A Survey on Design Approaches to Circumvent Permanent Faults in Networks-on-Chip

SEBASTIAN WERNER, JAVIER NAVARIDAS, and MIKEL LUJÁN,

The University of Manchester

Increasing fault rates in current and future technology nodes coupled with on-chip components in the hundreds calls for robust and fault-tolerant Network-on-Chip (NoC) designs. Given the central role of NoCs in today's many-core chips, permanent faults impeding their original functionality may significantly influence performance, energy consumption, and correct operation of the entire system. As a result, fault-tolerant NoC design gained much attention in recent years. In this article, we review the vast research efforts regarding a NoC's components, namely, topology, routing algorithm, router microarchitecture, as well as system-level approaches combined with reconfiguration; discuss the proposed architectures; and identify outstanding research questions.

CCS Concepts: • **Hardware** → **Network on chip; Failure recovery, maintenance and self-repair; System-level fault tolerance**; Aging of circuits and systems; • **Computer systems organization** → **Redundancy; Fault-tolerant network topologies**

Additional Key Words and Phrases: Permanent failures, topology, routing algorithms, router microarchitecture, system-level redundancy, reconfiguration

ACM Reference Format:

Sebastian Werner, Javier Navaridas, and Mikel Luján. 2016. A survey on design approaches to circumvent permanent faults in networks-on-chip. *ACM Comput. Surv.* 48, 4, Article 59 (March 2016), 36 pages.

DOI: <http://dx.doi.org/10.1145/2886781>

1. INTRODUCTION

Relentless advances in semiconductor technologies have led to scaling of transistor sizes into the deep submicron domain. Intel, for instance, introduced their fourth generation of processors, codenamed Haswell, a family of products implemented on Intel's 22nm trigate process technology [Kurd et al. 2014]. Furthermore, the International Roadmap for Semiconductors (ITRS) predicts 5nm technology nodes by 2020 [Committee 2014]. At current CMOS technology nodes, it is already feasible to implement hundreds of cores on a single chip, which renders current established interconnection technologies (mostly buses) impractical due to their lack of scalability and the single point of failure. Networks-on-Chip (NoCs), a new on-chip interconnection paradigm somewhat similar to computer networks, has emerged to overcome these limitations and is being adopted by industry, for example, in Tile64 [Bell et al. 2008], Xeon Phi [Chrysos 2014], or Kalray MPPA [de Dinechin et al. 2013]. In these large-scale many-core systems,

This research was conducted with support from the UK Engineering and Physical Sciences Research Council, on grants EP/K015699/1, EP/K008730/1, and EP/J016330/1. Mikel Luján is supported by a Royal Society University Research Fellowship.

Authors' addresses: S. Werner, J. Navaridas, and M. Luján, Kilburn Building, University of Manchester, Oxford Rd, Manchester M13 9PL; emails: {sebastian.werner, javier.navaridas, mikel.lujan}@manchester.ac.uk. Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

2016 Copyright is held by the owner/author(s). Publication rights licensed to ACM.

ACM 0360-0300/2016/03-ART59 \$15.00

DOI: <http://dx.doi.org/10.1145/2886781>

This work is licensed under a Creative Commons Attribution International 4.0 License

ACM Computing Surveys, Vol. 48, No. 4, Article 59, Publication date: March 2016.

Fig. 1. Bathtub curve model of device effect exposure [Constantinides et al. 2006].

NoCs play an increasingly important role and have become the limiting factor regarding performance, energy consumption, and area.

Decreasing transistor sizes also rendered semiconductors more prone to permanent faults due to two major effects: First, the increasing complexity of chip manufacturing gives rise to higher rates of post manufacturing defects caused by inaccuracies of the photolithographic and etching processes, leading to variability of material impurities, doping concentrations and size, and geometries of structures [Commitee 2014]. Secondly, decreasing feature sizes cause faster transistor aging and eventually transistor wear out, caused by Hot Carrier Injection (HCI), Bias Temperature Instability (BTI), Electromigration, and Time Dependent Dielectric Breakdown (TDDB) [Radetzki et al. 2013].

The probability of faults over time is typically illustrated by the bathtub curve model (see Figure 1). The infant period describes the time after fabrication and shows the effects of manufacturing defects on the yield. The burn-in method [Liang 1996] exercises the on-chip components prior to placing the chip in service. This forces faults to occur under supervision, which allows one to get an understanding of the load capacity of the system. As time passes on, the fault rates of the infant period decrease into a stable period, during which graceful system degradation is desired. Finally, transistor wear out begins and produces a quick degradation period. The entire bathtub curve is expected to be lifted up with future technology nodes, presenting higher fault rates in all periods.

Based on these forecasts on future technologies, designers are forced to consider the impacts of permanent faults on their implementations in order to maximize yield and to ensure correct operation, even in the presence of faults. This emphasizes the significance of robust design solutions and has led to fault tolerance becoming a *primary* design constraint.

Given the high fault rates in future CMOS technology nodes and the importance of the on-chip interconnection networks in many-core systems, NoCs should be designed in such a way that makes them robust against faulty components. Fault tolerance means that the NoC contains mechanisms to maintain correct operation in the face of permanent faults. In doing so, these mechanisms should ideally impose as little additional cost as possible while maintaining performance and power levels.

The level of robustness, that is, how many faults a NoC should be able to cope with, depends on the system demands and lifetime requirements. Faults in different parts of the NoC have different causes but all can result in severe consequences from

tremendous impact on performance and power, to incorrect functionality. Therefore, this article covers fault tolerance of the main NoC components.

1.1. Scope

The article at hand extensively reviews the research carried out in recent years regarding architectural measures to contain permanent faults. Other fault classes, such as transient and intermittent errors that could be caused by radiation, electromagnetic interference, or temperature variations, to name a few, exist [Constantinescu 2003], but exceed the scope of this article. Moreover, we note that the design of efficient fault models, formal verification [Parikh and Bertacco 2011], and fault detection mechanisms, such as BIST [Strano et al. 2011; Kakoe et al. 2011; Dimitrakopoulos and Kalligeros 2012], are further factors that significantly impact the efficiency of the resulting fault-tolerant architecture. Both of these design aspects will not be considered in this article as the architectural means to contain permanent faults constitute such a voluminous topic already. However, we refer to the survey paper [Radetzki et al. 2013] on fault-tolerant NoC methods, which thoroughly discusses fault models, detection mechanisms, and transient/intermittent faults. The difference to our article is that we exhaustively focus on the architectural aspects of the different NoC components, rather than walking through the communication layer approach [Radetzki et al. 2013], and include more recent proposals of the last few years. We also elaborate work on topologies and system-level approaches, and consider recent emerging technologies such as 3D integration, photonic interconnects, and wireless NoCs.

We will start discussing NoC topologies, in particular the traits that make a topology robust against faults, and will review some novel fault-tolerant topologies. Closely related to the topology is the need for fault-tolerant routing algorithms capable of exploiting the theoretical capabilities of the topology, even in the presence of faults, while considering possible issues such as congestion, changes in the topology, isolation of nodes, etc.

Router architecture is also of extreme importance. Faults within the router could lead to incorrect routing computation or switch allocation, and ultimately to misrouted packets. This can cause higher latencies, congestion, or even make nodes unreachable. Faults on links can be contained by maintaining partially functional links and thereby preserving the topology and avoiding congestion, rather than abandoning the whole link and lose connectivity.

Finally, there are a number of system-level approaches exploiting the abundance of transistors in future many-core chips by providing network reconfiguration, that is, the integration of spare cores, routers, or links into existing NoCs in case the original components fail. The efficient use of spare components can significantly extend a chip's lifetime, while providing close to maximum performance levels.

Each of the sections will briefly cover background information on the corresponding part of a NoC, followed by a thorough literature review and a critical discussion on the strengths and weaknesses of these approaches. Finally, we analyze research trends and highlight open challenges to guide future research on this topic. We note that efficient, fault-tolerant NoC design is an interdisciplinary research effort as all the different components of a NoC have a direct impact on each other and must be chosen accordingly to result in an overall efficient and fault-tolerant design.

As studies have shown that metal wires are unlikely to satisfy future power and throughput demands [Owens et al. 2007], emerging technologies, namely, 3D integration, wireless and photonic interconnects are increasingly considered as their replacement. Since they are in their infancy and their components are still prone to faults, we foresee that robust interconnection designs may be of increasing interest for these

technologies. Therefore, fault-tolerant design proposals including these technologies will be included in this article as well.

2. RESEARCH ON FAULT-TOLERANT TOPOLOGIES

Increasing fault rates and number of processing elements in future technology nodes inherently call for topologies that are both scalable and robust against permanent faults. Given the stringent constraints regarding the power budget and performance in many-core systems, topology solutions fulfilling these requirements should ideally not impose any negative drawbacks on these metrics. This section will review important topology characteristics, their effect on the fault tolerance of a topology, discuss commonly deployed topologies regarding their robustness, and review recently introduced topologies that aim to provide high resilience to faults. We will critically review each of these topologies and analyze them regarding their scalability, performance, and physical Very-Large-Scale Integration (VLSI) implementation.

2.1. Background

Network topology is a key factor on performance, cost, and fault tolerance. The topology defines how nodes and links/channels are arranged. Vertices represent the modules on a chip (e.g., Processing Elements (PEs), memory blocks, routers, or network interfaces) and edges the wires. Topologies can either be direct or indirect. In direct topologies each node is both a terminal and a switch, whereas in indirect topologies a node is either a terminal or a switch. Concentration or clustering can be applied to topologies, in which several PEs access the network over a shared router. For fault tolerance, the following topological characteristics have to be considered [Gillard and Li 2011]:

- Diameter*: The longest path between any pair of nodes. The average distance is occasionally used as a related metric. These affect message latency and energy consumption since each hop incurs some delay and use of energy.
- Node Degree*: (or radix) The number of links connected to a node, and thus the number of neighbors. A network is called irregular when the node degree is not uniform, and regular otherwise. While high radices can be exploited to decrease hop length, radix significantly contributes to I/O complexity and route area (allocators scale quadratically with the radix [Kim et al. 2007]). Therefore, smaller node degrees are often preferred due to less hardware costs.
- Bisection Width*: The minimum number of links that have to be removed to split the topology into two disconnected halves. A larger bisection bandwidth provides more paths between two subnetworks, and so impacts performance and fault resilience.
- Symmetry*: There are two types of symmetry [Dally and Towles 2004]: A topology is *node symmetric* when all nodes have the same view of the topology, which simplifies routing as they all share the same routing space. Similarly, a topology is *link symmetric* if all links have the same local environment, which simplifies load balancing. The presence of permanent faults would break this symmetry.
- Path Diversity*: The availability of multiple independent paths between most pairs of nodes. This adds to NoC robustness by balancing load and allowing to tolerate faults more efficiently [Dally and Towles 2004].

The occurrence of faults in links or nodes directly affects these topological characteristics, and as a result the performance and energy consumption of the entire NoC. Given the increasing error rates of permanent faults predicted for future technology nodes [Committee 2014], topology selection should consider robustness. The ring topology (see Figure 2(a)) has been used commercially (e.g., Cell processor [Gschwind et al. 2006]), and exemplifies how low path diversity can have a tremendous impact in faulty scenarios. If a link breaks between two neighboring nodes in a ring topology with N

Fig. 2. Common NoC Topologies [Dally and Towles 2004].

nodes, the distance between these two nodes changes from 1 to $N - 1$, affecting NoC load and latency significantly. Other well-known topologies like the star or binary tree, in Figures 2(b) and 2(c), respectively, are even more sensitive to faults. With a single link fault, these topologies will be divided into two disjoint subnetworks, making them unsuitable for fault prone environments. Fault resilient topologies feature a richer number of links and thus higher connectivity, for example, the Mesh and Torus (direct), or the Fat Binary Tree and Butterfly (indirect) in Figures 2(e), 2(f), 2(d), and 2(g). A more advanced topology trying to increase performance and load balancing is the Flattened Butterfly [Kim et al. 2007] (see Figure 2(h)).

A further characteristic of topologies supporting fault tolerance is whether the topology is Hamiltonian, that is, if it contains a Hamiltonian cycle: A path that traverses every node exactly once and in which the first and the last node are the same [Chen et al. 2006]. This means that nodes have at least two ports able to reach every other node, effectively ensuring some degree of connectivity. It is thus desired that a graph remains Hamiltonian in the presence of permanent faults, that is, a high number of faults shall be required to render a graph non-Hamiltonian. While this allows the network to cope with broken links without a significant degradation, it can also increase the wire complexity and node degree, leading to more complex VLSI layout and router microarchitecture.

High PE counts exacerbate the wiring complexity problem, stressing the need for scalable topologies that find a good balance between fault tolerance, integration complexity, scalability, performance, and power consumption.

One approach to address the scalability problem is using hierarchical NoCs, in which a network is divided into several subnetworks and thereby locally using the advantages of lower scale networks without affecting the scalability of the entire network too much. A simple illustration example for a hierarchical topology is the globally Mesh locally Star topology in Figure 3. Another emerging solution is 3D NoCs, that leverage 3D and/or 2.5D integrated circuits and thereby decreasing the diameter of a 2D plane without the need of additional wiring. Figure 4 illustrates a 3D Mesh topology, in which the horizontal connections are implemented using conventional wiring, and the vertical connection usually using Through-Silicon Vias (TSVs).

2.1.1. Implementation Concerns. The high-level theoretical analysis of a topology gives a good first estimate of its characteristics, particularly regarding the robustness of a

Fig. 3. Hierarchical Topology [Flich and Bertozzi 2010].

Fig. 4. 3D Mesh [Flich and Bertozzi 2010].

design. However, implications of this analysis on the physical silicon implementation of the topology are not always straightforward and can have a significant impact on the performance, area, and feasibility of the NoC. Three major design pitfalls should be noted regarding VLSI implementation [Flich and Bertozzi 2010]:

- (1) *Wiring Complexity* influences the number of metal layers required to implement the design and hence its cost. A too high wiring complexity may also render the design infeasible.
- (2) *Express/Long Links* may come with a considerable price for more complex topologies as they may turn out to be very long and thereby limiting the maximum frequency in the topology. Repeater insertion and link pipelining are techniques to solve this problem; however, these may entail a significant area overhead.
- (3) *Nonuniform Switch Radix* leads to the case where the highest switch radix in a network limits the final maximum frequency at which the network is run. Therefore, the operating frequency is determined by the highest radix switch and the longest link in the topology. A side effect of this is that lower radix switches will be synthesized with less area, making the implementation more imbalanced.

Given these aspects, dealing with robust topologies with high radix switches and high connectivity has to be considered carefully with feasibility and scalability in mind.

2.2. Recursive Diagonal Torus

The Recursive Diagonal Torus (RDT) topology consists of recursively structured mesh/torus connections, with different sizes in the diagonal directions and can use the originally proposed routing algorithm called “vector routing” [Yulu and Amano 1996]. The basis of RDT is a 2D square array of nodes with four links from each node to its neighbors, forming the rank-0 torus (T_0). To form a rank-1 torus (T_1), we add four extra links rotated 45° and connected to neighbors at distance n . Similarly a rank-2 torus (T_2) is formed from a (T_1), giving the RDT its recursive structure (T_r is built on top of a T_{r-1}). RDT is typically generalized as $\text{RDT}(n, R, m)$ [Yang et al. 2001], in which the distance between ranks is n and every node belongs to T_0 and m other upper tori ($T_i; i \in [1, R]$). Depending on how we assign nodes to ranks, referred to as torus assignment, we can find different arrangements, for example, $\text{RDT}(n, R, m)/\alpha$ [Yang et al. 2005] and $\text{RDT}(n, R, m)/\beta$ [Fan et al. 1999]. A special case of this class is the perfect RDT, $\text{PRDT}(n, R)$, in which each node belongs to all possible ranks ($m = R$ and so, omitted). A 4×4 $\text{PRDT}(2, 1)$ is shown in Figure 5. Another variant is the quartered RDT, which has only rank-0 and rank-1. Denoted as N -QRDT, it has $N \times N$ nodes, with $N = 4n$ for any positive integer, n . A N -QRDT has N^2 rank-1 links and $4N^2$ rank-0 links.

Figure 7 shows the diameter analysis results, as reported in Yu et al. [2005], which motivates the use of the RDT given its much smaller diameter compared to the Mesh/Torus.

Fig. 5. 4×4 PRDT(2,1) Topology [Duan et al. 2007].

Fig. 6. Binary Generalized de Bruijn graph $GDB(2, 14)$ [Hosseinabady et al. 2007].

Fig. 7. $RDT(2,2,1)/\alpha$ Diameter [Hosseinabady et al. 2007].

RDT topologies, in particular the $RDT(2, 4, 1)$, were first introduced as a general purpose interconnection network for massively parallel computing systems and showed their superiority to meshes, tori, and hypercubes, in terms of diameter, scalability, and bisection bandwidth.

Various other variants of the RDT topology class have been studied. Yu et al. [2005] proposed a $RDT(2, 2, 1)/\alpha$ scalable NoC design, and showed that it is feasible to implement using six metal layers. This rank assignment has simpler structure and routing than the original RDT, and constant node degree of 8, making it a robust NoC topology solution by providing a high bisection bandwidth.

Duan et al. [2007] introduced the PRDT topology. They study the $PRDT(2, 1)$ variant of this topology, in which each node has a degree of 5, thereby significantly reducing the wiring complexity and effectively creating a more scalable solution than the $RDT(2, 2, 1)/\alpha$, while still providing a fairly high bisection bandwidth to support fault tolerance. Adaptive routing was thereby shown to be more efficient than deterministic as it reacts more efficiently to occurring hot spots in a 4×4 $PRDT(2, 1)$ NoC.

PRDT can be implemented using just two metal layers [Yang et al. 2007], with the possibility of reducing channel widths significantly when more metal layers are available. Still in the $PRDT(2, 1)$, Xinming and Xuemei [2010] proposed a fault-tolerant routing algorithm capable of routing around faulty regions, while ensuring deadlock-free operation by using virtual channels. Their evaluation shows that the packet latency degrades gracefully.

Tan et al. [2008] dealt with the implementation of a shortest path routing algorithm for QRDT, codenamed Johnson Coded Vector Routing (JCVR). By applying minor modifications of their algorithm, they show that it is capable of handling single link/node faults. Superiority of the QRDT compared to Mesh, Torus, and Hypercube regarding

diameter and scalability was confirmed. JCVR is superior to VR routing as it can ensure minimal routing and deals with single link/node faults, with only moderate area increase.

2.2.1. Analysis. RDT topologies provide high radix, high connectivity, and path diversity, which makes them suitable for robust design solutions. The discussed work showed that the RDT topology requires six metal layers, while PRDT only needs two, which suggests that the latter is a more cost-efficient approach. While it scales well regarding diameter and provides a high bisection bandwidth, it also introduces more wiring compared to the Mesh/Torus topologies: however, its feasibility has been shown, provided that area constraints are met. Another PRDT feature is its constant node degree, which in turn results in even timing closure. It can be observed that a large number of express links will be required to implement this topology, which usually entails area overhead for repeater and pipelining insertion. This can be considered as the major drawback regarding this topology. The use of on-chip photonics to replace these long links with optical links may be worth investigating in this context, as they were shown to be efficient in replacing very long electrical wires [Pan et al. 2009]. The synthesis results of the discussed work used a $0.18\mu\text{m}$ technology, which may not be very representative for current technology nodes at 22nm. Implementation results at these feature sizes would be interesting to identify trends.

Routing algorithms that allow fault tolerance were proposed, but their performance evaluation lacks comparisons to other topologies. This would be helpful to confirm the analytical advantages of RDT to other topologies under consideration of different benchmarks and workloads. Apart from PRDT, no other RDT has been provided with a fault-tolerant routing algorithm. Making efficient use of the robust attributes of this family of topologies is the key study that should be conducted. Router microarchitecture for RDT networks has not been sufficiently addressed yet. Only an efficient architecture for the QRDT topology has been proposed by the time this article has been written. Router architectures for the others would be necessary to further understand performance/power/resilience trade-offs.

All in all, there are several reasons that suggest that RDT, especially PRDT, are promising topologies to tackle high fault rates and scalability. The common trend in industry, however, is to use simpler topologies such as meshes or rings. RDT topologies introduce more design complexity in this aspect, but also bring along various design advantages. Further research on RDTs may therefore be helpful once the Mesh topology fails to scale or provide sufficient fault tolerance.

2.3. De Bruijn Graphs

The second fault-tolerant group of topologies of interest (due to its high connectivity, small diameter, and good scalability) are those based on the de Bruijn (DB) graph. The generalized de Bruijn graph $GDB(d, n)$ was proved by Baker [2011], to have a diameter scaling logarithmically with the number of nodes, making it a suitable choice for large-scale NoCs. GDB is also known to be Hamiltonian [Du et al. 1991].

A common GDB is the binary GDB, $GDB(2, n)$, shown in Figure 6. It ensures that the majority of the nodes have a radix of four, allowing a fairly high connectivity in the NoC for fault tolerance, while a noninsignificant part of the nodes has three or two links, thereby decreasing the wiring complexity, allowing a reasonable trade-off between fault tolerance and scalability.

The feasibility of implementing DB graphs in VLSI has been shown [Samatham and Pradhan 1989] and deadlock-free routing techniques based on wormhole routing have been proposed [Park and Agrawal 1995; Ganesan and Pradhan 2003].

Hosseinabady et al. [2007] proposed a reliable NoC design based on the GDB topology and on a switch design including a detouring fault-tolerant routing algorithm. They particularly deal with a GDB(2,14)–14 nodes. High-level results confirmed the superiority of the GDB compared to the Mesh and Torus in terms of latency and energy consumption.

Hosseinabady et al. [2008] showed that only two metal layers are required to implement the wiring. While keeping self-loops and redundant links to have a constant node degree and thus a uniform switch design, they implemented and synthesized a switch design in $0.18\mu\text{m}$. Area savings of up to 40% and 80% compared to Mesh and Torus are reported, with better scalability. The GDB router was also shown to use less energy compared to Mesh/Torus routers. Finally, they show that the wiring complexity is comparable to that of the Torus for different network sizes and tile dimensions. Hosseinabady et al. [2011] extended the investigation of $GDB(2, n)$ graphs as a NoC topology to explore a fault-tolerant routing algorithm capable of dealing with permanent link faults by detouring faulty regions. Lower average packet latency and energy consumption than Mesh and Torus networks were depicted, with better scalability. Compared to the fault-free case, their fault-tolerant routing algorithm only features 3.6% dynamic power consumption overhead with the injection of faults in 20% of the channels.

The DB graph was also studied in the context of 3D NoCs. Chen et al. [2009] proposed a partitioning into horizontal and vertical plane networks, in which both planes are DB graphs, along with a routing algorithm proposal that features lower latency than a 3D Mesh but higher power consumption.

2.3.1. Analysis. DB graphs provide a scalable topology with moderate wiring complexity and varying switch degrees, implementable with two metal layers. As previously discussed, the varying switch degrees have the effect that routers will not be equally sized and the maximum frequency is limited by the highest radix in the network. In this context, the RDT provides a more regular design, but at the cost of more wiring. As GDBs feature very long wires they are very difficult to evaluate without actually physically implementing them. The maximum frequency, however, will be limited by the long wires connected to the nodes with the highest switch degree. The trade-off between moderate wiring, bisection bandwidth, diameter, and path diversity, which is theoretically excellent in DB graphs, may lead to problems in the physical implementation. Most of the discussed studies only dealt with high-level evaluations of DB-based NoCs. GDBs have been synthesized in $0.18\mu\text{m}$, which may not be a meaningful result for the deep submicron domain, where the delay is exceedingly high compared to transistors. Also, this study lacks physical design implementation. Studying the actual physical implementation would be of high interest as this seems GDB's only possible weakness. Large-scale GDB NoCs can offer significant area savings compared to meshes and tori but can have a nonnegligible overhead for small NoCs. GDB significantly outperforms meshes and tori in terms of packet latency, even for smaller networks (<40 nodes). Although GDBs appear to be superior compared to standard topologies in terms of performance and robustness, they are far from being adopted by industry. The main reasons for this neglect are a complex structure together with a lack of proper physical implementation and testing works with current technology nodes. Applying representative benchmarks, such as PARSEC [Bienia and Li 2011], would also be required to evaluate GDB's affinity to real applications.

2.4. Super Fault-Tolerant Hamiltonian Graphs

Fault tolerance at the topological level can also be achieved by featuring high connectivity through the Hamiltonian properties of topologies. Chen et al. [2006] dealt with

Fig. 8. Super fault-tolerant Hamiltonian graphs [Chen et al. 2006].

the construction of what they call *super fault-tolerant Hamiltonian graphs* (SFHGs), which are k -regular (i.e., constant node degree of k) Hamiltonian connected graphs that remain Hamiltonian if no more than $k - 2$ nodes and/or edges are removed and remain Hamiltonian connected if no more than $k - 3$ nodes and/or edges are removed. This allows high connectivity and maximum fault tolerance for a given node degree. Figure 8 illustrates two of these SFHGs. These topologies would inherently support fault tolerance for NoC, and can be optimized for certain application patterns or network sizes.

2.4.1. Analysis. To the best of our knowledge, no VLSI implementation proposals were addressed in recent work regarding SFHGs. This study would certainly be intriguing as the wiring seems more complex than in other topologies, so that further studies could be justified. An interesting attribute of these topologies is the constant node degree, which facilitates major parts of VLSI implementation. The most critical part of these topologies is certainly the large wiring requirements, which may be costly and require many metal layers. In return, however, they will provide the highest degree of fault tolerance and would therefore be suitable for high-reliability environments. The floorplanning is not straightforward as various connections exist that will be implemented as express channels. The cost overhead in this context would be interesting to evaluate, particularly for current technology nodes.

2.5. Complex Network Inspired Topologies

A completely different approach was taken by Ganguly et al. [2011] who proposed the use of complex network inspired topologies to provide fault tolerance, given that they are inherently fault tolerant. As these largely irregular topologies feature very long links, they implemented them using wireless connections in the form of carbon nanotube antennas. Their evaluation results show a very graceful performance degradation even at very high rates of wireless link faults. To the best of our knowledge, this has been the only study including both fault tolerance and wireless links for NoCs. While this study proved tremendous resilience against high fault rates, it is probably difficult to implement any sort of quality of service in such highly irregular networks. Its usefulness for industrial designs is therefore limited.

3. RESEARCH ON FAULT-TOLERANT ROUTING ALGORITHMS

In this section, we will discuss all different kinds of on-chip routing algorithms and how they have been used to provide fault tolerance in NoCs. These are, in particular, Flooding and Stochastic Routing, as well as Static and Adaptive Routing. We can

consider two different routing schemes, depending on where the routing information is held and where the routing decision is made:

- In *distributed routing*, packets carry the destination address and routers decide an output port either by looking up in routing tables or by performing a hardware routing function with Finite-State Machines (FSMs).
- In *source routing*, routing decisions are made in the Network Interface of a PE and included in the packet headers, removing the need for routing tables or functions in intermediate routers but increasing the overall size of the packets.

Routing algorithms in NoCs commonly follow the former scheme and are usually customized and dependent on the underlying topology, but can also be topology agnostic. In general, the decision on which routing algorithm to use should be made based on the following characteristics [De Micheli and Benini 2006]:

- Power* is influenced by the number of hops required by a message to reach its destination. Therefore, it is desired to have minimal routing, that is, only the shortest path is chosen. Routing complexity is another contributor to power dissipation, as more complex routing functions or tables require more energy.
- Area* is determined by the VLSI resources required to implement the corresponding routing scheme, normally enforced by either routing tables or FSMs.
- Performance* is, on the one hand, proportional to the number of hops, but is also influenced by how well the traffic is balanced over the network.
- Reliability* should be provided by implementing robust routing schemes capable of coping with both link/router faults and changes in traffic patterns.

Routing algorithms that support fault tolerance in NoCs can be further categorized based on the following attributes:

- Topology Agnostic*: The routing algorithm may be applied to regular or irregular topologies and is not limited to specific ones.
- Complete Reachability*: The routing algorithm is always capable of finding a path, provided it exists, for any given source-destination pair.
- Fault Independence*: Means that the algorithm is capable of coping with faults that may occur in any router and/or links at any moment.
- Scalability*: Router area overhead imposed by the routing algorithm is independent or increases only marginally with the NoC size.

3.1. Background

3.1.1. Deadlock Avoidance Techniques. Deadlock avoidance is one of the major concerns regarding on-chip routing algorithms. Deadlock situations occur when there is a cyclic dependency for channels (see Figure 9). The presence of link or router faults can lead to deadlock situations that do not occur in normal operation. Most deadlock avoidance mechanisms try to avoid the formation of cycles, typically either by restricting turns or by employing Virtual Channels (VCs):

- Turn Model* prohibits packets taking particular turns to avoid cycles [Glass and Ni 1992]. In the preceding example, if one packet had not turned counterclockwise, deadlock would have been avoided. In the turn model, all possible turns that packets can take in a network are analyzed along with the cycles that these turns could cause. An acyclic dependency graph is generated, resulting in a deadlock-free network.
- Virtual Channels* in adaptive routing can be used to support deadlock avoidance. VCs can be added to reconnect the network VCs to essentially provide additional buffers for packet propagation [Dally and Aoki 1993]. In the preceding example, if any of the packets had been put into a different VC, the cycle would have disappeared and

Fig. 9. Deadlock situation.

Fig. 10. Convex (blue)/concave (black) shapes.

so the deadlock. The number of VCs is generally kept to a minimum as they impose complexity, and thus area and power overheads, to the router microarchitecture.

3.1.2. Fault-Tolerant Routing Methodologies. There are two different methodologies that are employed to perform routing when errors are present in the network, forcing the original routing algorithm to adapt to the change in the network.

- Handling Convex/Concave Shapes* is based on defining fault ring/chains around faulty network elements, forming *fault regions*. Some methodologies are too restrictive meaning that functional nodes/links might be included in the resulting fault regions and thus are disabled to form a specific shape that the routing algorithm can deal with to prevent deadlock situations [Fick et al. 2009b; Sui and Wang 1997; Park et al. 2000]. Convex and concave shapes that could be formed as a reaction to faulty links/nodes are demonstrated in Figure 10 in blue and black, respectively. All components within fault regions are excluded from the network functionality, indicated by crossed out nodes or blank links, that are surrounded by these shapes.
- Contour Strategies* are implemented in the routing algorithm to circumvent permanent faults without the need of defining fault regions or disabling healthy nodes. Solutions dealing with deadlock situations can be categorized into those that use VCs [Koibuchi et al. 2008] and those that do not [Wu 2003; Fick et al. 2009a].

3.2. Fault Tolerance Through Packet Replication

In *Flooding*, several copies of a packet are dispatched into the network to increase packet delivery rate. *Simple flooding* is the basic technique in which each router sends a copy of an incoming packet to all output ports. This can be combined with *Stochastic routing*, where a router forwards a packet based on a particular probability. Different variants of these algorithms have been introduced. A router using *Stochastic flooding* will pass on incoming messages to an output port with probability p . *Directed Flooding* is a destination-aware variant of stochastic flooding, in which the likelihood of passing on a packet is computed based on the destination address. These variants mainly aim to decrease the number of packets traveling through the network. In *N random walk*, the injection of a fixed number of copies of a message into the network is allowed [Zhu et al. 2007]. A study comparing XY-routing (deterministic), Negative First Algorithm (partially adaptive), and N random walk (stochastic) for different N shows that packet replication schemes are inefficient regarding energy consumption and throughput, as they rapidly saturate the NoC and, hence, cannot compete with standard single-copy algorithms. For this reason, their use is discouraged and so, this article will not cover them in detail.

3.3. Fault-Tolerant Static Routing

In *static routing*, permanent paths are computed for each source-destination pair. This facilitates the implementation of router logic, the interaction between routers, and avoids the need for packet-reordering mechanisms as in-order delivery is ensured. The price to pay is, normally, the inefficient handling of permanent faults in links/routers, given that a single link fault would suffice to force each PE to update their routing tables.

3.3.1. Distributed Routing. Extended XY routing [Wu 2002] achieves fault-tolerant, deadlock-free operation by using rectangular blocks to contain faulty elements. As a high number of healthy nodes has to be sacrificed within the fault regions, further studies [Wang 2003] refined these regions by introducing minimal-connected components (MCCs) so as to significantly reduce sacrificed nodes. Wu and Wang [2005] further modified this routing for 2D Meshes. Based on the odd-even turn model, this deterministic routing algorithm does not require VCs for deadlock avoidance.

Segment-based Routing (SR) [Mejia et al. 2006] splits a topology into subnets, and these into segments and allows one to place turn restrictions locally. It can be applied to any irregular topology derived from the 2D Mesh and increases flexibility because turn restrictions within segments can be placed independently. Combinations are possible with any deterministic or partially adaptive routing algorithms. The SR strategy has been extended to become topology agnostic [Mejia et al. 2009; Fu et al. 2014; Lee et al. 2014].

DR-NoC [Ali et al. 2007] sends packets through the shortest available path. When a link/router fails, the fault info is disseminated to all network components and then Dijkstra's algorithm is used to compute new shortest paths.

Fick et al. [2009a] proposed an algorithm for 2D Mesh and Torus topologies capable of overcoming a large number of faults by reconfiguring the NoC around faulty components in a distributed manner, where each router holds the information on the shortest routes. Even in the face of high fault rates it manages to maintain connectivity and correct operation. Evaluation results show that it requires less than 300 extra gates per router and provides 99.99% reliability for 10% of faulty links.

3.3.2. Source Routing. SRN [Kim and Kim 2007] consists of two mechanisms: Route Discover and Route Maintenance, which respectively allow nodes to initially discover routes to other nodes and to fix them in case of link faults. The whole algorithm works on demand with route requests and replies, which detect link faults. SRN provides comparable fault tolerance as flooding algorithms but reduces packet overhead.

Mediratta and Draper [2007] proposed a routing algorithm for k-ary two-cube based on path exploration for affected source-destination pairs when faults are detected. This is done either during system reconfiguration or after reset, and hence it improves fault tolerance but at the cost of stopping system operation upon new faults.

Wachter et al. [2013] introduced a fault-tolerant source routing algorithm with path exploration that is topology agnostic and can thus be applied to both regular and irregular topologies. Moreover, the total path search and computation time is small compared to literature and features an area overhead of 50%, but also ensures full reachability even under severe fault scenarios.

3.3.3. Analysis. In static routing the path for a given source-destination pair is static and known beforehand regardless of the network status, that is, link/router faults or traffic loads. While it has the advantage of in-order delivery, it does not provide the flexibility that may be required in fault-prone environments. Each source-destination pair whose path is affected by a permanent fault would have to recalculate a new path, which significantly contributes to the network delay. As the NoC scales up, this problem

will exacerbate as potentially more paths are affected. This, in combination with the fact that deterministic routing algorithms cannot adapt to traffic loads, suggest that static routing may not be suitable for the fault-tolerance and high-performance demands of future many-core applications that require scalable, high-throughput networks.

3.4. Fault-Tolerant Adaptive Routing

In *adaptive routing*, also known as dynamic routing, a packet may use different paths to reach its destination. It can further be distinguished between *partially adaptive*, in which packets are restricted to use only some of the shortest paths, and *fully adaptive*, where a packet is allowed to propagate on any shortest path. Adaptive routing allows one to react to faults more quickly and provides more flexibility as several paths can be chosen from. The drawback is that larger routing tables are required to store the different routing information, which is costly in terms of area and power consumption, and has limited scalability. Research efforts dealing with fault-tolerant adaptive routing algorithms can be categorized into those approaches that require VCs to ensure deadlock-free routing, and those that do not. Moreover, logic-based approaches exist that do not require routing tables, and thereby saving hardware resources. The vast majority of routing algorithms are based on wormhole switching, which is more efficient than virtual-cut-through as it allows higher performance and more effective buffer usage. Fault regions, however, could pose restrictions on which switching technique can be used. Therefore, in order to be able to use the superior wormhole switching, routing algorithms have to be designed with this in mind. All discussed design proposals use wormhole switching, except *LBDR* [Flich and Duato 2008], which uses virtual cut-through. However, *LBDR* was further extended to support wormhole switching.

3.4.1. Fault-Tolerant Adaptive Routing with Virtual Channels. Chaix et al. [2010] proposed an algorithm capable of routing a message in the presence of any set of link/node faults. The algorithm is tailored to a 2D Mesh and employs VCs to avoid deadlocks by using two virtual networks: an output hierarchy and node stamping. Failure information is exchanged between neighboring nodes. The two variants Virtual Source Routing and the Echo Mode offer sufficient adaptability to provide high fault tolerance and guarantee complete reachability. Perfect delivery is ensured even under complex fault patterns and up to fault rates of 40%. Moreover, they do not require routing tables, which ensures scalability. *FTCAR* [Valinataj et al. 2010] requires only two VCs for both adaptivity and fault tolerance in a 2D Mesh. It is capable of reconfiguring itself after permanent faults are detected and does congestion-aware load balancing. Packets do not contain any extra information for the routing process and impose only little area overhead for fault handling, making their approach low cost and scalable. In case a link breaks, a contour surrounding around this fault is created, based on which a new routing scheme is defined. This new scheme does not affect message passing outside the contour area, which will keep using the original scheme. Several link faults within a network can be tolerated as long as their contours do not overlap, providing a relatively high yield with little implementation overhead. Its lack of a global view on the network, however, leads to limited link containment capabilities. Minimal routing can thus not be guaranteed and hot spots may be created around faults. *HiPFaR* [Ebrahimi et al. 2013a] is capable of dealing with several faults by using one and two VCs along the X and Y dimension, respectively, in the 2D Mesh. Based on the fault information of four neighbors of a faulty node, the algorithm is able to deliver the packets on the shortest path. The algorithm is only forced to choose nonminimal routes in case source and destination are both in the same row or column as the faulty node. Less area overhead

than a baseline detour router [Zhang et al. 2008b] was reported. In a 8×8 Mesh, they can achieve 98% successfully delivered packets, under random traffic when six faults are present. *FTLR* [Vitkovskiy et al. 2013] bypasses faulty links by rerouting packets in a U-shaped path around the faulty links/regions. While faults are being bypassed, the algorithm stores local detouring information dynamically in the packet header. If a fault is encountered during the detouring process, the rerouting continues recursively in the same fashion. When the detouring is complete, no more additional detouring information remains in the header. *FTLR* relies on local information, is fully distributed, and does not require global routing information. This leads to a lightweight design, as there is no dependency on look-up tables. *FTLR* requires two VCs in each dimension and has an area and power consumption overhead of 5% and 2.5%, respectively. Simulation results show a graceful performance degradation for up to 12% of link faults in a 8×8 Mesh. *CAFTA* [Dimopoulos et al. 2014] can handle any arbitrary set of link and node faults. At the same time, it is capable of load balancing to avoid hot spots without using routing tables. It combines North-Last and South-Last adaptive routing that uses turn restrictions to avoid deadlocks while using four VCs per input port. When a link or node fails, all neighbors are informed, so nodes are aware of adjacent faults allowing them to route messages around the faulty links. In order to balance the traffic, a “flits-remain” signal is used to count the number of remaining flits that still have to pass through the router, which is increased/decreased accordingly. *CAFTA* provides 97.68% successfully transmitted packets in a 16×16 mesh when 384 links are defective simultaneously, and 93.40% for 103 defective nodes. *AFRA* [Akbari et al. 2012] tolerates faults on the vertical links of a 3D Mesh by using ZXY and XZXY routings in the absence and presence of faults, respectively. *AFRA* guarantees tolerating single faults without the need for any VCs if they have the same direction (i.e., upwards or downwards). This allows using these VCs for performance improvement rather than for deadlock avoidance. *AdaptiveZ* [Rahmani et al. 2011] routes packets in a 3D layered chip based on the relative position of a sending node to the heat sink, located at the bottommost layer, to mitigate thermal issues while providing fault tolerance. It combines this adaptive algorithm with XY routing for intralayer communication. Deadlock-free operation is ensured by using available VCs. Loi et al. [2011] introduced an adaptive routing algorithm based on postmanufacturing study and reconfiguration of the electrical resources for 3D stacked chips. This scheme only leverages a small amount of on-chip spares and was shown to provide yields up to 98% with a minimum silicon cost of 20.9% per TSV link in 130nm, which is expected to further decrease to 14.2% at 65nm.

3.4.2. Fault-Tolerant Adaptive Routing Without Virtual Channels. Most fault-tolerant routing solutions that do not require VCs either tolerate only one fault [Duato 1993] or disable a number of healthy nodes [Duato 1995; Fukushima et al. 2009] to ensure deadlock-free operation. There are, however, a number of adaptive routing proposals that do not need to disable healthy nodes. *FDWR* [Schonwald et al. 2007] balances traffic uniformly over the entire network to avoid link congestion on links adjacent to faulty components. It can be applied to different network topologies without any changes in the routing algorithm. Before sending the actual flit, a router sends out an ADDRESS flit to all its neighbors, which answer with the shortest available distance to the destination node. If there is a link/router fault, no answer will be received which the sending router will recognize after a timeout period, indicating that this link/router is faulty. The distance received from the neighboring nodes is the number of hops plus a penalty parameter, which represents the current congestion status of this route, to support load balancing. *DCDG* [Ren et al. 2014] avoids deadlocks by constructing an acyclic channel dependency graph, that is, prohibiting particular turns. When a fault occurs, this graph is

updated (offline) to preserve deadlock-free operation. Evaluated in an 8×8 2D Mesh topology, 99.73% of the injected packets are successfully delivered. *uDIREC* Parikh and Bertacco [2013] treats links as two unidirectional channels, which help to make the network connectivity more robust. It is particularly superior to recent work at high numbers of faults, leading to much higher throughput and lower packet drops. *ARI-ADNE* [Aisopos et al. 2011] discovers paths between all connected nodes. Subsequently, it performs up*/down*, which is a deadlock-free routing algorithm. The area overhead of this approach is 1.97% as it requires only a small amount of additional wiring and hardware modifications, and minimizes communication among nodes. It ensures complete reachability regardless of the fault configuration. Compared to other state-of-the-art fault-tolerant solutions, it provides 40%–140% latency improvement when subject to 50 faults in a 64-node NoC. *4NP-First* [Pasricha and Zou 2011] combines turn models and opportunistic replication to increase successful packet arrival rate while optimizing energy consumption for 3D Mesh NoCs. It features low implementation overhead and provides better fault resilience than existing dimension-order, turn-model, and stochastic random walk schemes extended to 3D NoCs. *HLAFT* [Ahmed and Abdallah 2014] is a minimal hybrid routing algorithm for 3D NoCs that combines the benefits of look-ahead routing and local routing for better routing choices. Moreover, for deadlock-free operation, it employs Random-Access Buffer (RAB), a low-cost technique for deadlock recovery that takes advantage of look-ahead routing to detect and remove deadlocks without considerable hardware complexity.

3.4.3. Reducing the Impact of Routing Tables. Region-Based Routing [Mejia et al. 2009], *RBR*, groups destinations into regions, which allows one to reduce the number of entries in the routing tables by routing at the region level. While it was evaluated in a 2D Mesh, it could be employed in other topologies. By using 16 regions in an 8×8 mesh, they could provide 99% of reliability with up to seven link faults. In addition, they showed that *RBR* achieves the same performance as table-based solutions, while requiring much less area and power consumption due to their logic-based implementation.

Logic-Based Distributed Routing (LBDR) [Flich and Duato 2008] approaches aim to provide fault tolerance while decreasing the required on-chip resources for implementing routing tables. It enables the implementation of most existing distributed routing algorithms for both regular and irregular topologies. It deploys only two routing and one connectivity bit to mimic the behavior of routing algorithms implemented with routing tables, which can be reconfigured to react to faults in the network. These bits are employed to calculate the relative position of the current switch and the destination switch, which is then used in combination with the connectivity bit to determine the set of possible output ports. The computation of the routing bits is determined by the used routing algorithm. LBDR requires a module named *FORKS*, which replicates some messages for that, and at the same time requires virtual cut-through switching a complex router arbiter, which imposes a high overhead in buffer area. Moreover, a complex centralized reconfiguration strategy is required for some fault combinations to avoid deadlock situations. Universal LBDR (*uLBDR*) [Rodrigo et al. 2010] is another alternative to using routing tables, which is also capable of adapting to any irregular topology derived from the 2D Mesh and supports more complex algorithms than LBDR. This mechanism requires eight routing bits and four connectivity bits to define routing decisions and connectivity of a switch. These bits are then used by combinational logic to compute the routing decision. Faults are handled by setting the connectivity bits accordingly and routing around faulty elements through configuring the routing bits. How many topologies of the set of irregular topologies derived from 2D Mesh are supported by this mechanism determines the implementation overhead that has to be paid. Therefore a trade-off has to be taken regarding performance and coverage of

topologies. d^2 -LBDR [Bishnoi et al. 2015] is a further improvement that adds a distance register to the closest fault, enabling the support of more fault combinations without an excessive implementation cost, and allows wormhole switching. It achieves the same fault coverage as LBDR, without the complex switching strategies or any dynamic reconfiguration strategy. d^2 -LBDR has the same performance as LBDR, while it only has 3% area overhead, as opposed to the 300% overhead of LBDR. *ZoneDefense* [Fu et al. 2014] is a mechanism that does not only include faults into convex faulty blocks, but also spreads the information on the faulty blocks' position in the corresponding columns. Nodes that know the position of faulty blocks form the Defense Zones. This allows packets to find the faulty block in advance and route around it, which reduces the complexity of handling deadlock and the number of healthy nodes that were sacrificed. *ZoneDefense* does not impose any performance degradation on the network in the fault-free case. The comparison to Wu [2003] and Zhang et al. [2008b] shows that *ZoneDefense* has a very small area overhead (2.6% and 1.1%), while sacrificing a much smaller number of nodes.

3.4.4. Analysis. In general, adaptive routing has been shown to be more suitable for future NoCs as they provide high flexibility, quicker reaction, and better load balancing ability. The main drawback of using adaptive routing is that routing tables are required at each router, which significantly contribute to the overall area and power consumption. Not using VCs for deadlock prevention in this context requires less resources or allows VCs solely for increasing performance. However, recent work showed that most approaches only provide fault tolerance to a limited number of faults without performance degradation or require one to disable healthy nodes. LBDR constitute a promising solution to solve the scalability problem of adaptive routing while providing high flexibility by using a few extra connectivity and routing bits for fully combinational routing computation. Apart from being a lightweight solution, it can also be extended to all kinds of topologies, such as 3D topologies, which will be required in future systems as 3D stacking consolidates.

3.5. Reconfiguration of Routing Algorithms

At this point, we note that routing scheme reconfiguration has a significant impact on the overall performance and cost. Improving the reconfiguration process has also been explicitly addressed aiming to reduce complexity and reconfiguration time. *OSR-Lite* [Strano et al. 2012] is a centralized approach combining static and dynamic reconfiguration to obtain a simpler design and reduce reconfiguration time by 50%. Ghiribaldi et al. [2013] propose an approach using a global controller with full NoC visibility, able to reprogram the routing scheme. They added a lightweight ring-based control NoC responsible for exchanging diagnosis and configuration bits. To avoid the limitations of centralized approaches, there has been a recent effort to perform distributed reconfiguration [Lee et al. 2014; Balboni et al. 2015]. Balboni et al. [2015] overcome the main limitations of *OSR-Lite* by distributing it. The overhead of this approach is that it requires an escape network that is used during reconfiguration. Their approach relies on segment-based routing and uLBDR. The modifications of the configuration registers performed by the uLBDR routing algorithm are encoded in small tables for each router, making it a distributed approach. *BLINC* [Lee et al. 2014] is a minimalistic reconfiguration algorithm that uses precomputed routing metadata to quickly evaluate localized detours upon fault manifestation. Uninterrupted NoC operation is provided during reconfiguration, leading to negligible performance degradation. However, the complexity of this scheme is significant with poor scalability (more than 500 extra bits per router in an 8×8 mesh).

3.5.1. Analysis. Ideally, the reconfiguration process should affect network operation minimally. This has been addressed by using escape networks so that network operation can maintain during network reconfiguration, or by trying to minimize the reconfiguration time as much as possible. The major aspect of reconfiguration is whether to implement a centralized mechanism that has global visibility of the network, or a distributed mechanism dealing with local knowledge on the network. Distributed configuration strategies avoid single points of failure at the cost of high resource overheads for configuration logic and to improve the percentage of supported fault patterns. This is because every node has to be aware of the state of their neighboring nodes to explore a deadlock- and livelock-free routing path due to the lack of global visibility, often leading to overprovision of the NoC architecture (e.g., VCs, large storage). This process comes with large latency overhead and may not even be able to guarantee deadlock-free paths. The benefit is that no separate control network or global controller is required. Paying the cost of implementing a control NoC and module, however, allows a global view of the network, and therefore the exploration of minimal, deadlock-free routing paths between any source-destination pairs. Both distributed and centralized configuration may be critical regarding scalability for systems with large core counts. Separating a network in different subnetworks for fault detection and reconfiguration may be a sweet spot by trading off the pros and limitations of centralized and distributed approaches.

3.6. Discussion

In this section, we reviewed novel fault-tolerant routing schemes and methodologies. Packet replication was shown to have huge performance and energy overheads.

Some proposals use source routing in which nodes need to be notified when errors occur and subsequently perform path exploration in order to obtain new routing information. While path exploration is not particularly inefficient in terms of performance, it does not constitute a solution that scales particularly well as resource requirements grow quadratically with the number of nodes.

Static routing requires lower router area but lacks flexibility in fault-prone environments. Adaptive and partially adaptive routing algorithms have been extensively studied for fault tolerance. Designs that do not implement VCs face the challenge of minimizing the sacrifice of healthy components, which is required to create fault regions. Logic-based approaches using connectivity and routing bits instead of routing tables allow area savings in the router architecture while avoiding deadlock by either routing adaptively around fault regions under various turn restrictions, or statically. Therefore, they can be considered the most suitable adaptive approach to satisfy the demands of future NoCs, as they provide highly flexible, low resource implementations. Also, its complexity only depends on the number of input and output ports of a switch, rather than on the network size.

In most of the discussed designs there is a trade-off between general design characteristics—power, area, performance, and reliability—and fault tolerance—complete reachability, fault independence, and scalability. Table I lists the recent approaches and their attributes. NI means that no information was provided in the according approach regarding an attribute. The vast majority of proposed routing algorithms focused on the 2D Mesh, while only a few approaches covered other topologies or were topology agnostic. This is due to the fact that 2D Meshes are still satisfying the communication requirements of current many-core applications, and due to their simple VLSI implementation and design—properties commonly preferred by industry. Evaluation studies of the proposed designs often used network sizes of 16×16 , which shows that scalability has also been considered. However, future power budgets cannot be satisfied by 2D Meshes [Owens et al. 2007]. A moderate number of approaches dealt

Table I. Fault-Tolerant Routing Approach

Approach	Topology	Distributed	Source	Adaptive	Static	Fault Independent	Complete Reachability	Minimal
Chaix et al. [2010]	2D Mesh	✓		✓		✓	✓	NI
FTCAR	2D Mesh	✓		✓		✓	✓	✓
HiPFaR	2D Mesh	✓		✓		✓	✓	
FTLR	2D Mesh	✓		✓		✓	✓	
CAFTA	2D Mesh	✓		✓		✓	✓	
AFRA	3D Mesh	✓		✓			✓	✓
AdaptiveZ	3D Mesh	✓		✓		✓	✓	
Loi et al. [2011]	3D Mesh	✓		✓		✓	✓	✓
FDWR	2D Mesh/Torus	✓		✓		✓	✓	✓
DCDG	2D Mesh	✓		✓		✓	✓	NI
uDIREC	Agnostic	✓		✓		✓	✓	✓
ARIADNE	Agnostic	✓		✓		✓	✓	✓
4NP-First	3D Mesh	✓		✓		✓	✓	✓
HLAFT	3D Mesh	✓		✓		✓	✓	✓
uLBDR	2D Mesh	✓		✓		✓	✓	
d^2 -LBDR	2D Mesh	✓		✓		✓	✓	
ZoneDefense	2D Mesh	✓		✓		✓	✓	✓
RBR	2D Mesh	✓		✓		✓	✓	✓
Wu and Wang [2005]	2D Mesh	✓			✓	✓	✓	✓
SR	Agnostic	✓			✓	✓	✓	✓
DR-NoC	2D Mesh	✓			✓	✓	✓	✓
Fick et al. [2009a]	2D Mesh	✓			✓	✓	✓	✓
SRN	2D Mesh		✓		✓	✓	✓	✓
Mediratta and Draper [2007]	k-ary two-cube	✓	✓		✓	✓	✓	✓
Wachter et al. [2013]	Agnostic	✓			✓	✓	✓	✓

with 3D Meshes, supporting the forecast that 3D stacked chips are emerging to tackle the scalability problem. Dividing a network into regions is superior in terms of area and performance than global approaches, indicating that with an increasing network size, a partitioning of the network or using hierarchical networks may be more efficient. Most approaches provide complete reachability and fault independence, which are major goals for fault tolerance in NoCs given its impact on the performance and preservation of functionality. Observing the trend of fault-tolerant routing algorithm design shows that approaches evolve along with the technology trends of future many-core designs, that is, efficient fault-tolerant routing for 3D NoCs, hybrid NoCs with silicon photonic links, and systems with hundreds of cores.

4. RESEARCH ON FAULT-TOLERANT ROUTER MICROARCHITECTURE

Permanent faults occurring in network links or within a router's components must be considered and handled in order to contain their effects on network functionality, connectivity, and performance. Fabrication imperfections and resulting defects are assumed to occur randomly, and so their probability is assumed proportional to a component's area. This assumption is consistent with the experimentally observed breakdown

patterns studied in Keane et al. [2011]. We will first introduce the generic architecture of a NoC router and its components, and the impact that permanent faults could have on their operation. Subsequently, the research efforts regarding router design will be introduced and discussed. These can be divided into those dealing with robust design solutions within the router and that handling link faults. In addition, some holistic design proposals covering both of these parts will be covered. At last, fault tolerance for high-radix and 3D switches will be discussed.

4.1. Background

In a generic, baseline NoC router, an incoming packet will arrive at one of the input ports and subsequently be stored in the according input buffers. We only consider Wormhole routers, which is the commonly employed state-of-the-art flow control technique, in which a data packet is divided into flits before it is sent over the network. Depending on the topology, a router has a varying number of input and output ports. Each input port can include a certain number of VCs and corresponding multiplexers and demultiplexers to distribute packets over them. Upon arrival, the following steps are performed consecutively, forming the router pipeline [Poluri and Louri 2013]:

- Routing Computation (RC)* computes the output port for incoming head flits containing information on the destination address. Faults could cause the routing logic to forward all flits in the same direction, or even lead to a complete stall of the generation of routing signals. This can lead to misrouting of flits, which could cause deadlock situations in deterministic routing, or traffic hot spots that increase network congestion. A complete router breakdown would isolate affected PEs.
- Virtual Channel Allocation (VA)* allocates head flits in the input VCs to an empty output VC in the downstream router, based on the results obtained in the RC stage. Permanent faults in the VA can cause flit loss if the VA by mistake allocates an input VC to a downstream VC that is already occupied. In the worst case, the VA is not capable of maintaining operation, which is difficult to contain due to its interdependencies to other router components.
- Switch Allocation (SA)* controls the access of the flits residing in an input VC to the output port of the crossbar. Faults in the SA component can have similar effects as in the VA stage. Two input VCs could be assigned to the same crossbar output, causing overwriting of flits. If a flit is misrouted to an unoccupied output port, this may still cause deadlock situations or nonminimal paths.
- Crossbar (XB)* consists of multiplexers allowing one to send from any input to any output. The selected signals of these multiplexers are controlled by the SA unit and configured based on its computed arbitration. Faults in the crossbar component can lead to situations in which some output ports cannot be reached anymore, causing flit drops or in extreme cases complete breakdown of the router. If the affected output port is the one leading to the local processing element, this impedes its flit reception.

Faults in the input buffers, which are the connection between the input ports and the remaining routing components, can also degrade the performance of the system in case not all buffering resources can be used as a result of that. If a buffer is rendered defective, this decrements the number of available VCs, and in the worst case, isolates the entire input port. All the respective components of these stages are demonstrated in Figure 11.

The VC state stores the computation results of the different stages, along with other control signals such as read/write pointers for the buffers or the credit count.

Table II shows a typical area split of the different components. While the size of the area of a component has an influence on the probability of permanent faults to occur, the smaller components also need to be considered in the fault model as a single fault

Fig. 11. A conventional baseline NoC router architecture.

Table II. Area of Router Components [Liu et al. 2011]

	Buffer	Link	Crossbar	Allocators	Routing
Area	50%	39%	5%	3%	3%

in one of them could compromise correct operation of the entire router or even render it defective.

Besides the permanent faults that could occur within the router, the links between them are also a source of faults. Faulty links could cause wrongly transmitted bits between routers, possibly breaking the connection between them in case correct operation cannot be guaranteed anymore.

4.2. Hardware Redundancy

Several fault-tolerant router designs exist with different strategies to address fault tolerance of router components. The simplest approach to contain the effects of permanent faults is to employ hardware redundancy or additional circuitry. This allows the network to maintain its operation or to degrade its performance gracefully, while requiring more area for the additional backup circuitry. In *Bulletproof* [Constantinides et al. 2006], the components of a switch are structured into different partitions each having a spare component. Two resource sparing techniques are studied, namely, “dedicated sparing” where each spare is owned by a partition and its use is thus restricted to when this particular partition fails, and “shared sparing” where a spare can replace a set of partitions. While the latter seems more practical in terms of reusability, it also introduces additional interconnect and logic overhead, as it must be capable of fitting several different partitions. As fault tolerance is achieved through area overhead, they define the metric *Silicon Protection Factor* (SPF), as depicted in Equation (1), which is the ratio of the mean number of defects required to cause a router to fail (#defects) to the area overhead caused by the used technique to ensure fault tolerance.

$$SPF = \frac{\#defects}{area\ overhead}. \quad (1)$$

Default Backup Paths (DBPs) [Koibuchi et al. 2008] makes use of additional wiring, in order to maintain network connectivity between healthy routers and PEs in the presence of link faults, as well as alternative data paths to circumvent faulty components within a faulty router. In the worst case, these backup paths form a unidirectional ring network, effectively retaining a connected network even if all regular links break. Evaluated in a 2D Mesh with wormhole flow control, their results show that DBPs only require an area overhead of 12.6%.

ROBUST [Collet et al. 2011] is based on redundancy, in which the router is split into six sub-blocks, with five sub-blocks belonging to each of the five input ports, and one sub-block being dedicated to the crossbar. In parallel to these sub-blocks, they place so called *Universal Logic Blocks* (ULBs), which consist of several transmission gates acting like switches that can be configured to be multiplexers or demultiplexers. These ULBs are implemented with input sizes depending on which sub-block of the router they are supposed to protect. Hence, they have ULBs for replacing faulty VC buffers, next to one ULB for each input of the crossbar working as a multiplexer to connect to all possible output ports, and thereby replacing the crossbar switch. They compare their *ROBUST* implementation to *Bulletproof* and *Vicis* (see Section 4.5) showing that the former requires over $8\times$ transistors, and the latter around 10% more than *ROBUST*. In addition, they propose a new metric (Equation (2)), called Reliability Improvement Factor (RIF), which is defined as the ratio of the probability that permanent faults occur in the baseline router, F , to the probability that irreparable permanent faults occur in the fault-tolerant router, F_p . Based on their RIF metric, their comparison shows that, for any given probability of permanent faults, *ROBUST* results in higher RIF factors.

$$RIF = \frac{F}{F_p}. \quad (2)$$

At this point, it should be noted that adding redundant hardware blocks entails according crossbar-like interconnect fabric in order to connect each spare to each possible entity that shall be used. Therefore, next to the area overhead introduced by the redundant hardware blocks, a higher number of hardware blocks would also lead to larger interconnects of these crossbars. This may result in wiring congestion during the place and route phase, possibly rendering the design infeasible—an effect that has been demonstrated in recent literature where large crossbars do not converge from a physical design viewpoint [Pullini et al. 2007]. A further aspect when dealing with hardware redundancy is the event of hardware faults in the redundant blocks themselves and how to efficiently deal with such a situation. To the best of our knowledge, this has not been addressed for router microarchitecture design yet.

4.3. Modular Design Approaches and Slicing

An alternative to merely replicating logic blocks for covering faulty router components is to divide them into finer-grained pieces, so that a defect would only render a part of them faulty. This allows one to maintain correct operation along the router's data path at decreased performance, as opposed to shutting the entire router down.

RoCo [Kim et al. 2006] is a modular router design for a 2D Mesh that increases fault tolerance by employing decoupled parallel arbiters, dual compact crossbars arranged row and column path sets, and four distinct sets of VCs to support dedicated row and column routing in the two crossbars. This structure allows defective components to be replaced by others, thereby enabling one to continue partially correct operation instead of a complete breakdown. In doing so, they protect all different parts of the router and implement according functionality to enforce component replacement, leading to graceful performance degradation in the presence of permanent faults without additional area overhead imposed by using hardware redundancy. Apart from improving reliability, it also shows improvements regarding performance and energy. For a meaningful evaluation, they propose a Performance, Energy, and Fault-tolerance (PEF) metric, as depicted in Equation (3), which is the ratio of the product of the average latency (Latency) and Energy per packet (Energy) to the Packet Completion Probability P_c . They show 50% and 35% better PEF results than a generic baseline router and path-sensitive

router [Kim et al. 2005], respectively.

$$PEF = \frac{\text{Latency} \times \text{Energy}}{P_c}. \quad (3)$$

A similar approach was introduced by Liu et al. [2011], who split the data path of a router, and organized them as a group of workable subcomponents, capable of backing each other up. In case of faults, the packets are fragmented and transmitted over the functional parts of the data path using time division multiplexing, offering high reliability given that, as long as one data path slice is left, the router can maintain its operation. As the area of a module is assumed to be proportional to the likelihood that an error might occur, smaller logic blocks within the router are protected with conventional Error Correcting Schemes (ECCs), while larger components are sliced as described. Different numbers and combinations of data path slicing are implemented and evaluated, such as data path salvaging with two and four slices for all components (DPSR4 and DPSR2), or four slices for buffer and link, and two for the crossbar (DPSRM). Evaluation results show that DPSR4, DPSR2, and DPSRM cause 65.4%, 26.5%, and 52.46% area overhead, indicating that with an increasing number of slices, the area increases along with the fault-tolerance capability. In order to compare their design to other implementations, they compare the SPF factors of their different slicing variants to Vicis [DeOrio et al. 2012] and to routers using Triple Modular Redundancy (TMR). While the area overhead of the slicing variants are comparable to Vicis, their implementations, except DPSR2, have a much higher SPF factor.

Compared to hardware redundancy, these modularization and slicing techniques require much less area overhead but provide comparable reliability.

4.4. Dealing with Faulty Ports

The study of how to deal with faulty ports or the reaction to faulty links has also been investigated.

Neishaburi and Zilic [2009] proposed the RAVC router. Their design allows one to reassign buffering from a healthy port connected to a faulty link or router so that it can be used by other ports in the same router—for example, in Figure 12, if router 10 fails, the packets will be detoured around it to be sent from router 5 to 15. The VCs assigned to the ports pointing to the faulty router, that is, the south port of 6, west port of 9, east port of 11, and north port for 14, will be recaptured to support the other remaining ports pointing to healthy routers with their buffering resources. This provides more buffer resources for the surrounding routers, prevents the faulty router to occupy network bandwidth, isolates it, and employs buffering resources that would otherwise be wasted. Evaluation results with inserted faults and under uniform and transpose traffic show a decrease in average packet latency of 28% and 16%.

While the idea of recapturing channel resources is good, its implementation leads to infeasible hardware overhead. Therefore, ERAVC [Neishaburi and Zilic 2011] was proposed to improve this implementation. In this approach, they used a Unified Buffer Structure (UBS) and a History Aware Free-slot Tracker (HAFT) to implement dynamic VC allocation. In addition, they propose a new fault-tolerant flow control that facilitates packet resubmission in the case of faults without requiring any additional buffer resources. This design provides the reliability of RAVC with a more optimal design solution and flow control. They further extended their study of the ERAVC methodology to ensure reliability and deadlock-free operation in subnetworks of hierarchical NoCs with their NISHA router [Neishaburi and Zilic 2013].

QORE [DiTomaso et al. 2014] uses Quad-Function Channel (QFC) buffers, which can store and send data in both forward and backward direction. Their NoC proposal uses these QFC buffers as links, rather than regular links and input/output buffers within

Fig. 12. RAVC VC reassignment [Neishaburi and Zilic 2009].

Fig. 13. Port swapping technique [DeOrio et al. 2012].

the router, as in the conventional router designs. Having a few control signals, they can change the direction in which the buffers forward packets, which allows them to dynamically change these directions based on traffic patterns or fault status. Each link has two VCs and is narrower than the baseline links. The links are designed in a way that there are four QFCs between two routers, two in each direction by default. If there is a link fault so that one buffer can only forward a packet in one direction, another QFC can simply be reversed and thereby deal with the fault, without any performance loss. This way, a high number of faults can be contained. In addition, there is a backup ring network in case there are so many faults that a router could be isolated. Evaluation results show that QORE saves 15% of energy compared to a baseline router, and has 22.6% area overhead, however, this can be explained as QORE also provides 237.2% more storage than a baseline router. They compared their design to Vicis and Ariadne (see again Section 3.4.2). Their results show that with an increasing percentage of faults in the network, the QORE implementation is capable of preserving network connectivity much longer. A speedup of 1.3% and improved throughput by 2.3% on synthetic traffic is reported, which makes QORE an interesting new design approach for fault-tolerant networks.

4.5. Holistic Router Design Approaches

Literature also includes holistic design approaches with several protection strategies working in tandem.

Vicis [DeOrio et al. 2012] is a two-level approach addressing both router architecture and routing algorithm to enhance reliability. Within the router, they employ redundancy at the crossbar using a bypass path, ECCs to protect data path elements, and a new logic supporting reconfigurability of the pointers within the FIFO buffers. The latter is implemented by using equally sized registers, and in case a FIFO storage line is faulty, the read/write pointers are redirected in such a way that faulty registers are skipped. Moreover, they tackle faults on links or input ports by providing a port-swapping solution that is capable of reorganizing ports so that faulty input/output ports of neighbors are assigned to each other and swapped with functional ones, thereby minimizing the effect of link faults (see Figure 13). Together with this router design, they implemented a distributed routing algorithm that is capable of routing around defective routers and links in a 2D Mesh or Torus topology.

PFTR [Poluri and Louri 2013] improves reliability by adding minimum extra circuitry and exploiting temporal parallelism. They protect all pipeline stages and are capable of coping with two faults at the RC, SA, and XB stages, and four faults at the VA stage. In the RC stage, they employ redundancy by adding a second RC unit that will be turned on when the original one is faulty. In the VA stage, they deploy resource sharing by allowing a VC to use the arbiter of a neighboring one, in case its own arbiter

fails. For the SA stage, a bypass path for each $v : 1$ arbiter, where v is the number of VCs, is added that can be employed to select a VC when the arbiter is defective. For the XB stage, they provide two paths for every output port of the crossbar, which they implemented by adding smaller sized multiplexers and demultiplexers. In the worst case, two faults in the same pipeline stage (RC, SA, or XB) can render the router unusable. However, with random faults this design can tolerate up to 27 faults, in the best case. Their design has an area overhead of 31%, which they claim to lead to a SPF of 11—they use the mean of these two numbers, $(2 + 28)/2 = 15$, which is arbitrary and probably unfair.

4.6. Structure Retaining Design Approaches

Given the large impact of faulty router components on the performance of the NoC, some research efforts focused on how to retain the topology/structure of the NoC, even if a router is faulty, thereby avoiding traffic hot spots and containing performance degradation. Greenfield et al. [2007] suggest that the actual traffic patterns play an important role when considering fault tolerance and have to be analyzed to improve performance degradation. Their analysis of a dimension-ordered router showed that the dominant router activity was routing packets from one side to the opposite side, motivating them to investigate a through-mode capable of bypassing the defective router logic. They qualitatively showed for a 2D Mesh topology that reachability is much improved with their through-mode router, with an area increase of just 5.6%.

MiCoF [Ebrahimi et al. 2013b] deals with the idea of retaining the structure of a NoC for a 2D Mesh by implementing a router that simply forwards a packet to the opposite direction in case the router logic or crossbar is faulty. A defective router can then be considered as a long wire, simply forwarding incoming packets. In this manner, no packets need to be routed around faulty routers, which simplifies the routing algorithm and maintains the original performance level by allowing one to take the shortest path between each source-destination pair. Their fully adaptive routing algorithm only requires the fault status of adjacent routers to find the optimal route, and needs only one VC along the X axis, and two VCs along the Y axis, which is the minimal number required for a fully adaptive routing algorithm in a 2D Mesh. Considering an 8×8 Mesh and uniform traffic, they result in 99.5% successfully delivered packets when up to six faulty routers are present, regardless of their position.

4.7. Link Error Containment

Much research on fault-tolerant microarchitecture focuses on link errors, in particular how partially faulty links can be kept functioning.

Lehtonen et al. [2007] proposed two different techniques to deal with permanent link errors. The first approach comprises the deployment of hardware redundancy by using four spare wires for 64-bit-wide links in combination with a reconfiguration circuitry so that in case of wire faults, the affected wires can be replaced by the spare ones. In the second approach, they split the data line in four equally sized sections, duplicate each of them, and send two interleaving sections and their duplications in each transmission, thus splitting the transmission into two parts. In addition, they use Hamming encoding, so that the receiver can identify when a link in one of the sections is faulty by comparing it to the duplicate. Their evaluation showed that latency is higher for split transmission (31%) than the spare wire approach (15%), and energy consumption increases by 20% and 15%, respectively. This can be explained by the fact that two transmissions are required for the split transmission, negatively affecting performance and energy metrics. However, the spare wire approach has a higher area overhead (105%) compared to split transmission (75%), as basically hardware redundancy is employed.

Table III. Comparison of Splitting Methods [Zhang et al. 2012]

Design	Area (%)	Power (%)	Throughput at 30 Faults (%)
Baseline	100	100	N/A
1/2 Splitting	116.6	133.1	52.68
1/4 Splitting	128.5	155.5	82.20
1/8 Splitting	150.1	199.4	89.89

Yu and Ampadu [2011] proposed Simple Flit Quad Splitting (SFQS), a design using split transmission through a comanagement between a physical layer and a data link layer. This comprises an ECC module that can switch from a powerful ECC to a simpler one, based on the current noise condition. Power ECCs require more wires, which is why some wires remain unused if only an easy ECC is needed. In this case, the free wires can be used as spare wires in case of defects. When both permanent faults and a high noise level are present, the transmission is split into two parts. Along with a new packet reorganization algorithm, they achieve a reduction of packet latency and energy per useful packet by 50% and 70%, respectively, compared to previously proposed methods.

Chen et al. [2012] introduced a novel flit serialization strategy, improving previous splitting proposals. Links and flits are divided into several sections, and flit sections of adjacent flits are serialized in order to send them over all currently available healthy link sections. According serialization modules in both transmitter and receiver are aware of the current link status, and perform, based on this information, either serialization or enforce the bypassing of this module in case of the absence of errors. Split sizes of four (S4) and eight (S8) are proposed. Compared to the spare wire approach, both sizes show better power and area characteristics, with S8 having higher requirements than S4. In addition, their evaluation shows that both implementations need more energy and area, but feature significantly less average packet latency than SFQS and PFLRM (see the following). Higher error rates lead to larger latency differences, with S4 being more effective than S8.

A more exhaustive evaluation of splitting methods was conducted by Zhang et al. [2012], who proposed router designs implementing fine-grained splitting methods, such as half-splitting, 1/4 splitting, and 1/8 splitting. Their results are listed in Table III, and have been obtained using 65nm CMOS technology and simulation of a 4×4 Mesh with 32-bit phits. The results show that with increasing granularity the sustained throughput under 30 randomly injected link faults increases at the cost of the area and energy costs increasing as well. This clearly outlines a trade-off that has to be made between area, power, and throughput. In addition, they compared the throughput of these splitting methods to a fault-tolerant routing algorithm detouring faulty links instead of using partially faulty links. Their results show that all splitting methods significantly outperform fault-tolerant routing, and that indeed, the difference becomes even more obvious as the number of faults increases.

Partially Faulty Link Recovery Mechanism (*PFLRM*) [Vitkovskiy et al. 2010] is capable of detecting and correcting flits transmitted over a partially faulty link, rather than containing permanent link faults with transmission splitting. After error detection, the receiver generates a fault vector and mask vectors, and requests the sender to retransmit the affected flit. Along with the vectors and retransmitted flits, the receiver is now capable of correcting the incoming flit. The higher the maximum error cluster, which is the number of adjacent link faults, the higher the latency. The overhead caused by additional logic was shown to be 36% compared to a baseline router. This is acceptable under the consideration that an arbitrary number of link faults can be corrected, which

allows a maximum connectivity in the network, and graceful performance degradation is provided.

4.8. High Radix and 3D Switches

While many of the introduced proposals were either particularly designed for mesh or torus, or evaluated only in these topologies, fault-tolerant router microarchitectures for 3D NoCs or high-radix switches were also studied.

LBDRhr [Yu et al. 2012] addresses fault tolerance in networks containing high-radix switches with an underlying Mesh topology, such as the Flattened Butterfly and Mesh with express channels. They consider high-radix switches with one-hop, two-hop, and three-hop ports for according links connecting to other routers that are further away than one hop. Conventional routing tables in the switches are replaced by a small set of hardwired bits. These are a few configuration bits that store local information about the status of the neighboring links, in particular eight bits for routing in each cardinal direction (including south-east, etc.), two bits at each input port for deroute options, and one connectivity bit for each output port. Based on a neighbor's status, packets are derouted over other links, including the higher hop links, to reach their destination. Fault-tolerant routing without the use of routing tables allows area and energy savings. Their evaluation shows that their design is capable of handling all errors in long-range links, and 80% of the fault combinations of the one-hop links.

Loi et al. [2008] proposed a robust link architecture for Through Silicon Vias (TSVs) to improve the yield of the TSV fabrication process. They employ the hardware redundancy approach in the form of spare pads in combination with reconfigurable routing hardware and dynamic routing. This method improves yield from 66% to 98% with a 17% overhead in terms of silicon per TSV link in 130nm technology. They predict that this overhead will increase to 25% at the 65nm technology node.

4.9. Discussion

This section covered the research on microarchitecture-level fault tolerance. The community has explored robust designs for all the different parts of a router. Using redundancy is the simplest approach, but is less efficient than other strategies, such as modular router design or data path slicing.

Protecting router ports to avoid wasting resources or even complete isolation of PEs is one of the main fault-tolerance approaches and comprises techniques as varied as buffer reassignment, port swapping, or bidirectional buffers.

Protecting the routing/switching logic is another area of interest and includes techniques such as temporary parallelism, redundancy, and slicing. Preserving the original structure of a network by supporting partial operation is more efficient than shutting the router down completely and routing around it. Structure preservation can either be ensured by using DBPs or simply by bypassing the router completely.

Handling link faults and maintaining operation by using partially functional links has been thoroughly studied, too. Faulty links are dealt with by many techniques such as using spare wires to replace faulty ones, splitting transmission, or recovery schemes employing rotation techniques to circumvent wire errors.

We can conclude that router designs capable of retaining the NoC structure seem to be an appropriate choice, as it allows graceful performance degradation without requiring complex routing algorithms or imposing significant overhead. Optionally, a fault-tolerant routing algorithm could be implemented on top of that in case a router failed completely, which would provide even more protection. Some holistic router designs were introduced along with metrics to measure the ratio between the overheads and the protection level of different techniques.

Fig. 14. YAT using redundancy [Zhang et al. 2008a].

While major parts of the studies were focusing on 2D Mesh networks, only very little research was dedicated to other topologies, such as high radix or 3D NoCs. Given the superior performance and energy metrics of some topologies, such as the Flattened Butterfly, studies on how to retain these characteristics if faults occur would be interesting and could prepare the way for their industrial adoption.

5. RESEARCH ON SYSTEM-LEVEL REDUNDANCY AND RECONFIGURATION

While design approaches at the microarchitectural level can effectively be employed to extend the operation of the NoC, similar protection can be taken up to the system level as well. Research efforts on the system level comprise the use of redundant building blocks, such as links, routers, or PEs, and according NoC reconfiguration. In this context, the question on how to make use of redundant blocks and how to efficiently integrate them is addressed.

5.1. Spare Cores

When PEs fail, adjustments to the underlying NoC are required to maintain throughput levels. Common approaches used redundancy at the microarchitectural level, which is appropriate when the number of PEs on the chip is rather small. However, these numbers have been increasing to a point at which a single core becomes inexpensive compared to the entire many-core system. Therefore, it has become more appropriate to use core-level redundancy for fault tolerance in order to minimize the complexity of redundancy at the microarchitectural level [Zhang et al. 2008a]. The effectiveness of different redundancy schemes is commonly evaluated using the Yield-Adjusted Throughput (YAT), a metric that shows the average chip throughput when a large number of chips is fabricated.

As depicted in Figure 14, with shrinking transistor sizes, there is a crossover point at around 100nm feature size [Zhang et al. 2008a], at which core-level redundancy will result in a better YAT than microarchitectural redundancy. Two schemes for many-core processors exist, which are *As Many As Available (AMAA)* and *As Many As Demand (AMAD)*. The former, as adopted in Sun's Ultra-SPARC T1 processors [Zhang et al. 2008a], degrades a chip's performance by disabling faulty cores only. In the AMAD scheme, there can also be fault-free cores left unused as long as all original cores are still functional. A 3×3 Mesh network with three additional back-up spare cores is shown in Figure 15.

NoC design approaches are chosen based on the employed scheme. Given the outlined benefits of core-level redundancy, similar measures can be adopted for on-chip interconnects in order to efficiently integrate back-up cores into the NoC. Zhang et al. [2008a] studied the problem of reconfiguring the network topology of a NoC when PEs are defective, so that the most efficient topology regarding performance and

Fig. 15. AMAD Layout.

Fig. 16. Spare Router Architecture [Chang et al. 2011].

energy consumption is chosen. To choose the optimal topology in the reconfiguration process, they propose the heuristic *RRCS-GSA*. Their experimental results show significant performance boost of the many-core processor compared to replacing faulty cores randomly. This shows that the choice of the healthy PE that shall replace a faulty one, and the resulting topology have a significant impact on the overall performance.

Khaleghi and Rao [2013] addressed the problem of how to efficiently integrate spare cores in the original NoC by implementing a spare sharing network containing a small number of connections to the spare cores. Their integration process also considered router faults, so that router faults do not prevent the utilization of healthy PEs. Their results show significant reliability improvement of the NoC, with a cost-efficient implementation that only requires a few extra link resources to form the spare unit network.

5.2. Spare Routers and Links

Rather than making modifications on the microarchitectural level of routers and links, some research focused on router-level solutions, mainly using spare routers or links. Refan et al. [2008] proposed a design based on additional spare interconnects between each PE and one of its neighboring routers, so that a faulty switch does not isolate a PE from the network. The best spare links for each PE are chosen among different possibilities, for example, in a 2D Mesh inner nodes have eight options. Application-specific performance analysis is performed before the NoC is programmed. Evaluation shows that this approach has a hardware and power consumption overhead, but at the same time significantly improves reliability while decreasing performance by 2%.

Chang et al. [2011] uses spare routers both as redundancies and to diversify paths between adjacent routers. Two configuration algorithms are introduced, SARA and DAPA, that take advantage of this path diversity. The design is applicable to any routing algorithm for a 2D Mesh topology as the output topology after the configuration is consistent to the original mesh. The architecture is composed as illustrated in Figure 16, in which SR stands for the Spare Routers. The ROM blocks are responsible for the reconfiguration of the network. Their evaluation results show remarkable improvements regarding fault tolerance, including reliability, Mean Time to Failure, and yield. The yield and reliability gain of this approach increases for larger NoC sizes, and relative connection cost decreases at the same time. This makes it suitable for large-scale NoCs.

Quad-spare mesh [Ren et al. 2013] is a fault-tolerant architecture, based on 2×2 submeshes with one spare router in the middle, as in Figure 17(a). This topology can be reconfigured by changing control signals when a router fails without altering the underlying topology (thus retaining the original 2D Mesh)—see Figure 17(b). Along with this architecture, a dynamic reconfiguration technique and routing algorithm are presented. Evaluation results with a 4×4 Mesh topology show that if one router fails,

Fig. 17. Quad-spars Mesh Topology [Ren et al. 2013].

throughput only decreases by 5.19% and latency increases by 2.4%. The hardware overhead caused by additional ports in the original routers in order to connect to the spare routers, multiplexers, and the actual spare router is estimated to be approximately 50%.

R-3PO [Morris et al. 2012] is a 3D optical NoC consisting of 16 decomposed photonic interconnect based crossbars placed on four optical communication layers. They propose a reconfiguration strategy that bypasses faulty channels by adapting available network bandwidth by multiplexing signals on crossbar channels that are either idle or healthy. This is enabled by having additional disconnected waveguides that can be used for reconfiguration and a crossbar consisting of several smaller crossbars. Reconfiguration runs in the background, does not impede network operation, and allows graceful performance degradation. This has, to the best of our knowledge, been the only study on architectural approach to deal with faults in networks featuring photonic links.

5.3. Discussion

This section reviewed research on system-level redundancy. In particular, topology reconfiguration as a reaction to faulty network elements was discussed, as well as the most efficient use of spare routers and links to support fault tolerance. Moreover, the use of spare cores coupled with their integration into existing NoCs was covered. Redundancy at this level is particularly efficient for small technology nodes enabling a high number of cores on a chip. However, for a small number of nodes, microarchitectural approaches for providing fault tolerance are preferred, especially when there are stringent area constraints that do not allow the duplication of switches or PEs. The vast majority of the proposals were dealing with the 2D Mesh and were trying to retain this structure after configuration. The efficient use of system-level redundancy would also be of high interest for other topologies, in particular for more scalable topologies than the mesh, in which system-level redundancy would be efficient given the large number of on-chip modules.

6. CONCLUSION

Fault tolerance continues to be an important issue in NoC design as underlying technologies are getting less and less reliable. The integral role of NoCs in many-core chips regarding performance and energy consumption stresses this even further. In this article, we reviewed recent publications on fault-tolerant NoC design, focusing on their components, namely, topology, router microarchitecture, routing algorithms, as well as system-level approaches. We further evaluated recent work based on their relevance, feasibility, covered areas, and tried to identify possible future directions.

First, we discussed topologies inherently robust to link and node faults, such as DB graphs, RDT, and SFHGs. While enabling design solutions were introduced, there are still many optimizations in these topologies to address, such as efficient fault-tolerant

routing, router architecture, or floorplanning. Highly related to topologies is routing and so, we discussed its different types, such as deterministic, adaptive, and stochastic routing, and their pros and cons for fault tolerance. We also analyzed the effects of VCs on routing complexity and use of resources in the presence of permanent faults.

We continued our discussion by looking at router and link microarchitecture. In particular, their effects on the correct operation of the network were considered together with according fault containment. While router designs for the 2D Mesh topologies have been thoroughly investigated, effects of faults in routers in other topologies have not been extensively studied yet.

We closed our discussion by exploring how redundancy of NoC elements (routers or links) and cores can be exploited at the system level by means of reconfiguration in the presence of faults. This comprises the task of efficiently using and integrating redundant building blocks, while using minimal additional resources and maintaining or gracefully degrading system performance and energy consumption.

In general, we found that the 2D Mesh topology gained a lot of attention regarding fault tolerance, given its easy VLSI implementation and routing. On the other hand, in more advanced topologies superior to the 2D Mesh in terms of performance and energy consumption, effects of permanent faults in links or nodes have barely been considered. As the power dissipated on links is estimated by the ITRS to be over 70% of the entire chip [Committee 2014], scalable solutions not exceeding the power budget are required, and the 2D Mesh does not constitute an efficient scalable topology. Further research in scalable, fault-tolerant NoC design, and more efficient topologies is thus required, along with according routing algorithms and router microarchitecture.

Wireless and optical NoCs have been recently introduced as emerging technologies to circumvent the power wall inherent to electric wiring by providing higher bandwidth at lower energy consumption. Given that many on-chip antenna technologies, such as carbon nanotube based antennas [Nojeh et al. 2008], are in their infancy and thus fault-prone, robust NoC designs using wireless communication should be explored to achieve yield improvements. As wireless links would generally be beneficial to replace long wires that are present, for example, in Torus or Flattened Butterfly topologies, findings of research that focused on robust routing algorithms in these topologies could be adopted. Some of the introduced fault-tolerant topologies featuring long wires or many wire crossings that preclude them from being implemented in VLSI, may be enabled by these technologies.

The use of Optical NoCs (oNoCs) or hybrid NoCs introduces a paradigm shift in how routing is performed, which design challenges occur, and which resources are valuable or costly. Hybrid NoCs in this context are those with both electrical and optical components. ONoCs based on passive microring resonators seem to be the most efficient solution [Hamedani et al. 2014]. In this methodology, packets are routed based on their wavelength, in which each destination has a dedicated wavelength assigned to it. Network topology and routing algorithm can be designed in a way that avoids contention inherently, or additional contention handling mechanisms could be employed, which is, however, a less efficient solution as they introduce overhead. Permanent link or router faults could thus cause contention in a network that was designed to be contention-free. The study of how to preserve contention-free operation in the presence of faults constitutes an interesting field for future work. Also, the study of how to minimize the cost of these NoCs while providing fault tolerance is a research field that has, to the best of our knowledge, not been addressed in the literature. The investigation of how to employ the techniques that have been reviewed in this article for fault tolerance in electrical, wired based NoCs in oNoCs, and to what extent they are adoptable, may be significant for future NoC fabrics.

REFERENCES

- Akram Ben Ahmed and Abderazek Ben Abdallah. 2014. Graceful deadlock-free fault-tolerant routing algorithm for 3D network-on-chip architectures. *J. Parallel Distrib. Comput.* 74, 4 (2014), 2229–2240.
- Konstantinos Aisopos, Andrew DeOrio, Li-Shiuan Peh, and Valeria Bertacco. 2011. Ariadne: Agnostic reconfiguration in a disconnected network environment. In *PACT'11*. IEEE, 298–309.
- Sara Akbari, Ali Shafee, Mahmoud Fathy, and Reza Berangi. 2012. AFRA: A low cost high performance reliable routing for 3D mesh NoCs. In *DATE'12*. IEEE, 332–337.
- Muhammad Ali, Michael Welzl, and Sven Hessler. 2007. A fault tolerant mechanism for handling permanent and transient failures in a network on chip. In *ITNG'07*. IEEE, 1027–1032.
- Joel Baker. 2011. De Bruijn graphs and their applications to fault tolerant networks. Oregon State University.
- Marco Balboni, José Flich, and Davide Bertozzi. 2015. Synergistic use of multiple on-chip networks for ultra-low latency and scalable distributed routing reconfiguration. In *DATE'15*. EDA Consortium, 806–811.
- Shane Bell, Bruce Edwards, John Amann, Rich Conlin, Kevin Joyce, Vince Leung, John MacKay et al. 2008. Tile64-processor: A 64-core soc with mesh interconnect. In *ISSCC'08*. IEEE, 88–598.
- Christian Bienia and Kai Li. 2011. *Benchmarking Modern Processors*. Ph.D. dissertation, Princeton University.
- Rimpy Bishnoi, Vijay Laxmi, Manoj Singh Gaur, and José Flich. 2015. D 2-LBDR: Distance-driven routing to handle permanent failures in 2D mesh NOCs. In *DATE'15*. EDA Consortium, 800–805.
- Fabien Chaix, Dimitar Avresky, Nacer-Eddine Zergainoh, and Michael Nicolaidis. 2010. Fault-tolerant deadlock-free adaptive routing for any set of link and node failures in multi-cores systems. In *NCA'10*. IEEE, 52–59.
- Yung-Chang Chang, Ching-Te Chiu, Shih-Yin Lin, and Chung-Kai Liu. 2011. On the design and analysis of fault tolerant NoC architecture using spare routers. In *ASP-DAC'11*. IEEE, 431–436.
- Changlin Chen, Ye Lu, and Sorin D. Cotofana. 2012. A novel flit serialization strategy to utilize partially faulty links in networks-on-chip. In *NOCS'12*. IEEE, 124–131.
- Yiou Chen, Jianhao Hu, and Xiang Ling. 2009. De Bruijn graph based 3D network on chip architecture design. In *International Conference on Communications, Circuits and Systems (ICCCAS'09)*. IEEE, 986–990.
- Yiou Chen, Jianhao Hu, and Xiang Ling. 2006. A recursively construction scheme for super fault-tolerant Hamiltonian graphs. *Appl. Math. Comput.* 177, 2 (2006), 465–481.
- George Chrysos. 2014. Intel® xeon phi coprocessor—The architecture. *Intel Whitepaper* (2014).
- Jacques Henri Collet, Ahmed Louri, Vivek Tulsidas Bhat, and Pavan Poluri. 2011. ROBUST: A new self-healing fault-tolerant NoC router. In *NoCArc'11*. ACM, 11–16.
- International Roadmap Committee. 2014. International technology roadmap for semiconductors. (2014). www.itrs.net.
- Cristian Constantinescu. 2003. Trends and challenges in VLSI circuit reliability. *MICRO'03* 4 (2003), 14–19.
- Kypros Constantinides, Stephen Plaza, Jason Blome, Bin Zhang, Valeria Bertacco, Scott Mahlke, Todd Austin, and Michael Orshansky. 2006. Bulletproof: A defect-tolerant CMP switch architecture. In *HPCA'06*. IEEE, 5–16.
- William J. Dally and Hiromichi Aoki. 1993. Deadlock-free adaptive routing in multicomputer networks using virtual channels. *TPDS'93* 4, 4 (1993), 466–475.
- William James Dally and Brian Patrick Towles. 2004. *Principles and Practices of Interconnection Networks*. Elsevier.
- Giovanni De Micheli and Luca Benini. 2006. *Networks on Chips: Technology and Tools*. Academic Press.
- Andrew DeOrio, David Fick, Valeria Bertacco, Dennis Sylvester, David Blaauw, Jin Hu, and Gregory Chen. 2012. A reliable routing architecture and algorithm for NoCs. *IEEE Trans. Comput.-Aided Design Integr. Circ. Syst.* 31, 5 (2012), 726–739.
- Giorgos Dimitrakopoulos and Emmanouil Kalligeros. 2012. Low-cost fault-tolerant switch allocator for network-on-chip routers. In *INA-OCMC'12*. ACM, 25–28.
- Michael Dimopoulos, Yi Gang, Lorena Anghel, Mounir Benabdenbi, Nacer-Eddine Zergainoh, and Michael Nicolaidis. 2014. Fault-tolerant adaptive routing under an unconstrained set of node and link failures for many-core systems-on-chip. *Microprocess. Microsyst.* 38, 6 (August 2014), 620–635.
- Dominic DiTomaso, Avinash Kodi, and Ahmed Louri. 2014. QORE: A fault tolerant network-on-chip architecture with power-efficient quad-function channel (QFC) buffers. *HPCA'14* (2014).
- Ding-Zhu Du, D. Frank Hsu, Frank K. Hwang, and X. M. Zhang. 1991. The Hamiltonian property of generalized de Bruijn digraphs. *J. Combin. Theory, Ser. B* 52, 1 (1991), 1–8.

- Xinming Duan, Yulu Yang, Mei Yang, Lei Li, and Yingtao Jiang. 2007. Topology and binary routing schemes of a PRDT-based NoC. In *ITNG'07*. IEEE, 920–924.
- José Duato. 1993. A new theory of deadlock-free adaptive routing in wormhole networks. *IEEE Trans. Parallel Distrib. Syst.* 4, 12 (1993), 1320–1331.
- José Duato. 1995. A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks. *IEEE Trans. Parallel Distrib. Syst.* 6, 10 (1995), 1055–1067.
- Benoît Dupont de Dinechin, Pierre Guironnet de Massas, Guillaume Lager, Clément Léger, Benjamin Orgozgo, Jérôme Reybert, and Thierry Strudel. 2013. A distributed run-time environment for the Kalray MPPA®-256 integrated manycore processor. *Proc. Comput. Sci.* 18 (2013), 1654–1663.
- Mojtaba Ebrahimi, Masoud Daneshmand, and Juha Plosila. 2013a. High performance fault-tolerant routing algorithm for NoC-based many-core systems. In *PDP'13*. IEEE, 462–469.
- Mojtaba Ebrahimi, Masoud Daneshmand, Juha Plosila, and Hannu Tenhunen. 2013b. Minimal-path fault-tolerant approach using connection-retaining structure in networks-on-chip. In *NOCS'13*. IEEE, 1–8.
- Qin Fan, Yulu Yang, Akira Funahashi, and Hideharu Amano. 1999. A torus assignment for an interconnection network recursive diagonal torus. In *I-SPAN'99*. IEEE, 74–79.
- David Fick, Andrew DeOrio, Gregory Chen, Valeria Bertacco, Dennis Sylvester, and David Blaauw. 2009a. A highly resilient routing algorithm for fault-tolerant NoCs. In *DATE'09*. European Design and Automation Association, 21–26.
- David Fick, Andrew DeOrio, Jin Hu, Valeria Bertacco, David Blaauw, and Dennis Sylvester. 2009b. Vicis: A reliable network for unreliable silicon. In *DAC'09*. ACM, 812–817.
- Jose Flich and Davide Bertozzi. 2010. *Designing Network on-Chip Architectures in the Nanoscale Era*. CRC Press.
- Jose Flich and Jose Duato. 2008. Logic-based distributed routing for NoCs. *Comput. Architect. Lett.* 7, 1 (2008), 13–16.
- Binzhang Fu, Yinhe Han, Huawei Li, and Xiaowei Li. 2014. Zonedefense: A fault-tolerant routing for 2-D meshes without virtual channels. *IEEE Trans. Very Large Scale Integr. Syst.* 22, 1 (2014), 113–126.
- Yusuke Fukushima, Masaru Fukushi, and Susumu Horiguchi. 2009. Fault-tolerant routing algorithm for network on chip without virtual channels. In *DFT'09*. IEEE, 313–321.
- Elango Ganesan and Dhiraj K. Pradhan. 2003. Wormhole routing in de Bruijn networks and hyper-de Bruijn networks. In *ISCAS'03*, Vol. 3. IEEE, III–870.
- Amlan Ganguly, Paul Wettin, Kevin Chang, and Partha Pande. 2011. Complex network inspired fault-tolerant NoC architectures with wireless links. In *NOCS'11*. ACM, 169–176.
- Alberto Ghiribaldi, Daniele Ludovici, Francisco Triviño, Alessandro Strano, José Luis Sánchez, Francisco Alfaro, Michele Favalli, and Davide Bertozzi. 2013. A complete self-testing and self-configuring NoC infrastructure for cost-effective MPSoCs. *TECS'13* 12, 4 (2013), 106.
- Paul Gillard and Cheng Li. 2011. Network-on-chip (NoC) topologies and performance: A review. (2011), 1–6.
- Christopher J. Glass and Lionel M. Ni. 1992. The turn model for adaptive routing. In *ACM SIGARCH Comput. Architect. News* 20 (1992), 278–287.
- Daniel Greenfield, Arnab Banerjee, Jeong-Gun Lee, and Simon Moore. 2007. Implications of rent's rule for NoC design and its fault-tolerance. In *NOCS'07*. IEEE, 283–294.
- Michael Gschwind, H. Peter Hofstee, Brian Flachs, Martin Hopkins, Yukio Watanabe, and Takeshi Yamazaki. 2006. Synergistic processing in cell's multicore architecture. *IEEE Micro* 26, 2 (2006), 10–24.
- Parisa Khadem Hamedani, Natalie Enright Jerger, and Shaahin Hessabi. 2014. QuT: A low-power optical network-on-chip. In *NOCS'14*. IEEE, 80–87.
- Mohammad Hosseinabadi, Mohammad Reza Kakoei, Jimson Mathew, and Dhiraj K. Pradhan. 2007. Reliable network-on-chip based on generalized de Bruijn graph. In *HLVDT'07*. IEEE, 3–10.
- Mohammad Hosseinabadi, Mohammad Reza Kakoei, Jimson Mathew, and Dhiraj K. Pradhan. 2008. De Bruijn graph as a low latency scalable architecture for energy efficient massive NoCs. In *DATE'08*. ACM, 1370–1373.
- Mohammad Hosseinabadi, Mohammad Reza Kakoei, Jimson Mathew, and Dhiraj K. Pradhan. 2011. Low latency and energy efficient scalable architecture for massive NoCs using generalized de Bruijn graph. *IEEE Trans. Very Large Scale Integr. Syst.* 19, 8 (2011), 1469–1480.
- Mohammad Reza Kakoei, Valeria Bertacco, and Luca Benini. 2011. A distributed and topology-agnostic approach for on-line NoC testing. In *NOCS'11*. ACM, 113–120.
- John Keane, Shrinivas Venkatraman, Paulo Butzen, and Chris H. Kim. 2011. An array-based test circuit for fully automated gate dielectric breakdown characterization. *IEEE Trans. Very Large Scale Integr. Syst.* 19, 5 (2011), 787–795.

- Soroush Khaleghi and Wenjing Rao. 2013. Spare sharing network enhancement for scalable systems. In *DFT'13*. IEEE, 249–254.
- Jongman Kim, Dongkook Park, Theo Theodorides, Narayanan Vijaykrishnan, and Chita R. Das. 2005. A low latency router supporting adaptivity for on-chip interconnects. In *DAC'05*. ACM, 559–564.
- Jongman Kim, Chrysostomos Nicopoulos, Dongkook Park, Vijaykrishnan Narayanan, Mazin S. Younis, and Chita R. Das. 2006. A gracefully degrading and energy-efficient modular router architecture for on-chip networks. In *ACM SIGARCH Comput. Architect. News*, 34 (2006), 4–15.
- John Kim, James Balfour, and William Dally. 2007. Flattened butterfly topology for on-chip networks. In *MICRO'07*. IEEE Computer Society, 172–182.
- Young Bok Kim and Yong-Bin Kim. 2007. Fault tolerant source routing for network-on-chip. In *DFT'07*. IEEE, 12–20.
- Michihiro Koibuchi, Hiroki Matsutani, Hideharu Amano, and Timothy Mark Pinkston. 2008. A lightweight fault-tolerant mechanism for network-on-chip. In *NOCS'08*. IEEE Computer Society, 13–22.
- Nasser Kurd, Mashrur Chowdhury, Edward Burton, Thomas P. Thomas, Christopher Mozak, Brent Boswell, Manoj Lal, Anant Deval, James Douglas, Mahmoud Ellassal, Ankireddy Nalamalpu, Timothy M. Wilson, Matthew Merten, Srinivas Chennupaty, Wilfred Gomes, and Ravindra Kumar. 2014. 5.9 Haswell: A family of IA 22nm processors. In *ISSCC'14*. IEEE, 112–113.
- Doowon Lee, Ritesh Parikh, and Valeria Bertacco. 2014. Brisk and limited-impact NoC routing reconfiguration. In *DATE'14*. IEEE, 1–6.
- Teijo Lehtonen, Pasi Liljeberg, and Juha Plosila. 2007. Online reconfigurable self-timed links for fault tolerant NoC. *VLSI design* 2007 (2007).
- Louis H. Liang. 1996. Methods and apparatus for test and burn-in of integrated circuit devices. (July 2 1996). US Patent 5,532,612.
- Cheng Liu, Lei Zhang, Yinhe Han, and Xiaowei Li. 2011. A resilient on-chip router design through data path salvaging. In *ASP-DAC'11*. IEEE, 437–442.
- Igor Loi, Subhasish Mitra, Thomas H. Lee, Shinobu Fujita, and Luca Benini. 2008. A low-overhead fault tolerance scheme for TSV-based 3D network on chip links. In *ICCAD'08*. IEEE, 598–602.
- Igor Loi, Federico Angiolini, Shinobu Fujita, Subhasish Mitra, and Luca Benini. 2011. Characterization and implementation of fault-tolerant vertical links for 3-D networks-on-chip. *IEEE Trans. Comput.-Aided Design Integr. Circ. Syst.* 30, 1 (2011), 124–134.
- Sumit D. Mediratta and Jeffrey Draper. 2007. Characterization of a fault-tolerant NoC router. In *ISCAS'07*. IEEE, 381–384.
- Andres Mejia, Jose Flich, Jose Duato, Sven-Arne Reinemo, and Tor Skeie. 2006. Segment-based routing: An efficient fault-tolerant routing algorithm for meshes and tori. In *IPDPS'06*. IEEE, 10 pp.
- Andres Mejia, Maurizio Palesi, José Flich, Shashi Kumar, Pedro López, Rickard Holmström, and José Duato. 2009. Region-based routing: A mechanism to support efficient routing algorithms in NoCs. *IEEE Trans. Very Large Scale Integr. Syst.* 17, 3 (2009), 356–369.
- Randy Morris, Avinash Karanth Kodi, and Ahmed Louri. 2012. Dynamic reconfiguration of 3D photonic networks-on-chip for maximizing performance and improving fault tolerance. In *MICRO'12*. IEEE Computer Society, 282–293.
- Mohammad Hossein Neishaburi and Zeljko Zilic. 2009. Reliability aware NoC router architecture using input channel buffer sharing. In *GLSVLSI'09*. ACM, 511–516.
- Mohammad Hossein Neishaburi and Zeljko Zilic. 2011. ERAVC: Enhanced reliability aware NoC router. In *SQED'11*. IEEE, 1–6.
- Mohammad Hossein Neishaburi and Zeljko Zilic. 2013. NISHA: A fault-tolerant NoC router enabling deadlock-free interconnection of subnets in hierarchical architectures. *J. Syst. Architect.* 59, 7 (2013), 551–569.
- A. Nojeh, P. Pande, A. Ganguly, S. Sheikhaei, B. Belzer, and A. Ivanov. 2008. Reliability of wireless on-chip interconnects based on carbon nanotube antennas. In *IMS3TW'08*. IEEE, 1–6.
- John D. Owens, William J. Dally, Ron Ho, D. N. Jay Jayasimha, Stephen W. Keckler, and Li-Shiuan Peh. 2007. Research challenges for on-chip interconnection networks. *MICRO'07* (2007), 96–108.
- Yan Pan, Prabhat Kumar, John Kim, Gokhan Memik, Yu Zhang, and Alok Choudhary. 2009. Firefly: Illuminating future network-on-chip with nanophotonics. In *ACM SIGARCH Comput. Architect. News* 37 (2009), 429–440.
- Ritesh Parikh and Valeria Bertacco. 2011. Formally enhanced runtime verification to ensure NoC functional correctness. In *MICRO'11*. ACM, 410–419.
- Ritesh Parikh and Valeria Bertacco. 2013. Udirec: Unified diagnosis and reconfiguration for frugal bypass of NoC faults. In *Micro'13*. ACM, 148–159.

- Hyunmin Park and Dharma P. Agrawal. 1995. A novel deadlock-free routing technique for a class of de Bruijn graph based networks. In *IPDPS'95*. IEEE, 524–531.
- Seungjin Park, Jong-Hoon Youn, and Bella Bose. 2000. Fault-tolerant wormhole routing algorithms in meshes in the presence of concave faults. In *IPDPS'00*. IEEE, 633–638.
- Sudeep Pasricha and Yong Zou. 2011. A low overhead fault tolerant routing scheme for 3D Networks-on-Chip. In *ISQED'11*. IEEE, 1–8.
- Pavan Poluri and Ahmed Louri. 2013. Tackling permanent faults in the network-on-chip router pipeline. In *SBAC-PAD'13*. IEEE, 49–56.
- Antonio Pullini, Federico Angiolini, Srinivasan Murali, David Atienza, Giovanni De Micheli, and Luca Benini. 2007. Bringing NoCs to 65 nm. *IEEE Micro* 27, 5 (2007), 75–85.
- Martin Radetzki, Chaochao Feng, Xueqian Zhao, and Axel Jantsch. 2013. Methods for fault tolerance in networks-on-chip. *CSUR* 46, 1 (2013), 8.
- Amir-Mohammad Rahmani, Pasi Liljeberg, Khalid Latif, Juha Plosila, Kameswar Rao Vaddina, and Hannu Tenhunen. 2011. Congestion aware, fault tolerant, and thermally efficient inter-layer communication scheme for hybrid NoC-bus 3D architectures. In *NOCS'11*. IEEE, 65–72.
- Fatemeh Refan, Homa Alemzadeh, Saeed Safari, Paolo Prinetto, and Zainalabedin Navabi. 2008. Reliability in application specific mesh-based NoC architectures. In *IOLTS'08*. IEEE, 207–212.
- Pengju Ren, Qingxin Meng, Xiaowei Ren, and Nanning Zheng. 2014. Fault-tolerant routing for on-chip network without using virtual channels. In *DAC'14*. ACM, 1–6.
- Yu Ren, Leibo Liu, Shouyi Yin, Jie Han, Qinghua Wu, and Shaojun Wei. 2013. A fault tolerant NoC architecture using quad-spare mesh topology and dynamic reconfiguration. *J. Syst. Architect.* 59, 7 (2013), 482–491.
- Samuel Rodrigo, Jose Flich, Antoni Roca, Simone Medardoni, Davide Bertozzi, J. Camacho, Federico Silla, and Jose Duato. 2010. Addressing manufacturing challenges with cost-efficient fault tolerant routing. In *NOCS'10*. IEEE, 25–32.
- Maheswara R. Samatham and Dhiraj K. Pradhan. 1989. The de bruijn multiprocessor network: A versatile parallel processing and sorting network for VLSI. *IEEE Trans. Comput.* 38, 4 (1989), 567–581.
- Timo Schonwald, Jochen Zimmermann, Oliver Bringmann, and Wolfgang Rosenstiel. 2007. Fully adaptive fault-tolerant routing algorithm for network-on-chip architectures. In *DSD'07*. IEEE, 527–534.
- Alessandro Strano, C. Gómez, Daniele Ludovici, Michele Favalli, María Engracia Gómez, and Davide Bertozzi. 2011. Exploiting network-on-chip structural redundancy for a cooperative and scalable built-in self-test architecture. In *DATE'11*. IEEE, 1–6.
- Alessandro Strano, Davide Bertozzi, Francisco Trivino, José L. Sánchez, Francisco J. Alfaro, and José Flich. 2012. Osr-lite: Fast and deadlock-free NoC reconfiguration framework. In *SAMOS'12*. IEEE, 86–95.
- Pao-Hwa Sui and Sheng-De Wang. 1997. An improved algorithm for fault-tolerant wormhole routing in meshes. *IEEE Trans. Comput.* 46, 9 (1997), 1040–1042.
- Xianfang Tan, Lei Zhang, Shankar Neelkrishnan, Mei Yang, Yingtao Jiang, and Yulu Yang. 2008. Scalable and fault-tolerant network-on-chip design using the quartered recursive diagonal torus topology. In *GLSVLSI'08*. ACM, 309–314.
- Mojtaba Valinataj, Siamak Mohammadi, Juha Plosila, and Pasi Liljeberg. 2010. A fault-tolerant and congestion-aware routing algorithm for networks-on-chip. In *DDECS'10*. IEEE, 139–144.
- Arseniy Vitkovskiy, Vassos Soteriou, and Chrysostomos Nicopoulos. 2010. A fine-grained link-level fault-tolerant mechanism for networks-on-chip. In *ICCD'10*. IEEE, 447–454.
- Arseniy Vitkovskiy, Vassos Soteriou, and Chrysostomos Nicopoulos. 2013. Dynamic fault-tolerant routing algorithm for networks-on-chip based on localised detouring paths. *IET Comput. Dig. Tech.* 7, 2 (2013).
- Eduardo Wachter, Augusto Erichsen, Alexandre Amory, and Fernando Moraes. 2013. Topology-agnostic fault-tolerant NoC routing method. In *DATE'13*. EDA Consortium, 1595–1600.
- Dajin Wang. 2003. A rectilinear-monotone polygonal fault block model for fault-tolerant minimal routing in mesh. *IEEE Trans. Comput.* 52, 3 (2003), 310–320.
- Jie Wu. 2002. A deterministic fault-tolerant and deadlock-free routing protocol in 2-D meshes based on odd-even turn model. In *Proceedings of the 16th International Conference on Supercomputing*. ACM, 67–76.
- Jie Wu. 2003. A fault-tolerant and deadlock-free routing protocol in 2D meshes based on odd-even turn model. *IEEE Trans. Comput.* 52, 9 (2003), 1154–1169.
- Jie Wu and Dajin Wang. 2005. Fault-tolerant and deadlock-free routing in 2-D meshes using rectilinear-monotone polygonal fault blocks. *Int. J. Parall., Emerg Distrib Syst* 20, 2 (2005), 99–111.

- Duan Xinming and Sun Xuemei. 2010. Fault-tolerant routing in a PRDT (2, 1)-based NoC. In *ICCET'10*, Vol. 2. IEEE, V2-506–V2-510.
- Guoqiang Yang, Mei Yang, Yulu Yang, and Yingtao Jiang. 2007. On the physical layout of PRDT-Based NoCs. *ITNG'07* (2007), 729–733.
- Mei Yang, Tao Li, Yingtao Jiang, and Yulu Yang. 2005. Fault-tolerant routing schemes in RDT (2, 2, 1)/ α -based interconnection network for networks-on-chip design. In *ISPAN'05*. IEEE, 6 pp.
- Yulu Yang, Akira Funahashi, Akiya Jouraku, Hiroaki Nishi, Hideharu Amano, and Toshinori Sueyoshi. 2001. Recursive diagonal torus: An interconnection network for massively parallel computers. *IEEE Trans. Parallel Distrib. Syst.* 12, 7 (2001), 701–715.
- Qiaoyan Yu, José Cano, José Flich, and Paul Ampadu. 2012. Transient and permanent error control for high-end multiprocessor systems-on-chip. In *NOCS'12*. IEEE, 169–176.
- Qiaoyan Yu and Paul Ampadu. 2011. A dual-layer method for transient and permanent error co-management in NoC links. *TCAS II: Express Briefs* 58, 1 (2011), 36–40.
- Yang Yu, Mei Yang, Yulu Yang, and Yingtao Jiang. 2005. A rdt-based interconnection network for scalable network-on-chip designs. In *ITCC'05* IEEE, Vol. 2, 723–728.
- Yang Yulu and Hideharu Amano. 1996. Message transfer algorithms on the recursive diagonal torus. *IEICE TRANSACTIONS on Information and Systems* 79, 2 (1996), 107–116.
- Lei Zhang, Yinhe Han, Qiang Xu, and Xiaowei Li. 2008a. Defect tolerance in homogeneous manycore processors using core-level redundancy with unified topology. In *DATE'08*. ACM, 891–896.
- Meilin Zhang, Qiaoyan Yu, and Paul Ampadu. 2012. Fine-grained splitting methods to address permanent errors in network-on-chip links. In *ISCAS'12*. IEEE, 2717–2720.
- Zhen Zhang, Alain Greiner, and Sami Taktak. 2008b. A reconfigurable routing algorithm for a fault-tolerant 2D-mesh network-on-chip. In *DAC'08*. IEEE, 441–446.
- Haibo Zhu, Partha Pratim Pande, and Cristian Grecu. 2007. Performance evaluation of adaptive routing algorithms for achieving fault tolerance in NoC fabrics. In *ASAP'07*. IEEE, 42–47.

Received March 2015; revised October 2015; accepted December 2015