Study of Fault Tolerance for King Topologies

Emilio Castillo, Esteban Stafford, Fernando Vallejo, Jose Luis Bosque, Carmen Martinez, Cristobal Camarero, Ramon Beivide. [1]

Resumen—

This paper analyzes the robustness of the king networks for fault tolerance. To this aim, a performance evaluation of two well known fault tolerant routing algorithms in king as well as 2d networks is done. Immunet that uses two virtual channels and Immucube, that has a better performance while requiring three virtual channels. Experimental results confirm the excellent behavior, both in performance and scalability, of the king topologies in the presence of failures.

Finally, taking advantage of the topological features of king networks, a new fault tolerance routing algorithm for these networks is presented. From a cost/performance point of view this algorithm is a compromise between the two previous algorithms.

I. Introduction

One of the key aspects that will determine the performance of future super-computers will be the interconnection network. Not only at a system level, but also within the processor itself. It is well known that processors, such as those used in super-computers, can contain many cores and their number will continue to grow in the near future. For example, Intel Labs has presented the Single-chip Cloud Computer(SCC), a research microprocessor containing 48 cores. It incorporated technologies intended to scale multi-core processors to a hundred cores and beyond[6]. With large amounts of cores, the importance of fast communications is key. Thus the performance of the processors is heavily conditioned by that of the interconnection network.

Our proposal is based on king topologies [14], [8] whose propierties were presented at this very same conference in [15]. These networks are a higher degree evolutions of the classic 2D mesh and torus. The added links allow packets to advance in eight directions like the king on a chessboard. Meaning that they double the degree (or radix) of their 2D counterparts while still presenting a straightforward layout. Furthermore, they exhibit excellent topological properties such as high throughput, low latency, easy partitioning and good scalability. The proposal of topologies using diagonal links has been considered in the past, in the fields of microprocessor design[5], FPGAs[7] and interconnection networks[16]. Also mesh and toroidal topologies with added diagonals have been considered, both with degree six[13] and eight[4].

This paper presents king topologies in the context of fault tolerance. The robustness of these networks is based on their high degree and path diversity. This is asserted by an experimental study that compares them with common 2D networks, like meshes or tori. For this, two well known fault tolerance routing algorithms have been used: Immunet[9] and Immucube[10], both adapted to king topologies. Experimental results confirm the excellent behavior of king topologies, both in performance and scalability.

In addition to using general algorithms, this paper also proposes a new algorithm for fault-tolerant routing in king topologies (KFT). It relies on the topological particularities of king networks. Experimental results show that it offers better cost-performance ratio than the algorithms cited above, as it affects only local areas around faults. This leaves the rest of the network to operate normally, allowing for higher scalability.

A great amount of research has been carried out around the topic of fault tolerance on interconnection networks. The first ideas in fault tolerant interconnection networks appeared in high performance computers at a system level[2][1]. In these, when a packet encounters a fault on the way to its destination, the routing algorithm tries to find an alternative path. Rodrigo et al. [12] proposed uLBDR (Universal Logic-Based Distributed Routing) as an efficient routing mechanism to support any irregular topology derived from 2D meshes. The complexity of the routers grows with the number of faults to handle and a trade-off must be reached.

The remainder of the paper is organized as follows. Some general considerations about fault tolerance and the description of Immunet and Immucube are shown in section II. In Section III the fault-tolerance algorithm for king networks is presented. Following, in section IV, is an experimental evaluation of the networks and algorithms mentioned in the paper. Finally, the paper concludes with section V summarizing the main findings of the paper.

II. Fault Tolerance in Interconnection Networks

This section starts by presenting a set of general concepts around the topic of fault tolerance in interconnection networks and the capacity of the king topologies to operate under these conditions. In addition it describes two well known fault tolerant routing algorithms that have been selected to evaluate these topologies when faults are considered. These are Immunet[9] and an optimization for mesh and toroidal topologies named Immucube[10].

The growing complexity of computer systems and the increase of the number of components they comprise, causes the rise of the Mean Time Between Failures (MTBF). In the context of high-performance computing, in some occasions the MTBF can be shorter than the mean execution time of some ap-

[1]Dpto. electrónica y computadores, Univ. Cantabria, e-mail: castilloe@unican.es

plications. This makes it necessary to develop techniques that allow interconnection networks to perform acceptably even in the presence of failures.

Fault tolerance is the ability of systems to provide the services that they were designed for regardless of failures they may suffer. Then, a fault tolerant system is one that can hide the occurrence of faults, usually by making use of redundancy within the system. A system is said to be N-fault tolerant if it can gracefully handle any combination of N faults. All the algorithms studied in this paper support any number of faults, as long as the network remains connected.

A fundamental aspect of fault tolerance is the redundancy. In general this is the property of having more resources than strictly necessary for fault-free operation. Interconnection networks in general have some inherent degree of redundancy as there is usually more than one way to reach a destination. The path diversity found in king networks gives them high redundancy. Then the probability of not finding a way to the destination, or that the network has been rendered disconnected, is very low. In addition the loss of a number of paths connecting two nodes has less impact in the performance.

In this paper the faults are considered to be permanent and non-reparable. These can occur in the links, but also within the nodes. Some faults in the crossbar or the message queues will affect the communication between the node and one of its neighbors, also causing a faulty link. A faulty node is one that has at least one faulty link. The communication between any pair of nodes will be possible as long as the network remains connected. Then all nodes must have at least one healthy link.

The mechanisms to avoid deadlock and livelock used for fault-free networks are usually ineffective in the presence of failures. Even in the case of a single fault they will not prevent the network from blocking. Therefore fault tolerance solutions must ensure that the network remains deadlock-free.

The algorithms presented in this paper fall into the category of fault tolerant routing algorithms. Using the intrinsic redundancy of the networks there is no need for additional resources and they can cope with a high number of faults with little performance degradation. In the event of failure, the network is reconfigured changing the routing tables to avoid faulty links with a special protocol.

The two algorithms used on king topologies are Immunet and Immucube [9], [10]. Both of them rely on the generation of a spaning tree contained in the network. This spaning tree is created any time a fault arise and replaces the deadlock-free routing algorithm of the network. Immunet presents some scalability problems. When congestion is very high, the throughput delivered approaches that of a ring instead of the higher degree network containing it. This problem was solved with the introduction of Immucube [10]. Immucube adds a third virtual channel. Instead of discarding the healthy-network deadlock avoidance mechanism that governs the escape channels, Immucube adds a third virtual escape channel for the ring. When the network is fault-free, the extra channel is used as a second adaptive virtual channel.

III. King Fault Tolerance Algorithm

To further test the fault-tolerance properties of king networks, this paper presents a new fault-tolerant routing algorithm that exploits their particularities. It has been named King-Fault-Tolerance (KFT). The algorithm was devised trying to achieve the performance of Immucube at the cost of Immunet. Thus, it uses only two virtual channels and its performance is close to Immucube for a low number of faults. The key feature is that it only affects the area close to faults. Therefore, it does not disturb the traffic in healthy areas. Therefore it is more scalable than the other two algorithms.

As in Immunet, KFT uses the concept of a tree traversed in preorder to create a ring. However, instead of one ring covering the whole network, KFT creates a small ring that encircle a faulty links. KFT takes advantage of the high connectivity of king topologies to allow small trees to cover fault areas, reducing the number of packets to generate them. The concept could be used in 2d networks, but the resulting algorithm would be more complex, falling beyond the scope of this paper.

The algorithm requires nodes to keep some local data.

- The *reconfiguration status*(RS) register contains the concatenation of two integers. On the high order part is the identifier of the node that generated the fault-alert propagation. And on the low order part the number of reconfiguration processes it started.
- During the reconfiguration process, the *passages table* is used describe the structure of the tree. Later, it will guide packets around the tree.
- A register with the current amount of active reconfiguration processes in the node.
- The *state* register tells whether the node is in fault mode, depending on the number of faulty links of the node.
- A list of pending communications with neighbors.

Also, the nodes must exchange some information. This is carried by a set of special KFT packets that only travel among neighbors, so they do not need routing.

- The *fault-alert* packet requests the recipient to become a member of the tree as a descendant of the sender.
- The *acknowledge(ACK)* and *decline(NACK)* packet conveys a positive or negative answer to the fault-alert packet.

The reconfiguration process can be divided into two algorithms. The first occurs in the nodes that

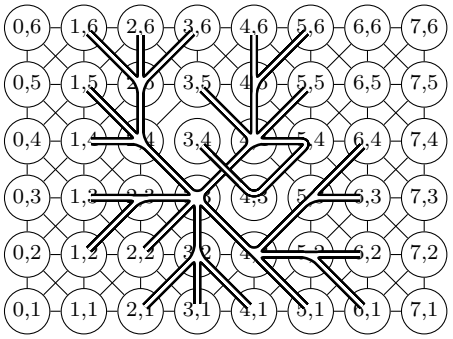


Fig. 1. Ring or tree generated by the algorithm.

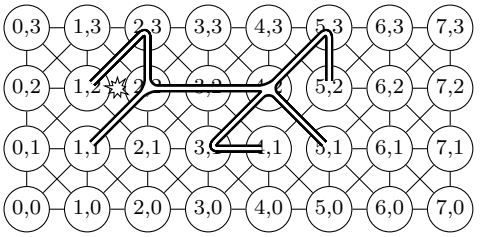


Fig. 2. Two faulty regions joined by one ring after a fault occurring between node (1,2) and (2,2).

detect a failure and triggers the reconfiguration process. The second happens when a KFT packet is received, it can be a fault-alert or a reply, and is how the formation of the ring takes place.

When a node senses that a link fails, it updates its RS register, increasing the amount of reconfiguration processes and, after stopping all normal traffic, it sends fault-alert packets to all its neighbors. The nodes in the vicinity, on the reception of the fault-alert, start the second algorithm.

Depending on the kind of KFT packet received, node behavior will differ. During the process of reconfiguration, a node can receive fault-alert packets from different neighbors. These packets request the recipient to join a tree, however, a node can only belong to one tree at a time. Thus, it must choose which request to acknowledge and which to decline. When a node receives a fault-alert packet, it will reply affirmatively only if the RS of the packet is greater than the one stored in the node. Otherwise, the node sends a decline packet back to the sender. This mechanism causes one tree to prevail and the others to be forgotten. The acknowledge must fulfill some other conditions that prevent the algorithm from covering the whole network.

If a node receives an acknowledge(ACK) packet, the sender is added as a child to the tree by updating the passage table, declinations(NACK) are simply ignored. In both cases (ACK and NACK), the sender is removed from the pending communication list. When the list is empty, the reconfiguration process is deemed complete and normal data communications can be resumed.

It is important to note that the way in which the RS values are used causes a single tree to eventually cover the faulty region. Once this happens, the information stored in the passage table guides the packets through the tree, in a ring-like fashion, avoiding the faulty links. As an example see figure 1.

.0.a Multiple simultaneous failures. The scenario presented above with a single faulty link is useful to describe the KFT ring generation algorithm. However, KFT can cope with more complex scenarios. It can withstand multiple failures happening simultaneously with the reconfiguration process of previous faults. If the new failure occurs in a healthy area, KFT creates a new ring isolated from the existing ones. However, if a new fault occurs close to an existing ring, then it is extended to cover the new fault. If as a consequence of a ring increasing its size it meets another, both are joined to form a single ring covering both faulty regions. An example of this process can be seen in Figure 2, where the last link to fail is that shown with ✸ . The process created a ring that joined two existing and isolated ones. Notice that this tree has some of its links pruned as it will be discussed later.

The scenario in which two rings are joined as above is not contemplated by the previous algorithm. And under some conditions, the two regions will not join. In the case that one of the rings is already stable and has a higher RS.high than the reconfiguring ring. The stable one will decline all the fault-alert packets from the reconfiguring one. This will lead to the two regions not joining. However this situation can be detected and trigger a reconfiguration process with higher RS.high, that will engulf both rings leaving only one.

The algorithm presented above must be modified to cope with these situations by adding two conditions. First, the fault-free nodes should broadcast fault-alert packets if they come from a faulty node and their RS.high is lower. The receiving node should not change its status. This causes that a fault-alert packet will be forwarded to a node from a different ring whose RS.High is higher. According to the algorithm above, this packet should be ignored, therefore a second condition is added. This causes that when a faulty-node receives a fault-alert from a fault-free node, it will respond with NACK and, after increasing its RS.high, start a new reconfiguration process. As it has a higher RS.high, it has a higher priority and will eventually merge the two existing rings.

.0.b Optimization: pruning the tree. Once a reconfiguration process concludes, the normal traffic on the network starts moving again. When a packet needs to cross a faulty link it enters the ring and only leaves when it gets closer to its destination. Obviously, the smaller the ring, the less detour the packet must make. With this in mind, KFT has a pruning phase that reduces the size of the tree and thus the size of the ring. As a consequence, the latency of packets affected by a fault is reduced.

As an example consider the ring in figure 1. In the figure it becomes apparent that the ring is covering more nodes than necessary to wrap the faulty

region. The optimization consists in pruning leaf nodes which are not needed. Once a node is in a stable state and is fault-free, it can easily determine if it is a leaf node. If there are no children in the passage table it sends a prune message to its parent and waits for the acknowledge. The parent node clears its child from the passage table and sends the acknowledge back. On reception the leaf node will no longer be part of the ring. Figure 2 shows a tree after the pruning stage.

Pruning must be done as a separate stage as some pathological cases require some leaf node to belong to the tree in order to successfully conclude the reconfiguration stage.

.0.c Routing. When a failure occurs, all the nodes involved in the reconfiguration will stop normal traffic flow until their passage tables are stable. Once this happens, these packets can resume their course. After the reconfiguration the virtual channels will be reorganized. The escape channels will continue to do static DOR routing. But the fully adaptive channels of the links that conform the ring will be used exclusively by packets in fault mode. The rest of the adaptive channels will remain adaptive. Therefore, the functionality of the fault-free regions is not changed and ABR routing is used as if no failure had occurred.

In faulty regions, when a packet needs to advance through a link that belongs to the ring, it must request the escape channel. While in the ring, packets must use the escape channel unless they need to advance through a faulty link. Then they enter fault mode. To the packets in fault mode a new header will be added. This will indicate their state and also the distance to their destination when they entered fault mode. Then, these packets advance through the nodes of the ring until they reach one that is closer to their destination and can exit the ring resuming their normal state.

In order to avoid deadlocks, packets entering or leaving the ring must abide by some rules. Like when a packet changes dimension in ABR, packets entering the ring must satisfy the bubble condition. When leaving, packets must first try to exit through an adaptive channel. If not possible, they should request an escape channel. However, these channels only route packets under DOR. And because the packets exiting the ring might violate this rule, it is necessary to re-inject them. This technique has been used by several authors like [3] but increasing the number of virtual channels instead of reinject the packets.

IV. Evaluation and Results

This section presents a set of experiments that allow to evaluate the fault tolerance properties of king networks compared with their 2D counterparts, conventional meshes and tori. It also includes a comparison of the fault-tolerant routing algorithms described. This is done evaluating their performance as well as their resource requirements.

All experiments have been carried out with Fsin, a functional simulator for interconnection networks belonging to the INSEE Environment [11]. They have been carried out on networks of four topologies (mesh, torus, king mesh and king torus) with 16×16 and 32×32 nodes. In order to ensure that king networks of 16×16 reach saturation, two injectors per router have been used.

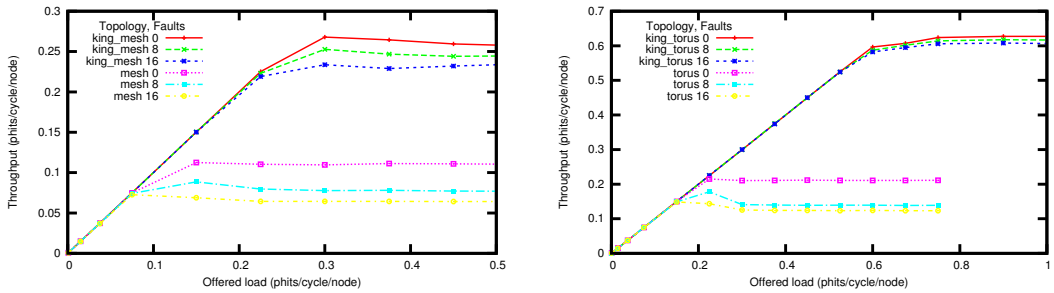
The traffic used has an uniform distribution, injecting packets of 16 phits at a constant rate. The fault generation model is random with an uniform distribution. Performance is evaluated by considering values of throughput and average latency. To achieve stability in the results, the values presented on the graphs are the average of five individual experiments with different random seed.

.0.a King topologies vs 2D. The results of the first experiment show a comparison of the performance of networks of 32×32 of different topologies. First, in figure 3(a), the results for meshes and king meshes are compared and then, those for tori and king tori are shown in figure 3(b). Both figures show the performance of healthy networks and two more examples with 8 and 16 faults. The algorithm used in all cases is Immucube, as Immunet is known to give poor performance in large networks and KFT is only applicable to king topologies.

Both figures show that the loss of performance due to faults is much less in king topologies. Meshes show a degradation of 36% for 8 faults and 45% for 16, while king meshes only decrease in 5% and 9% respectively. With tori, the difference is more pronounced as king tori only loose 1.6% and 3.2% and conventional tori 33% and 42% respectively.

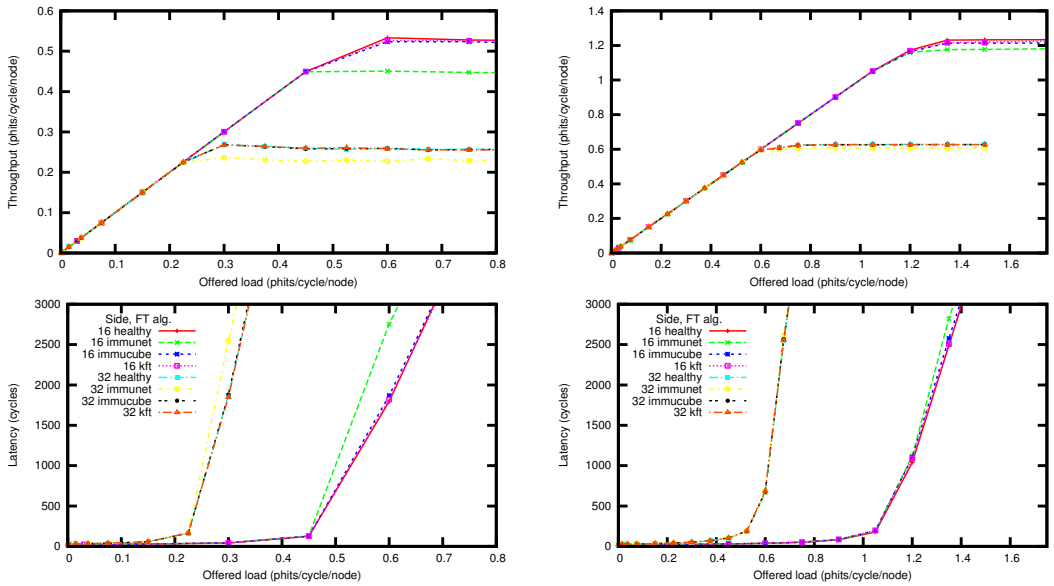
.0.b Experiments with a single fault. Focusing on the performance of king networks, the next experiment presents a comparison of the three fault-tolerant routing algorithms working on networks of 16×16 and 32×32. To visualize the degradation, the performance of the healthy network is also shown. Figure 4(a) shows throughput and latency for meshes and figure 4(b) for tori. At low loads, before saturation point, the three algorithms give the same throughput and latency as the healthy network. At high load, the performance loss in Immucube or KFT is negligible and quite apparent in Immunet. This effect is more pronounced in tori.

The degradation observed in Immunet is related to the fact that, when a fault is detected, the escape channels stop using DOR and use the ring that covers all the nodes, leaving the escape network notably reduced. And as the escape channels are mostly used in saturation, the performance deteriorates. In contrast, both Immucube and KFT have better results because they use DOR in the escape channels. However, the performance of KFT is slightly better as it only affects the vicinity of the failure, while Immucube does not allow packets that have encountered a fault in their way to use DOR. The latency graphs confirm this behavior; at high loads Immunet is slightly higher than the rest.



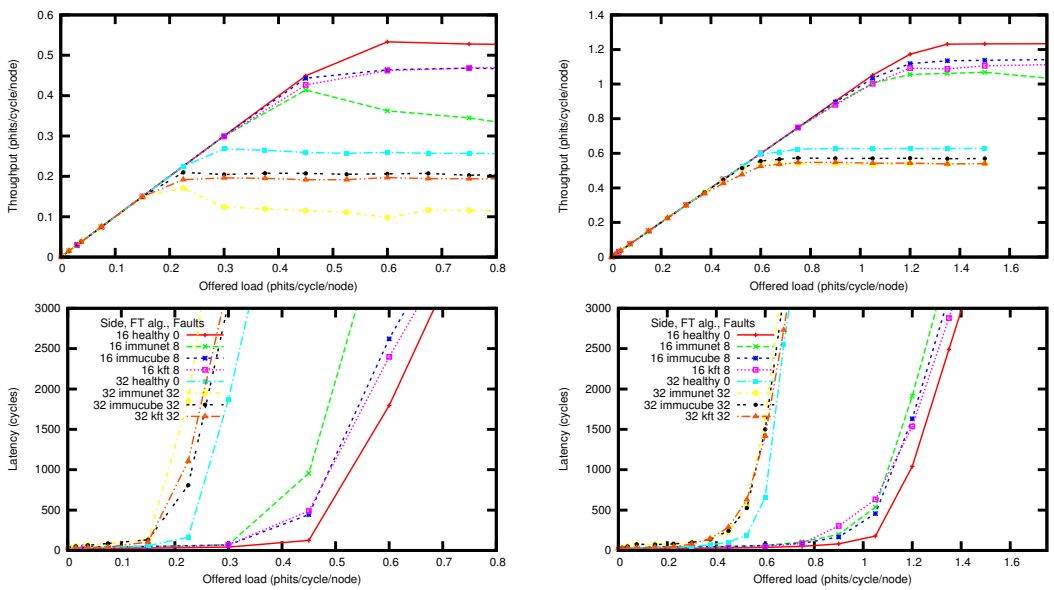
(a) Meshes vs. king meshes. (b) Tori vs. king tori.

Fig. 3. Throughput of 32×32 networks with Immucube.



(a) Mesh (b) Torus

Fig. 4. Throughput and latency of king topologies with one random fault.



(a) Mesh (b) Torus

Fig. 5. Throughput and latency of king topologies with several random faults.

.0.c Experiments with multiple faults. Figures 5(a) and 5(b) show the latency and throughput of networks of sizes 16×16 with 8 random faults and of 32×32 with 32 faults. These illustrate how the algo-

rithms behave when the number of faults increases. The tendencies observed are independent of the size of the networks, thus the evaluation will be done together.

On meshes (Fig. 5(a)), the higher number of faults is more noticeable as the performance loss is higher than before. Yet Immunet still shows the highest degradation and both KFT and Immucube have similar results. However, the third virtual channel used in Immucube allows for slight improvement over KFT, because it sacrifices adaptive channels to the ring. On tori (Fig. 5(b)), the tendency is the same but with a higher difference between KFT and Immucube, again due to the third virtual channel. On Immunet, the larger amount of faults increase the probability of packets requesting escape through the ring. This causes a faster congestion in it, therefore hampering the traffic throughout the network.

At the beginning of the saturation area where latencies are still small, KFT has higher latencies due to the loss of adaptive channels. However as the networks saturate, packets in Immunet and Immucube are forced to make longer detours through their large rings, while KFT is able to keep latencies slightly lower.

V. Conclusions

Current trends in super-computer design indicate that the number of cores per processor will continue to grow. Thus, their performance will be heavily affected by that of the interconnection networks of their processors. The large amount of component in these networks increase the fault probability, affecting the exploitation of super-computers. This rises the importance of using fault-tolerant networks and routing algorithms. In addition, these allow to improve the productivity of the fabrication process, as partially defective processors can still be used with a diminished performance.

King topologies have been previously presented as viable network fabric for future multi-core processors. This paper shows their performance in the context of fault tolerant architectures. The high degree and path diversity of these direct networks tolerate multiple faults with minimum performance loss. In addition, these networks exhibit high scalability as shown by the experimental results. Indeed, larger networks show less degradation even with proportionally more faults.

For the evaluation of these networks, three fault tolerant algorithms have been used. The first two, Immunet and Immucube, were previously published and have been adapted to king topologies. The last one, named KFT, was specifically developed to provide fault tolerance in these networks. Experimental results showed that KFT offers similar performance to that of Immucube in some situations but using less resources.

Acknowledgments

This work has been supported by the Spanish Ministry of Science under contracts TIN2010-21291-C02-02, AP2010-4900 and CONSOLIDER Project CSD2007-00050, and by the European HiPEAC Network of Excellence.

Referencias

[1] Chun-Lung Chen and Ge-Ming Chiu. A fault-tolerant routing scheme for meshes with nonconvex faults. *IEEE Trans. Parallel Distrib. Syst.*, 12:467–475, May 2001.

[2] W. J. Dally and H. Aoki. Deadlock-free adaptive routing in multicomputer networks using virtual channels. *IEEE Trans. Parallel Distrib. Syst.*, 4:466–475, April 1993.

[3] M. E. Gomez, N. A. Nordbotten, J. Flich, P. Lopez, A. Robles, J. Duato, T. Skeie, and O. Lysne. A routing methodology for achieving fault tolerance in direct networks. *IEEE Trans. Comput.*, 55:400–415, 2006.

[4] WH Hu, SE Lee, and N. Bagherzadeh. Dmesh: a diagonally-linked mesh network-on-chip architecture. *nocarc*, 2008.

[5] M. Igarashi, T. Mitsuhashi, A. Le, S. Kazi, Y.T. Lin, A. Fujimura, and S. Teig. A diagonal interconnect architecture and its application to risc core design. *IEIC Technical Report (Institute of Electronics, Information and Communication Engineers)*, 102(72):19–23, 2002.

[6] Intel Labs. Single-chip cloud computer. In *Enterprise Research*, page http://techresearch.intel.com/ProjectHome.aspx, 2011.

[7] A. Marshall, T. Stansfield, I. Kostarnov, J. Vuillemin, and B. Hutchings. A reconfigurable arithmetic array for multimedia applications. In *FPGA 99: Proceedings of the 1999 ACM/SIGDA seventh international symposium on Field programmable gate arrays*, pages 135–143, New York, NY, USA, 1999. Acm.

[8] C. Martinez, E. Stafford, R. Beivide, C. Camarero, F. Vallejo, and E. Gabidulin. Graph-based metrics over qam constellations. *Information Theory, ISIT. IEEE International Symposium*, pages 2494–2498, 2008.

[9] V. Puente, J. A. Gregorio, F. Vallejo, and R. Beivide. Immunet: A cheap and robust fault-tolerant packet routing mechanism. *SIGARCH Comput. Archit. News*, 32:198–, March 2004.

[10] Valentin Puente and Jose Angel Gregorio. Immucube: Scalable fault-tolerant routing for k-ary n-cube networks. *IEEE Trans. Parallel Distrib. Syst.*, 18:776–788, June 2007.

[11] F. J. Ridruejo and J. M. Alonso. Insee: An interconnection network simulation and evaluation environment. In *Euro-Par 2005, Parallel Processing, 11th International Euro-Par Conference, Lisbon, Portugal*, pages 1014–1023. Springer, 2005.

[12] S. Rodrigo, J. Flich, A. Roca, S. Medardoni, D. Bertozzi, J. Camacho, F. Silla, and J. Duato. Addressing manufacturing challenges with cost-efficient fault tolerant routing. In *Networks-on-Chip (NOCS), 2010 Fourth ACM/IEEE International Symposium on*, pages 25 –32.

[13] K.G. Shin. Harts: a distributed real-time architecture. *Computer*, 24(5):25 –35, may 1991.

[14] E. Stafford, J. L. Bosque, C. Martínez, F. Vallejo, R. Beivide, and C. Camarero. A first approach to king topologies for on-chip networks. In *Proceedings of the 16th international Euro-Par conference on Parallel processing: Part II*, Euro-Par'10, pages 428–439, Berlin, Heidelberg, 2010. Springer-Verlag.

[15] E. Stafford, J.L. Bosque, C. Martinez, F. Vallejo, R. Beivide, and C. Camarero. A first approach to king topologies for on-chip networks. In *XXII Jornadas de Paralelismo, CEDI 2011*, 2011.

[16] K.W. Tang and S.A. Padubidri. Diagonal and toroidal mesh networks. *Computers, IEEE Transactions on*, 43(7):815–826, 1994.