Assessing the Reliability of a Network-on-Chip through Physical Validation

Gustavo S. Mafra, Thiago H. Rausch, Douglas Almeida dos Santos, Luigi Dilillo, Eduardo Augusto Bezerra, Douglas Rossi de Melo

Title: Assessing the Reliability of a Network-on-Chip through Physical Validation

Author(s): Gustavo S. Mafra, Thiago H. Rausch, Douglas A. Santos, Luigi Dilillo, Eduardo A. Bezerra, and Douglas R. Melo

Document version: Post-print version (Final draft)

Please cite the original version:
G. S. Mafra et al., ”Assessing the Reliability of a Network-on-Chip through Physical Validation,” 2022 5th IAA Latin American CubeSat Workshop and 3rd IAA Latin American Symposium on Small Satellites (IAA-LA), 2022, pp. 1-10.

Assessing the Reliability of a Network-on-Chip through Physical Validation

Gustavo S. Mafra[(1)], **Thiago H. Rausch**[(1)], **Douglas A. Santos**[(2)], **Luigi Dilillo**[(2)], **Eduardo A. Bezerra**[(3)], **Douglas R. Melo**[(1)]
[(1)] *LEDS, University of Vale do Itajaí, Brazil*
[(2)] *LIRMM, University of Montpellier, CNRS, France*
[(3)] *SpaceLab, Federal University of Santa Catarina, Brazil*

Keywords: *Networks-on-Chip, Fault Tolerance, Neutron Radiation.*

Abstract

With the increased use of embedded computing and the number of cores in integrated systems, communication architectures more robust than the bus became necessary. Networks-on-chip are a solution proposed by academia and industry that makes systems more scalable with increased cores. The increase in cores is also observed in systems for use in critical environments, such as in space applications. However, architectures aimed at these applications suffer from radiation problems and extreme temperatures. Due to the flexibility and availability of logic elements, programmable logic devices are an attractive solution for developing embedded systems for space applications. However, these devices are sensitive to radiation, which makes fault tolerance techniques a requirement for their effective use in space and verification at the physical level since their circuit tends to suffer from radiation effects that cause error propagation. In this context, this work seeks to evaluate the reliability of a Network-on-Chip through physical prototyping tests. The solution employs traffic generators and meters to verify the correct functioning of the network with the complement traffic pattern. Thus, it becomes possible to certify that the behavior of the network in a physical device corresponds to the same presented in a simulation model. The network was first prototyped in the FPGA device Xilinx Zynq-7000 to obtain metrics from the network connected to the traffic components, then it was prototyped at the M2S010 FPGA for testing in a particle accelerator. The results obtained from the particle accelerator tests converge with those obtained in simulation, allowing an initial validation of the network's reliability.

1. Introduction

Currently, there is as increase necessity for devices with higher performance, but smaller area and with lower energy consumption [1]. Thus, components such as memories, controllers, and processors had to be miniaturized, making it possible to create Systems-on-Chip (SoCs). Recently, SoCs are incorporating a growing number of processing cores. In systems that integrate multiple cores, architectures more robust than the bus became necessary due mainly to the bus' lack of parallelism. Therefore, a solution proposed by academia and industry is the utilization of Networks-on-Chip (NoCs), which use integrated routers to make the systems more scalable by supporting a higher number of cores [2].

Email addresses: gustavo_mafra@edu.univali.br (**Gustavo S. Mafra**), drm@univali.br (**Douglas R. Melo**)

The continuous increase in the use of embedded computing and the number of cores in integrated systems is also observed in critical environments such as space and avionics applications. However, they suffer from the hostility of these environments, which subject electronic devices to effects from radiation and extreme temperatures. This hostility leads to temporary, permanent damages or intermittent failures that affect the behavior of computer systems [3].

Field Programmable Gate Array (FPGA) devices are an attractive solution for developing SoCs composed of multiple cores due to their high flexibility and availability of logic elements. However, as these devices are sensitive to radiation, the provision of fault tolerance techniques is required for their effective use in space applications [4].

Regarding the integration of multiple cores for critical environments, the work [5] proposes an NoC aiming at the physical implementation in a programmable logic device. The network uses finite state machines (FSMs) in the controller unit of the input flow regulation, packet routing, and channel arbitration structures, and applies triple modular redundancy (TMR) as a protection technique for these controllers. However, in that work, the architecture's reliability was validated only through simulation, not having obtained results in physical prototyping at the time.

In this context, we propose to prototype an NoC system to evaluate its reliability in harsh environments. We validated this system with a physical prototyping test in a particle accelerator, which exposed the system to neutron radiation.

2. Dependable Network-on-Chip

The NoC architecture developed by [5] aimed at space applications by implementing fault tolerance in its internal architecture elements. The designed architecture implemented triple modular redundancy in controllers and Hamming error correction code in buffers to ensure network reliability in critical environments. In addition, a set of architecture parameters were defined to evaluate the routers controllers' different implementation approaches, constituting Moore or Mealy state machines, in a standard or protected version.

The architecture of the router was designed with a focus on regularity, flexibility, and low area overhead. To fulfill these requirements, the authors used the wormhole switching technique since it provides lower latency for less cost. They also used input buffers capable of storing *n-words*. In addition, the router was conceived to integrate 2-D mesh topology networks.

The router is composed of five data ports, named *Local*, *North*, *East*, *South*, and *West*. The *Local* port is the terminal at which a processing core is attached, and the other ports connect the router to its neighbors. The input channels comprise controllers responsible for input flow regulation and packet routing, while the output channels include controllers performing channel arbitration and output flow regulation.

The controllers responsible for flow regulation implement a 4-stage handshake protocol for receiving and sending packet *flits* (a *flit* is the smallest piece of data over which is performed the flow regulation). The routing controller runs the *XY* algorithm to request an output channel to forward an incoming packet. The arbitration controller consists of a Round-Robin arbiter that schedules the use of the output channel by the packets in the router input channels.

The authors evaluated the architecture described using simulation, which consisted of simulating the traffic generation and the NoC operation with injection of single-event upset (SEU) faults in the registers using built-in commands of the ModelSim simulator that force bit-flips. The evaluation comprised measuring the throughput and error rate for different FSM combinations in the controllers of flow regulation, routing, and arbitration, resulting in eight different architectural configurations for each router evaluated in protected and standard forms.

3. Proposed evaluation platform

3.1. Test architecture

This work presents a solution for validating the reliability of the Network-on-Chip described above, using traffic generation (TG) and traffic meter (TM) components. Each router is connected to a TG component, responsible for generating and sending data packets to the network, and to a TM component that receives and validates the data by comparing it to the expected data. Figure 1 represents the structure adopted for the architecture validation. In this work, we used a 4x4 network with an 8-bit data width to perform the tests.

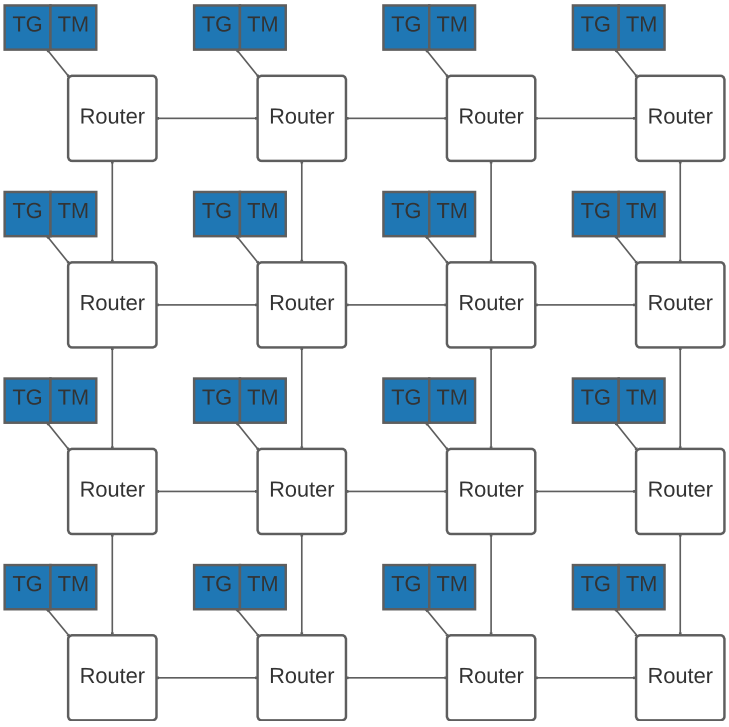

Figure 1: Proposed test architecture.

3.2. Traffic components

As previously described, the NoC flow regulation controller implements a 4-stage handshake protocol for receiving and sending packet *flits*. Thus, in order to send and receive data, we implemented a finite state machine (FSM) to send and receive the requests. Since the router was designed to enable the validation of different types of FSM (Moore or Mealy) for each controller, the components used for the network validation also allow FSM type switching.

Furthermore, the routing controller runs the *XY* algorithm to request an output channel to forward an incoming packet. The implemented traffic generator uses different traffic models to generate the packet header, constituted by the *X* and *Y* addresses of the receiver router. To evaluate the operation and reliability of the architecture, the complement traffic pattern was used.

Figure 2 shows the packet format used for the verification. The router's frame control is performed with one bit of data, and for sending a packet, only one of the *flits* will be the header, and only one will be the trailer, the others are payload *flits*. Both the header and the last payload *flit* (trailer) use '1' as the frame bit, while the regular payloads use '0'. To ensure the functioning of sending the packets, this principle presented was used in the generation of each *flit*, thus avoiding problems with misrouting or incomplete packets.

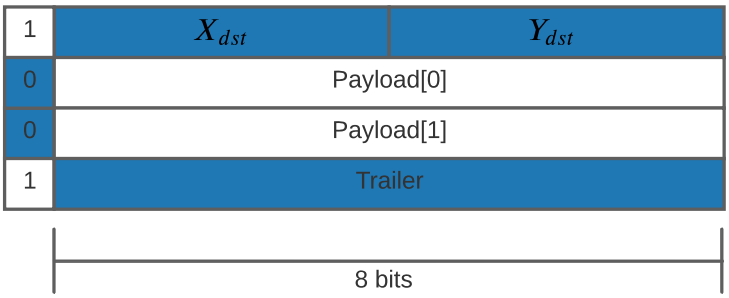

Figure 2: Packet format for verification.

The payload *flits* represent the useful data that will be forwarded by the processor cores connected to the NoC. Thus, in order to use random data for each message sent, the Linear-feedback shift register (LFSR) technique was used, given its application to generate pseudo-random numbers [6]. Thus, for each router, an initial value for the LFSR was defined based on its *XY* position in the network, this value is called a seed. Because the register has a finite number of possible states, it must eventually enter a repeating cycle. However, an LFSR with a well-chosen feedback function can produce a sequence of bits that appears random and which has a very long cycle [7].

To validate the received data on the router *Local* port, the traffic meter component performs a four-state handshake in order to work properly with the NoC flow regulation controller. The *flit* received by the component is compared with the expected *flit*, thus checking for possible errors in the received message.

The expected *flit* is generated using the LFSR technique. Thus the same polynomial is used in both components, ensuring that the sequence generated in the traffic meter component is the same as the traffic generator component. The seed used by the meter is defined with the first payload *flit* received, thereby the data comparison is performed only after receiving the first data packet.

The meter component is also responsible for measuring and storing the number of received packets and errors. Errors are incorrect data, such as bit-flips in the received *flit*. The number of received packets and identified errors from each router are stored in registers, which are accessible for further analysis.

4. Materials and methods

The traffic generators and traffic meter components were described using VHDL on a 4x4 NoC to generate and meter traffic through the *Local* port. Subsequently, the proposed test architecture was synthesized using Xilinx Vivado 2020 for cost and performance analysis. We selected the Zedboard Zynq-7000 development kit as the target device, and the synthesis tool's optimization options were kept at default.

To validate the generator and meter components implementation, we compared the meter's output to an equivalent implementation in testbench that used the same traffic pattern and traffic data. To generate traffic for the testbench, we used text files provided by a Python script that applied the same LFSR seed as the generator component, then compared both outputs via another Python script to check for mismatches.

4.1. Particle accelerator test

The hardware implementation was tested using neutron particles and the Microsemi SMF2000 FPGA, connected through a UART interface to output the data to a host computer. The tests developed in a particle accelerator seek to analyze the functioning of the network against neutron radiation. In this way, we carried out tests at Rutherford Appleton Laboratories – UK, more specifically at ChipIr beamline [8].

ChipIr, for Chip Irradiation, is an instrument designed to mimic the atmospheric neutron environment. Atmospheric neutrons are a major cause of Single Event Effects (SEE), which disrupt the correct operation of microelectronics devices and systems [8]. This instrument irradiates the device under test (DUT) with a flux up to 10^9 times greater than the natural radiation environment [9]. This high flux enables accelerated testing of electronic devices. According to [10], the average flux provided by this beamline is $5.6 \times 10^6 n/cm^2/s$ for energies above 10 MeV.

The test was set up so that, every 60 seconds, a reset signal is sent to the system to restart the test, so any possible errors wouldn't last over the run time. The output data received by the host computer consists of each router error and received packet counter, and also when the reset occurred. To analyze the data a Python script was used to count the errors and the number of received packets during the run period, and plot the values in two graphs for each router.

4.2. Traffic pattern

The work [11] presents different types of traffic patterns for the analysis of operation and performance in Networks-on-Chip. In this work, we chose to use the uniform traffic pattern named complement, where all routers have the same probability of being destinations. Since, in the study of data communication networks, uniform distribution is the most frequently used for NoC evaluations.

Figure 3 shows the complement traffic pattern, where each node is represented by its coordinates in binary format a_{n-1}, a_{n-2}, ..., a_1, a_0, and sends data to the node $\overline{a}_{n-1}$, $\overline{a}_{n-2}$, ..., $\overline{a}_1$, $\overline{a}_0$. This operation describes an inversion of the values of all bits in the address.

Origin	Destination
[0,0]=[0000]	[3,3]=[1111]
[0,1]=[0001]	[3,2]=[1110]
[0,2]=[0010]	[3,1]=[1101]
[0,3]=[0011]	[3,0]=[1100]
[1,0]=[0100]	[2,3]=[1011]
[1,1]=[0101]	[2,2]=[1010]
[1,2]=[0110]	[2,1]=[1001]
[1,3]=[0111]	[2,0]=[1000]
[2,0]=[1000]	[1,3]=[0111]
[2,1]=[1001]	[1,2]=[0110]
[2,2]=[1010]	[1,1]=[0101]
[2,3]=[1011]	[1,0]=[0100]
[3,0]=[1100]	[0,3]=[0011]
[3,1]=[1101]	[0,2]=[0010]
[3,2]=[1110]	[0,1]=[0001]
[3,3]=[1111]	[0,0]=[0000]

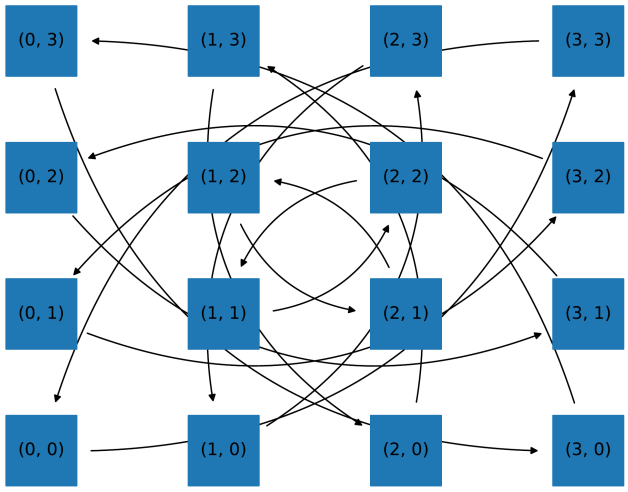

Figure 3: Complement traffic pattern.

5. Results

Table 1 presents the synthesis results for the NoC and traffic components, with the Zedboard Zynq-7000 development kit as the target device. As reported by the authors in [5], we see that the Mealy-based controllers require fewer FFs because the FSMs encode few states. Furthermore, using Mealy controllers implies a longer critical path and a lower operating frequency.

Table 1: NoC synthesis results with the traffic components on Zynq-7000.

Controllers FSM	LUTs	FFs	Fmax (MHz)
Moore	4386	4207	225.94
Mealy	3760	3841	160.78

Table 2 shows the increase in the use of LUTs and FFs, and the decrease in maximum frequency, caused by the insertion of traffic generator and traffic meter components in the *Local* port of the NoC for the physical tests.

Table 2: NoC synthesis overhead with traffic components.

Controllers FSM	LUTs	FFs	Fmax (MHz)
Moore	17.59%	19.00%	-29.12%
Mealy	21.41%	21.43%	-22.73%

The simulation with error insertion demonstrated equivalent results for the components described in the hardware and the testbench. It was observed that errors inserted into the network are measured by the meter's components, which subsequently increment the error counter of the router where the error was observed. Furthermore, the packet counter presented the expected operation, incrementing the counter whenever the *flit* trailer of a packet was received. With this validation, it was possible to start the test campaigns in particle accelerators.

5.1. Results obtained in particle accelerator

For the particle accelerator test, we used the M2S010 FPGA device from the Smartfusion2 family by Microsemi. This FPGA was chosen due to its reliable FPGA configuration memory, which is resistant to neutron radiation [12, 13]. Table 3 presents the synthesis results for the NoC with traffic components, and also the UART and the FSM used to control the experiment.

Table 3: NoC synthesis results with the traffic components on M2S010 FPGA.

Controllers FSM	LUTs	FFs	Fmax (MHz)
Moore	7074	4229	136.57
Mealy	7345	4217	79.30

In the first test campaign, we tested the Network-on-Chip with its controllers on Moore against neutron radiation. Furthermore, in the second test campaign, we redid the tests with the NoC using the controllers on Mealy. In both experiments, we estimate the delivery of packages and the incident of errors in 60 runs of 60 seconds.

5.1.1. Network controllers on Moore

Figure 4 demonstrates the router (0,0), which we consider to have an ideal operation since it has a very small variation in data throughput during the execution period, and errors weren't measured in the received packets. The variability in the number of received packets is observed in all routers, indicating normal variations due to the peripherals used to reset each run.

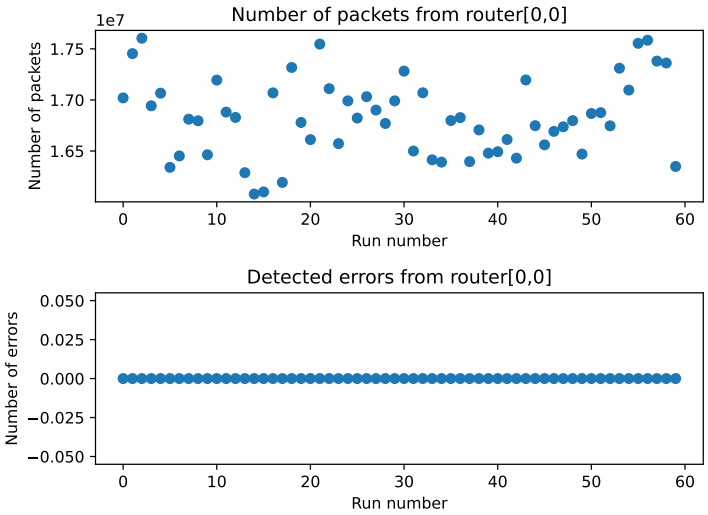

Figure 4: Router (0,0) results considered an ideal operation.

In Figure 5, we observe the occurrence of errors in the router (3,2). The error occurs in run 5, and errors are measured until the reset of this run. This error characterizes a bit-flip in the data buffer, thus affecting the data generated by the LFSR, causing a difference in the comparator present in the traffic meter component.

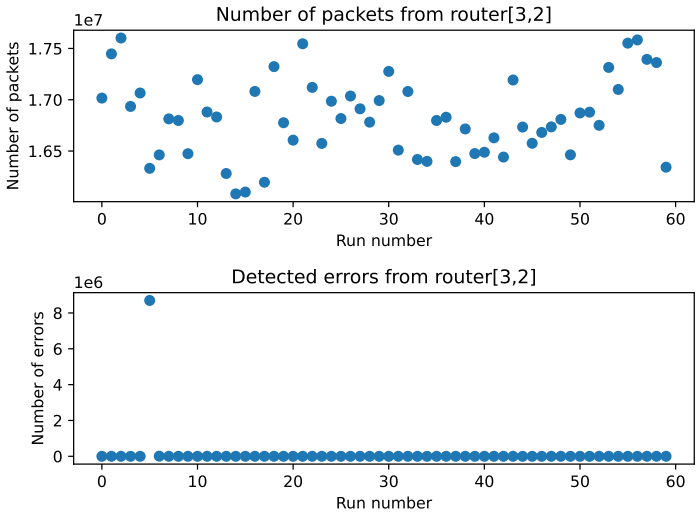

Figure 5: Router (3,2) results show errors in received data.

In Figure 6, it is possible to see misroutings occurring in run 6. This observation is based on the low number of packets received by the router in that run, indicating that the packets were lost before delivery. This decrease was not noticed on other routers during these runs, indicating that the problem occurred only when forwarding packets addressed to that router.

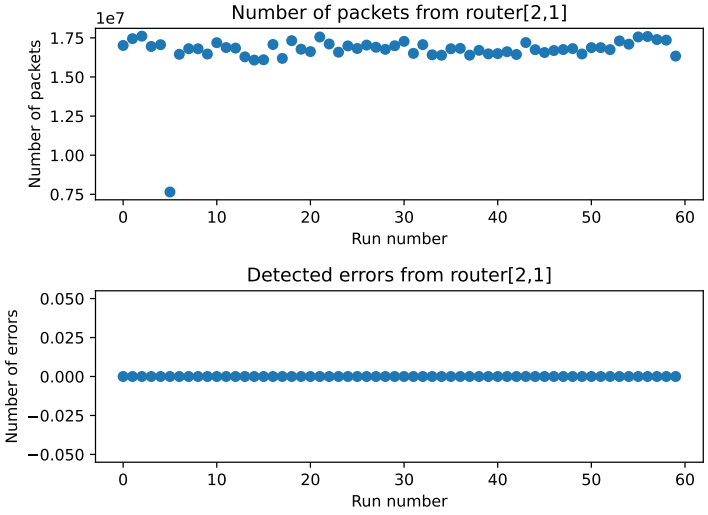

Figure 6: Router (2,1) results demonstrate misrouting.

5.1.2. Network controllers on Mealy

Figure 7 demonstrates the router (0,0) that we consider to have an ideal operation with Mealy controllers, i.e., errors weren't measured in the received packets. In addition, it is possible to observe the variation of packets received in each run. However, as previously mentioned, this variation occurs because the peripherals used to perform the resets interfere with the duration of each run.

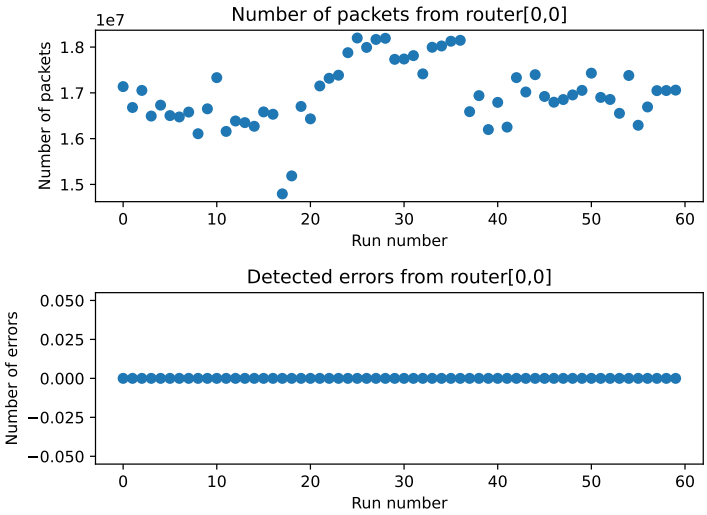

Figure 7: Router (0,0) results considered an ideal operation.

In Figure 8, it is possible to observe the occurrence of errors in the router (1,2). The error occurs in run number 2, and errors are measured until the reset of this run, the same type of error found with Moore controllers. Thus, we found that with Mealy controllers, errors occur less frequently than with the Moore FSM, but with the same consequences when they appear.

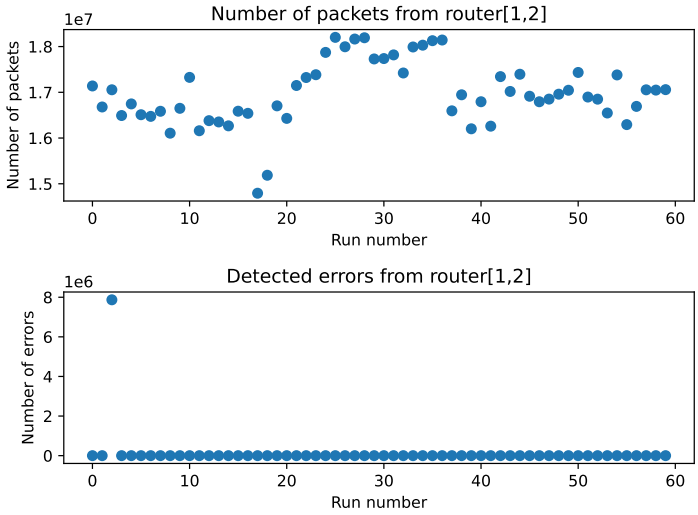

Figure 8: Router (1,2) results show errors in received data.

Tests with the controllers on Mealy didn't present any misrouting errors in the analyzed samples. In addition, errors were shown to be exceedingly rare in both Mealy and Moore controllers, but with the use of controllers on Mealy, there was a decrease in the occurrence of errors during the tests.

5.2. Discussion

In the 500 runs analyzed, 11 errors were observed using the controllers on Moore, 3 in the payload *flits*, and 8 in the header *flits*, which characterizes a misrouting. However, using controllers on Mealy only 1 error was observed on the packet payload, demonstrating a 91% increase in the network reliability.

The results obtained in physical prototyping and testing in particle accelerator converge with those obtained previously in simulation by the authors in the work [5], demonstrating that Mealy is the more reliable choice for the network controllers. However, further tests need to be conducted to accurately verify the reliability increase in the network using Mealy controllers.

6. Conclusion

The tests using the complement traffic pattern validate the operation of the NoC, thus enabling an understanding of the normal operating behavior of the network. Thereby, with the particle accelerator tests, it was possible to verify the behavior of the architecture in critical environments. The network proved to be tolerant most of the time, but it was also detected errors caused by the incidence of particles in the FPGA.

In future work, we intend to improve the evaluation in a particle accelerator by using other traffic models, in addition to a new test campaign with the reliable version of the network, in order to verify its operation and reliability in a hostile environment.

Acknowledgments

This work was supported by the Foundation for Support of Research and Innovation, Santa Catarina (FAPESC-2021TR001907), the Region d'Occitanie and the École Doctorale I2S from the University of Montpellier (20007368/ALDOCT-000932), and the RADNEXT EU project (Horizon 2020 - Grant Agreement no. 101008126).

References

[1] S. Pasricha, N. Dutt, On-chip communication architectures: system on chip interconnect, Elsevier, 2008.

[2] D. Jose, S. Yalamanchili, L. Ni, Interconnection Networks: an Engineering Approach, Morgan Kaufmann, 2003.

[3] M. Yang, G. Hua, Y. Feng, J. Gong, Fault-tolerance techniques for spacecraft control computers, John Wiley & Sons, 2017.

[4] F. Kastensmidt, P. Rech, FPGAs and Parallel Architectures for Aerospace Applications, Springer, 2016.

[5] D. Melo, C. Zeferino, L. Dilillo, E. Bezerra, Maximizing the inner resilience of a network-on-chip through router controllers design, Sensors (2019).

[6] M. Luby, Pseudorandomness and Cryptographic Applications, Princeton University Press, 1996.

[7] A. K. Panda, P. Rajput, B. Shukla, Fpga implementation of 8, 16 and 32 bit lfsr with maximum length feedback polynomial using vhdl, International Conference on Communication Systems and Network Technologies (2012).

[8] C. Cazzaniga, C. D. Frost, Progress of the scientific commissioning of a fast neutron beamline for chip irradiation, Journal of Physics: Conference Series (2018).

[9] C. Cazzaniga, M. Bagatin, S. Gerardin, A. Costantino, C. D. Frost, First tests of a new facility for device-level, board-level and system-level neutron irradiation of microelectronics, IEEE Transactions on Emerging Topics in Computing 9 (2021) 104–108.

[10] D. Chiesa, M. Nastasi, C. Cazzaniga, M. Rebai, L. Arcidiacono, E. Previtali, G. Gorini, C. D. Frost, Measurement of the neutron flux at spallation sources using multi-foil activation, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 902 (2018) 14–24.

[11] L. P. Tedesco, Uma proposta para geração de tráfego e avaliação de desempenho para NoCs, Master's thesis, Pontifícia Universidade Católica do Rio Grande do Sul, 2005.

[12] Microsemi, IGLOO2 and SmartFusion2 Datasheet Version DS0128, 2018. Rev. 12.

[13] D. Dsilva, J.-J. Wang, N. Rezzak, N. Jat, Neutron SEE testing of the 65nm SmartFusion2 flash-based FPGA, in: 2015 IEEE Radiation Effects Data Workshop (REDW), pp. 1–5.