the stored outputs to the ones obtained from the golden run.

We adopted the SEU fault model for the fault injection campaign. In each experiment a single fault is injected by inverting the logic value of the target signal. If the output of any port differs from the golden run, then it is assumed that an error propagated in the circuit and resulted in a failure.

For each configuration, we performed 1,000 simulations running for 100μs each. This approach was employed for a more accurate measurement of the error propagation rate in all scenarios.

VI. Results

The router architectures were elaborated and synthesized using Intel® Quartus Prime software suite, Version 18.1, targeting the FPGA device 5CGTFD9E5F35C7 from the Cyclone V family. We obtained metrics related to Look-Up Tables (LUTs), Flip-Flops (FFs) and maximum operating frequency from the synthesis tool. Later, we got the error propagation ratio using ModelSim.

The synthesis and simulation results for routers with 4-flit buffers are presented in Table I. All the implementations with the same flit width use the equivalent amount of FFs, and presents an increase in terms of LUTs in Mealy implementations, due to the extra logic needed for output signals decoding. This increase also leads to performance degradation. However, the approaches using Mealy FSMs have a significantly lower error rate than those using Moore.

Table II presents results using a fixed buffer size (128 bits) in all the configurations. As observed in the previous approach, the similar use of FFs and the increase of LUTs was also present, as well as the decrease of the maximum operating frequency in architectures using Mealy controllers. Regarding error propagation, Moore approaches presented a rate variation in comparison with the ones in Table I, while Mealy-based routing presented an even more significant reduction in the error rate.

As expected, the wider the data flit is, the greater is the logical resources usage and also the lower the maximum operating frequency. The difference in error propagation between Moore and Mealy approaches is given mainly due to buffers occupancy rate. Architectures using Mealy save a cycle on routing, which speeds-up data forwarding and causes flits to be stored for a shorter time. This behavior occurs regardless buffers capacity, as can be seen in the error rate in Table I and Table II.

VII. Conclusion

Future systems for aerospace applications will embed several processing cores in a SoC. Such processing power is a demand from applications that require high performance in space due to the increasing amount of high-resolution onboard sensors, and the strict bandwidth requirements in satellite-Earth links. NoCs are the successors of the shared bus performing multicore interconnection. However, their silicon costs are not negligible. It is still a challenge to provide an interconnection architecture that meets dependability requirements at a low area overhead.

In this context, the primary goal of this work was to develop a simplified and parameterizable router architecture and to analyze the behavior of its customized architectures under the injection of SEU faults. The results obtained showed that the use of Mealy FSMs for the routing structures results in a significant decrease in the number of propagated faults, at a price of performance degradation.

As future steps, we intend to implement combinations of FSMs in the other controllers of the router to evaluate the error propagation, and also to apply fault tolerance techniques to protect the router's internal structures. We also intend to use the router in an actual NoC system for more complete results. The source code of the router presented here is available at [33].

Acknowledgment

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001 and by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) – Processes 315287/2018-7 and 436982/2018-8.

References

[1] N. E. Jerger and L.-S. Peh, "On-chip networks," *Synthesis Lectures on Computer Architecture*, vol. 4, no. 1, pp. 1–141, 2009.

[2] C. A. Zeferino and A. A. Susin, "Socin: a parametric and scalable network-on-chip," in *Integrated Circuits and Systems Design, 2003. SBCCI 2003. Proc. 16th Symp. on*. IEEE, 2003, pp. 169–174.

[3] D. Bertozzi, "The data-link layer in noc design," *Micheli, G.; Benini, L." Networks on Chips: Technology and Tools", New York: Morgan Kaufmann Publishers*, 2006.

[4] D. J. Sorin, "Fault tolerant computer architecture," *Synthesis Lectures on Computer Architecture*, vol. 4, no. 1, pp. 1–104, 2009.

[5] K. H. Walters, S. H. Gerez, G. J. Smit, S. Baillou, G. K. Rauwerda, and R. Trautner, "Multicore soc for on-board payload signal processing," in *Adaptive Hardware and Systems (AHS), 2011 NASA/ESA Conf. on*. IEEE, 2011, pp. 17–21.

[6] P. Guerrier and A. Greiner, "A scalable architecture for system-on-chip interconnections," in *Proc. of the Sophia Antipolis Forum on MicroElectronics*, 1999, pp. 90–93.

[7] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Design Automation Conf., 2001. Proc.* IEEE, 2001, pp. 684–689.

[8] L. Benini and G. De Micheli, "Networks on chip: a new paradigm for systems on chip design," in *Design, Automation and Test in Europe Conf. and Exhibition, 2002. Proc.* IEEE, 2002, pp. 418–419.

[9] A. Jantsch, H. Tenhunen *et al.*, *Networks on chip*. Springer, 2003, vol. 396.

[10] J. Duato, S. Yalamanchili, and L. M. Ni, *Interconnection networks: an engineering approach*. Morgan Kaufmann, 2003.

[11] W. J. Dally and B. P. Towles, *Principles and practices of interconnection networks*. Elsevier, 2004.

[12] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Computing Surveys (CSUR)*, vol. 38, no. 1, p. 1, 2006.
[13] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. on dependable and secure computing*, vol. 1, no. 1, pp. 11–33, 2004.
[14] C. Constantinescu, "Trends and challenges in vlsi circuit reliability," *IEEE micro*, vol. 23, no. 4, pp. 14–19, 2003.
[15] T. F. Pereira, D. R. de Melo, E. A. Bezerra, and C. A. Zeferino, "Mechanisms to provide fault tolerance to a network-on-chip," *IEEE Latin America Trans.*, vol. 15, no. 6, pp. 1034–1042, 2017.
[16] H. Bokhari, H. Javaid, M. Shafique, J. Henkel, and S. Parameswaran, "Supernet: multimode interconnect architecture for manycore chips," in *Design Automation Conf. (DAC), 2015 52nd ACM/EDAC/IEEE*. IEEE, 2015, pp. 1–6.
[17] H. Zimmer and A. Jantsch, "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, 2003. First IEEE/ACM/IFIP Int. Conf. on*. IEEE, 2003, pp. 188–193.
[18] D. Park, C. Nicopoulos, J. Kim, N. Vijaykrishnan, and C. R. Das, "Exploring fault-tolerant network-on-chip architectures," in *Dependable Systems and Networks, 2006. DSN 2006. Int. Conf. on*. IEEE, 2006, pp. 93–104.
[19] A. P. Frantz, F. L. Kastensmidt, L. Carro, and E. Cota, "Dependable network-on-chip router able to simultaneously tolerate soft errors and crosstalk," in *Test Conf., 2006. ITC'06. IEEE Int.* IEEE, 2006, pp. 1–9.
[20] A. Kohler and M. Radetzki, "Fault-tolerant architecture and deflection routing for degradable noc switches," in *Networks-on-Chip, 2009. NoCS 2009. 3rd ACM/IEEE Int. Symp. on*. IEEE, 2009, pp. 22–31.
[21] M. Pirretti, G. M. Link, R. R. Brooks, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "Fault tolerant algorithms for network-on-chip interconnect," in *VLSI, 2004. Proc. IEEE Computer society Annual Symp. on*. IEEE, 2004, pp. 46–51.
[22] É. Cota, F. L. Kastensmidt, M. Cassel, M. Herve, P. Almeida, P. Meirelles, A. Amory, and M. Lubaszewski, "A high-fault-coverage approach for the test of data, control and handshake interconnects in mesh networks-on-chip," *IEEE Trans. on Computers*, vol. 57, no. 9, pp. 1202–1215, 2008.
[23] S. Tosun, V. B. Ajabshir, O. Mercanoglu, and O. Ozturk, "Fault-tolerant topology generation method for application-specific network-on-chips," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 9, pp. 1495–1508, 2015.
[24] B. Bhowmik, J. K. Deka, and S. Biswas, "Towards a scalable test solution for the analysis of interconnect shorts in on-chip networks," in *Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2016 IEEE 24th Int. Symp. on*. IEEE, 2016, pp. 394–399.
[25] I. Loi, S. Mitra, T. H. Lee, S. Fujita, and L. Benini, "A low-overhead fault tolerance scheme for tsv-based 3d network on chip links," in *Proc. of the 2008 IEEE/ACM Int. Conf. on Computer-Aided Design*. IEEE Press, 2008, pp. 598–602.
[26] I. Loi, F. Angiolini, S. Fujita, S. Mitra, and L. Benini, "Characterization and implementation of fault-tolerant vertical links for 3-d networks-on-chip," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 1, pp. 124–134, 2011.
[27] A.-M. Rahmani, P. Liljeberg, K. Latif, J. Plosila, K. R. Vaddina, and H. Tenhunen, "Congestion aware, fault tolerant, and thermally efficient inter-layer communication scheme for hybrid noc-bus 3d architectures," in *Networks on Chip (NoCS), 2011 Fifth IEEE/ACM Int. Symp. on*. IEEE, 2011, pp. 65–72.
[28] A. Eghbal, P. M. Yaghini, N. Bagherzadeh, and M. Khayambashi, "Analytical fault tolerance assessment and metrics for tsv-based 3d network-on-chip," *IEEE Trans. on Computers*, vol. 64, no. 12, pp. 3591–3604, 2015.
[29] C. Feng, Z. Lu, A. Jantsch, M. Zhang, and Z. Xing, "Addressing transient and permanent faults in noc with efficient fault-tolerant deflection router," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 6, pp. 1053–1066, 2013.
[30] S. Pasricha and Y. Zou, "A low overhead fault tolerant routing scheme for 3d networks-on-chip," in *Quality Electronic Design (ISQED), 2011 12th Int. Symp. on*. IEEE, 2011, pp. 1–8.
[31] H. Ziade, R. A. Ayoubi, R. Velazco *et al.*, "A survey on fault injection techniques," *Int. Arab J. Inf. Technol.*, vol. 1, no. 2, pp. 171–186, 2004.
[32] R. Travessini, P. R. Villa, F. L. Vargas, and E. A. Bezerra, "Processor core profiling for seu effect analysis," in *Test Symposium (LATS), 2018 IEEE 19th Latin-American*. IEEE, 2018, pp. 1–6.
[33] D. R. Melo, C. A. Zeferino, L. Dilillo, and E. A. Bezerra, "XARC - eXtensible ARChitecture," 2019. [Online]. Available: https://xarc.org/