Combination of Strategies to Improve the Yield in Networks-on-Chip Links

[1]Anelise Kologeski, [2]Caroline Concatto, [1,2]Fernanda Lima Kastensmidt, [2]Luigi Carro

{alkologeski, cconcatto, fglima, carro}@inf.ufrgs.br

[1]Programa de Pós-Graduação em Microeletrônica (PGMICRO-UFRGS)
[2] Programa de Pós-Graduação em Computação (PPGC-UFRGS)

Abstract

The use of fault-tolerant mechanisms embedded in Network-on-Chips (NoCs) becomes essential to ensure connectivity in the presence of massive defects. Information from the manufacturing test can help to adapt the embedded fault-tolerant techniques in according to the location of defects in the NoC. In this paper, three strategies are used to cope with massive defects in NoC links. In this way, by combine the right strategies, it is possible to minimize performance and power overheads because only the defect regions will use the fault-tolerant techniques.

1. Introduction

The use of fault-tolerant methods in such architectures has become crucial to increase yield, due to the huge amount of defects that comes from the aggressive technology scaling reaching the limits of silicon. As the number of defects increases, fault-tolerant techniques are expected to cope with multiple faults at the same time, since that the defect rate can reach a significant number of wires in channel links.

A Network-on-Chip (NoC) offers better scalability and performance than a traditional bus, and it has been used as alternative architecture to provide communication inside of Systems-on-Chip (SoCs). Defects in NoC can happen in all components: routers, links, processing elements and network interfaces. Related techniques to mitigate faults in the links usually are based on parity check, retransmission, data splitting and adaptive routing [1-3]. Normally they assume a single fault per link scenario, and a traditional example of technique considered is single-error correction and double-error detection (SEC/DEC) is the Hamming code. It is placed in each routing to encode and decode the data at each link. This technique shows large area overhead because extra wires are required in every link for the parity bits. Moreover, it causes performance penalty and excessive power dissipation, because encoding and decoding blocks are placed in the critical path, protecting the link even when no defect is observed.

The model of a single fault per link is not valid any longer, since multiple manufacture defects are more common to be observed in locations close to each other as clusters of defects in nanometer technologies [4]. The problem is that the use of several embedded fault-tolerant techniques to cope with multiple faults in links significantly enlarges the overheads in area, performance and power. In [3], the authors propose an adaptive routing strategy to cope with multiple defects in each link and multiple faulty links. The adaptive routing provides minimal changes in the XY path and can cope with faults that affect up to 100% of wires in a single link, once that the faulty link can be completed avoided by using a different path to forward the packets. However, this technique has some limitations, as there is a limit of faulty links that can be considered in order to find an alternative fault free path to ensure the NoC connectivity. In order to ensure the best scenario to the connectivity, and to improve the number of faulty links that can be handled by the NoC, data splitting is used [5]. The combination of [3] and [5] can cope with more than 50% of faulty wires distributed in several links. When both strategies cannot find a solution for some faulty case, the remapping strategy might be employed. Our proposal needs off-line testing and diagnosis to adjust the techniques in according to the type, number and location of defects in the NoC links. Because defects are expected to be in clusters, it is possible to minimize the area, performance and power overheads turning on the fault tolerant mechanisms only in the defective links. In this way, the costs are not seen in all parts of the architecture.

2. Related Works

Recent related works have proposed solutions to tolerate single faults in NoC links, but these approaches are not enough to ensure reliability and yield. The technique proposed in [1] uses parity check, data splitting and retransmission of data to protect the links against faults. The parity check is used to discover a faulty link, and in the presence of faults, the erroneous half of the data is doubled and retransmitted. The main disadvantage of this method is the use of extra wires and the area overhead. The work proposed in [2] is similar, however it uses Hamming code and leads to a final area four times larger than the no-protected router

and the latency is increased around four times. Both works can cope with multiple faulty links, but not with multiple faults in each link. Parity check and Hamming also were used in [6] to protect the links. Voltage scaling was used and the channel was duplicated. Thus, [6] can protect the NoC with an area overhead of 22% and 200% more links.

The work in [7] proposes to use partially faulty links when the traffic in the network is high. The idea is to distribute the traffic uniformly on links. The links capacity can be split in 25%, 50%, 75%, and 100%, according to the faults in the link. This proposal has a low power penalty and an area overhead around 15% to 21%. However, [7] considers that all faults are concentrated within the same group of wires, nevertheless faults can be distributed among the link. In [8], redundancy was used in some components, but the connectivity kept is low when multiple faults happen.

The works presented in [9] and [10] use adaptive routing to avoid faulty links and routers, which implies in a relative low latency overhead. However, they use virtual channels and memory tables to avoid deadlock in the network, which are normally synonymous of area overhead and power consumption. In [3] the authors propose a partially adaptive routing strategy to cope faulty links based on the minimal change in the XY path. Consequently, virtual channels and tables are not used, and the technique in [3] has a smaller area overhead (1%). However, because the routing is only partially adaptive, it is not always possible to find an alternative faulty-free path, especially in the presence of multiple faults. Results in [3] have shown that by using only adaptive routing, 34% of faulty links are unprotected in a 12-cores NoC.

In [11] and [12] the combination of mapping and adaptive routing increases the reliability in NoCs. Both works present a mapping strategy that concurrently takes into account the application core graph, the fault probability in the links and the routing. The difference between [11] and [12] is the mapping algorithm. But, both works cannot solve the problem caused by faulty links between core and router, reducing their efficiency to 65% in a 12-cores NoC. As our work uses the information about the fault location together with the chosen fault tolerant solution, we can change the application mapping and expect a better behavior, while [11-12] make several computations in the design phase to find the best mapping for a given fault probability.

3. The Strategies

We propose a technique that copes with single and multiple permanent faults in NoC links, to ensure yield, by using adaptive routing, data splitting, and remapping. The combination of these approaches avoids the need of additional wires in the links, and minimizes additional hardware in the critical path. Consequently, minimal impacts are expected to make a NoC tolerate multiple defects in the links.

The proposed technique was implemented in the case-study SoCIN NoC [13] with 2D-torus topology. The router architecture was implemented in VHDL, and each router can be connected to four neighboring routers with two unidirectional channel links and in its local port has a processor element. The architecture uses the wormhole switching and deadlock-free XY-routing. Each input channel port has a buffer with 4 slots, and each router presents a latency of three clock cycles to transmit the packet.

In order to cope with defects, first the proposed fault-tolerant method attempts to use alternative paths to avoid the faulty-links [3]. For that, each router is configured with the manufacturing test information about faulty-links. An additional 10-bit register is added in each router with the test result information. So, each router knows if one or more of its links (L, N, S, W, and E for input and output channels) are faulty. In the 2D-torus topology of size mxn, a packet has two possible routes in the same dimension: it may go k steps to one way (positive) or m-k (or n-k) steps to the other way (negative). However, a packet travels no more than m-1 or n-1 steps from source to destination when m or n is odd, or only m or n steps when they are even. If an output channel is indicated as faulty, an alternative path replaces the original in the header and the packet is routed normally through the faulty-free path. As a consequence, the router dynamically changes the target address in the header in a packet when the original address intends to use a faulty link

Although each flit in the message is re-routed, the impact on the overall message is low, because of the wormhole switching approach. The area overhead is only 1%, and the power consumption is insignificant, because the average opposite path will not be much larger than the original path.

However, there is a set of faults for which the adaptive routing cannot cope with: faults in paths without a redundant path (for example, between core and router) and faults affecting the input or output channels in the same direction of a router, blocking the path through the router. For these situations, the faulty channel cannot be discarded, and another strategy must be used. So, for simplicity, we choose to use the data splitting.

The data splitting divides the flit in two parts, and two cycles are required to send the information, and as a penalty, the latency can be increased. Multiplexers are placed at the inputs and outputs of each channel to select the faulty-free wires will be used to transmit the data. Each multiplexer is configured by a register, based on the manufacturing test. All the flits in the packet are joined when they reach the final destination, ensuring that the latency will be the same, regardless of the number of faulty channels in the path.

When the data splitting is being used, only the routers of the end and beginning of each faulty link are using the strategy. But, when faults do not happen, the resources of data splitting can be turned off to save

power [14-15]. If more than 50% of the wires are faulty, then the strategy cannot be used and a third solution must be employed to ensure the right communication: the remapping of the tasks.

The remapping allows processing elements with low rates and low communication stay near to the faulty region, to minimize the impact in the communication time. To obtain the delay results for each element, the number of packets is multiplied by the bandwidth and divided by the rate, as presented in equation 1. Based on this result, the element with lower delay can be placed to the faulty position, to decrease the impact of the data splitting.

$$\Delta Delay = (\#\_packets * bandwidth) / injection\_rate \quad (1)$$

A simply remapping option can be achieved based on the original mapping choose in design time. The best mapping can provide more 3 possibilities of mapping keeping the original requirements: we can use the vertical mirroring, the horizontal mirroring and to combine both types of mirroring, as present in fig. 1, that considers one faulty link between the router 1 and the processor element. So, considering a specific traffic condition (hipotethical) we can see that with the horizontal mirroring is possible to obtain the minimal time communication when the faulty link is used in the network. Normally, the remap strategy needs some redundancy in the network. In this work, we consider that each element is placed inside of an identical processor, e because that, they can change of position very easily.

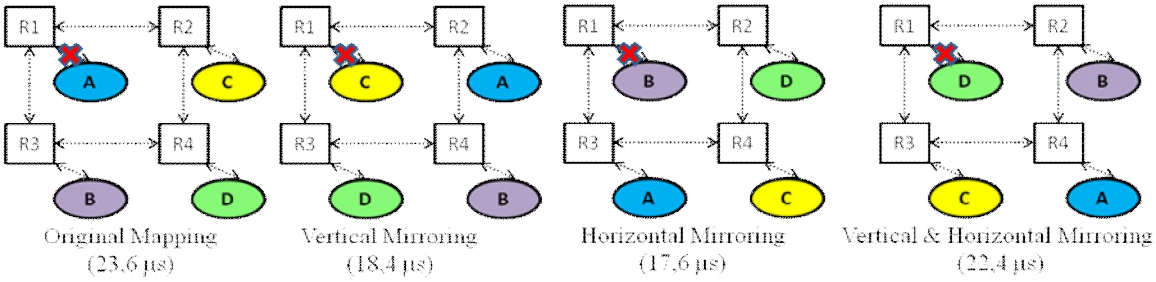


Fig. 1- Mapping possibilities without affect the original map features choose in design time.

4. Experimental Results

The proposal has been compared with Hamming solution, and results about area, frequency and power are presented in table I to one router, considering a 12-cores network. To evaluate the results, we considered links with channel-width equal to 8 bits. Synthesis results have been based on 65nm CMOS standard cell library using Synopsys Power Compiler tool. Our proposal has 30% of area overhead, but the impact in the critical path is lower than Hamming, because our proposal uses multiplexers 8:1 in the critical path, while Hamming needs long XOR cascades. Besides that, our proposal can win in energy, because our resources are turned on just in regions with faults. For each link in the network, Hamming also needs 4 extra wires to send the codification, while our proposal does not have extra wires.

Tab. 1 - Synthesis results

Router	Area (μm^2)	Critical Path Delay (ns)	Max. Freq. (MHz)	Router Power (µW)@Max. Freq.	Router Power (µW)@300MHz	# Total of wires
Non-protected	5360.3	1.11	900	334.06	111.3	864
Our proposal (DS on)	6978.7	1.71	585	498.49	255.9	864
Our proposal (DS off)	6978.7	1.71	585	216.64	112.7	864
Hamming	5948.8	2.04	490	295.21	180.6	1152

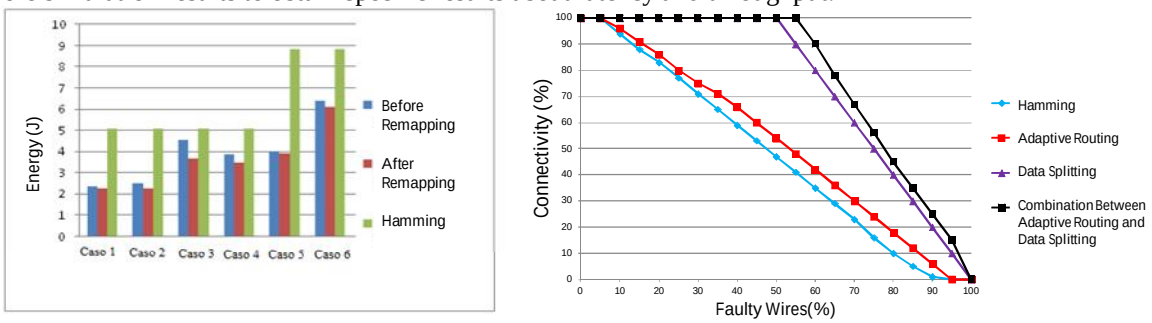
In fig. 2(a), the energy is presented for two benchmarks: MPEG4 [16] and VOPD [17]. The energy results were calculated with our proposal before and after the remapping, and both results are compared with Hamming solution. Each case considers a different and specific place where the fault occurs, and we will not explain the details because the space is short. For the energy results, we consider each communication sending 1000 packets from each source to target, and for each case the time considered is not the same. The energy results taken into account the power of the routers and wires (calculated based on [18] with HSPICE simulations). So, based on the results, we can see that is possible to reduce 19% of energy when the best mapping is choose (case 3), and our proposal can save 50% of energy when compared to Hamming (case 5).

We also consider the best case of tolerance for each proposal, considering multiple faults cases, as presented in fig. 2(b), for a 12-cores generic NoC. Hamming code has the best case of protection when there is just one fault in each link, the adaptive routing can consider until seven faulty links considering more than one fault in

each link. Data splitting presents good results when 50% of the wires are faulty-free, and our proposal can have better results because adaptive routing and data splitting have been combined to improve the results.

5. Conclusion

A NoC design with adaptive fault tolerance in the links to protect them against permanent faults was presented, in order to ensure a good yield. To provide the fault tolerance, we used a very simple adaptive routing strategy, combined with data splitting and tasks remapping. As a result, the energy can be reduced and the time impact can be better than other traditional techniques like Hamming code, since the critical path of our proposal has a reduced impact. Besides that, Hamming cannot cope with multiple faults in the interconnections and it needs extra wires, while our proposal does not need any extra wires in the links. The combination of adaptive routing, data splitting and tasks remapping here presented can ensure the yield because faulty links can be used even in the presence of massive faults in the wires, because only the faulty-free wires are used and configured to provide the right communication. As a future work, we intend to do more simulation results to obtain specific results about latency and throughput.



(a) (b)

Fig. 2 - (a) Comparison of the energy results (b) Best case of connectivity for each proposal.

6. References

[1] Braga, M.; Cota, E.; Kastensmidt, F.L.; Lubaszewski, M.; , "Efficiently using data splitting and retransmission to tolerate faults in networks-on-chip interconnects," *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, vol., no., pp.4101-4104, May 30 2010-June 2 2010.

[2] Lehtonen, T.; Liljeberg, P.; Plosila, J.;, "Online Reconfigurable Self-Timed Links for Fault Tolerant NoCs," *VLSI Design, IEEE International*, 2007.

[3] Concatto, C.; Almeida, P.; Kastensmidt, F.; Cota, E.; Lubaszewski, M.; Herve, M.; "Improving yield of torus NoCs through fault-diagnosis-and-repair of interconnect faults," *15th IEEE International On-Line Testing Symposium (IOLTS)*, pp.61-66, 2009.

[4] Agrawal, Vishwani D.; "Testing for Faults, Loooking for Defects," *Test Workshop (LATW), 2011 12th Latin American*, Keynote Talk, March 2011.

[5] Kologeski, A.; Concatto, C.; Carro, L.; Kastensmidt, F.L.; , "Adaptive approach to tolerate multiple faulty links in Network-on-Chip," *Test Workshop (LATW), 2011 12th Latin American*, vol., no., pp.1-6, 27-30 March 2011.

[6] Ganguly, A.; Pande, P.P.; Belzer, B.;, "Crosstalk-Aware Channel Coding Schemes for Energy Efficient and Reliable NOC Interconnects," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* , vol.17, no.11, pp.1626-1639, Nov. 2009.

[7] Palesi, M.; Kumar, S.; Catania, V.; , "Leveraging Partially Faulty Links Usage for Enhancing Yield and Performance in Networks-on-Chip," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* , vol.29, no.3, pp.426-440, March 2010.

[8] Kakoee, M.R.; Bertacco, V.; Benini, L.; , "ReliNoC: A reliable network for priority-based on-chip communication," *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011* , vol., no., pp.1-6, 14-18 March 2011.

[9] Koibuchi, M.; Matsutani, H.; Amano, H.; Mark Pinkston, T.; "A Lightweight Fault-Tolerant Mechanism for Network-on-Chip". *2nd ACM/ IEEE International Symposium on Networks-on-Chip*, pp. 13-22, 2008.

[10] Tornero, R.; Sterrantino, V.; Palesi, M.; Ordua, J.M.; "A multi-objective strategy for concurrent mapping and routing in networks on chip," *IEEE International Symposium on Parallel & Distributed Processing*, pp.1-8, 2009.

[11] Schonwald, T.; Zimmermann, J.; Bringmann, O.; Rosenstiel, W.; "Fully Adaptive Fault-Tolerant Routing Algorithm for Network-on-Chip Architectures," *10th Euromicro Conference on Digital System Design Architecture*, Methods and Tools, pp. 527-534, 2007.

[12] ta Choudhury, A.; Palermo, G.; Silvano, C.; Zaccaria, V.; "Yield Enhancement by Robust Application-specific Mapping on Network-on-Chips," *Second International Workshop on Network on-Chip Architectures (NoCArc'09)*, pp. 37-42, 2009.

[13] Zeferino, C.A.; Susin, A.A.; , "SoCIN: a parametric and scalable network-on-chip," *Integrated Circuits and Systems Design, 2003. SBCCI 2003. Proceedings. 16th Symposium on* , vol., no., pp. 169- 174, 8-11 Sept. 2003.

[14] Changbo Long; Lei He; "Distributed sleep transistor network for power reduction," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pp. 937-946, 2004.

[15] Shi, K.; Howard, D.; "Sleep Transistor Design and Implementation - Simple Concepts Yet Challenges To Be Optimum," *International Symposium on VLSI Design, Automation and Test*, 2006.

[16] 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.

[17] Vu-Duc Ngo; Huy-Nam Nguyen; Hae-Wook Choi; "Analyzing the Performance of Mesh and Fat-Tree topologies for Network on Chip design," *LNCS (Springer-Verlag)*, pp 300-310, 2005.

[18] Sakurai, T.;, "Approximation of wiring delay in MOSFET LSI," *Solid-State Circuits, IEEE Journal of* , vol.18, no.4, pp. 418- 426, Aug 1983.