A fault-tolerant deadlock-free adaptive routing for On Chip interconnects

Fabien Chaix
TIMA
Grenoble, France
Fabien.Chaix@imag.fr

Dimiter Avresky
IRIANC
Boston, USA
autonomic@irianc.com

Nacer-Eddine Zergainoh
TIMA
Grenoble, France
Nacer-Eddine.Zergainoh@imag.fr

Michael Nicolaidis
TIMA
Grenoble, France
Michael.Nicolaidis@imag.fr

***Abstract*—Future applications will require processors with many cores communicating through a regular interconnection network. Meanwhile, the Deep submicron technology foreshadows highly defective chips era. In this context, not only fault-tolerant designs become compulsory, but their performance under failures gains importance.**

In this paper, we present a deadlock-free fault-tolerant adaptive routing algorithm featuring Explicit Path Routing in order to limit the latency degradation under failures. This is particularly interesting for streaming applications, which transfer huge amount of data between the same source-destination pairs.

The proposed routing algorithm is able to route messages in the presence of any set of multiple nodes and links failures, as long as a path exists, and does not use any routing table. It is scalable and can be applied to multicore chips with a 2D mesh core interconnect of any size. The algorithm is deadlock-free and avoids infinite looping in fault-free and faulty 2D meshes.

We simulated the proposed algorithm using the worst case scenario, with different failure rates. Experimentation results confirmed that the algorithm tolerates multiple failures even in the most extreme failure patterns. Additionally, we monitored the interconnect traffic and average latency for faulty cases. For 20x20 meshes, the proposed algorithm reduces the average latency by up to 50%.

I. Introduction

As new applications require higher performance, the use of multicore chips with hundreds and thousands cores and parallel programming model are the current trend in the computer industry. This new paradigm supports well the performance scalability, but leads to new design challenges. Core interconnects become a "bottleneck" regarding scalability, power consumption, chip performance and reliability.

In the same time, the Deep submicron technology enables a lower silicon cost at the expense of significant reliability concerns. In this context, failures in the Core Interconnect are an important issue for the multicore chips. They may cause anomalies (deadlocks, core isolation, message losses) or degrade significantly the system performance (e.g. latency increase).

Some related work is given in Section II. In III, the router architecture is detailed. In IV, the novel routing algorithm is presented. In V, some experimental results are given. Section VI concludes the paper.

978-3-9810801-7-9/DATE11/©2011 EDAA

II. Related work

In the past years, the Network on Chip (NoC) has become a very popular solution for interconnecting cores. Most common implementations are based on the 2D Mesh topology, which allows a simple planar layout. However, the NoC paradigm is still in its infancy [1], because of exacerbated constraints such as the power consumption or silicon cost.

In [2], a general purpose router architecture is proposed based on Virtual Channels and Virtual Networks. The authors present an adaptive routing that improves significantly the average throughput over the XY algorithm.

On the other hand, as the technology scales down, failures become more frequent. At a higher level, a fault-tolerant routing algorithm is required for taking advantage of the regularity and high redundancy of the interconnection network. In effect, such routing algorithms allow messages to bypass faulty links and nodes and effectively improve the system fault tolerance [3]. In [4], the routes are discovered using broadcast, which is generating a significant traffic. Afterwards, the source node stores the route for following messages.

In addition to routers and links failures, designers face deadlock issues [5]. For regular networks, the prohibition of certain turns [6] during message routing guarantees the deadlock freedom. However, ensuring both deadlock freedom and fault tolerance often requires Virtual Channels.

In [7], the authors presented a fault-tolerant deadlock-free routing algorithm that guaranteed the message delivery from any source to any destination node, as long as a path existed, for 2D mesh interconnects of any size.

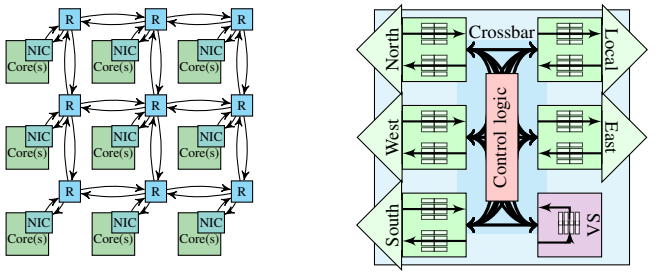
Our paper presents a novel routing algorithm using the algorithm proposed in [7]. This contribution leverages Explicit Path to reduce the latency, and simplifies the router design.

III. Router architecture

A. Context

We consider a Multicore chip using wormhole routing, based on a 2D Mesh NoC, as shown on Fig.1-a. On each node, a Network Interface Controller (NIC) converts processor requests in flits and manages end-to-end flow control, alike Acknowledgement (Ack) and message timeout.

The failure detection is not addressed in this paper, but many solutions from the literature can be used. In the rest of this



(a) 2D Mesh On-Chip interconnect (b) Router with Virtual Source

Figure 1. Generic 2D Mesh Interconnect and router architecture

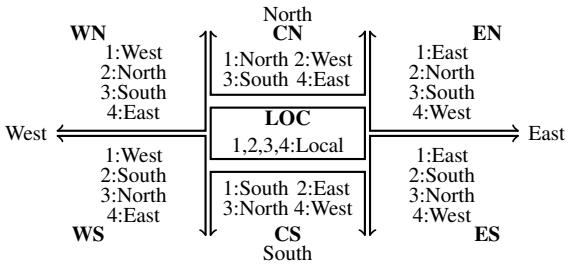


Figure 2. Output hierarchy depending on the destination node position relative to the current node. When destination is on the same column, South (CS), North (CN) or Local (LOC) output port is promoted

article, we assume that link and node failures are detected and neighbour nodes are notified instantly.

B. Virtual Channels and Virtual Networks

The router architecture shown in Fig.1-b is generic and is based on [7]. Each router's physical link is shared between several Virtual Channels (VCs).

For our routing algorithm, Virtual Networks (VNs) are defined as non overlapping groups of VCs, where messages are propagated. In this paper, we use 4 Virtual Channels per port, and 2 are assigned to each VN. The VN to use is chosen when the message is injected in the interconnect. If the destination is northern (resp. southern) of the source, South Last (resp. North Last) VN will be chosen.

C. Virtual Source and Node Stamping

However, when there are faulty nodes and links, routing messages between 2 nodes may take many turns. Some messages may even have no option but to break its Virtual Network turn restriction. In such case, the message uses a Virtual Source to continue towards its destination, as proposed in [8].

Definition 3.1 (Virtual Source): The Virtual Source is used to swap safely messages from a VN to another, or simply to break dependencies within the same VN. The complete message is stored in the VS buffer shown in Fig.1-b, thus eliminating the dependencies on previously visited buffers. Then, it is re-emitted starting from this node. ■

Unfortunately, supporting fully-adaptive routing requires an additional complexity for avoiding infinite looping. Therefore, the Node Stamping has been introduced for the messages to keep track of the traversed nodes, and avoid them in following routing decisions.

Definition 3.2 (Node Stamping): Each node used by a message m during routing is stored with it (i.e. stamped). When computing a new hop, the router avoids the listed nodes. We denote $V_m(x)$ the list of the output ports of node x, which are heading to nodes already visited by the message m. ■

D. Output hierarchy

The Output Hierarchy given in Def.3.4 leverages the interconnect regularity to route messages toward their destination in the presence of multiple faulty nodes and links.

Definition 3.3 (Eligible output ports): We denote $P(x)$ the set that contains the direction of all output ports of node x that can be used for forwarding a message, i.e. it exists and is fault-free. ■

Definition 3.4 (Output Hierarchy): Given Q the set of eligible output ports of the current node x, and d the message's destination node, $Hierarchy(Q, x, d)$ returns the port of Q the highest in the hierarchy presented in Fig.2. ■

E. Echo Mode

The Output Hierarchy alone is not sufficient to guarantee 0% message loss, even for low failure rates. In effect, because the routers have only a partial knowledge of the interconnect state, messages are trapped sometimes. In such circumstance, the Echo Mode, introduced in [7], returns the message to the previous node iteratively until another routing option is found.

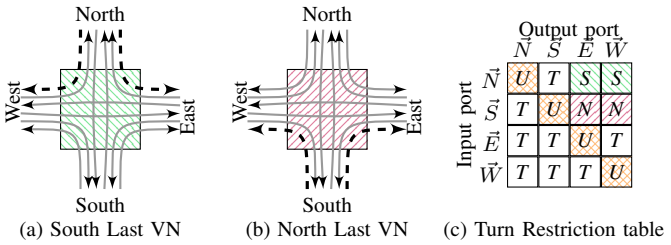
Definition 3.5 (Echo mode): If there is no usable direction to forward the message to the destination node, then the Echo mode will be applied, according to Fig.4. $Echo(m, x)$ returns the output port through which the message m entered the node x for the first time. Therefore, the Echo Mode enables the message to rewind until it finds another path or reach the source node. ■

IV. Proposed routing algorithm

A. Explicit Path Routing Mode

The major contribution to the latency reduction of the routing algorithm proposed in Fig.4 is based on the fact that the path of messages can be stored on source and destination node. This is particularly beneficial when messages have activated Echo mode, as a consequence of failures. This will allow throughput to be significantly improved and thus improve the system efficiency. This is especially important for Streaming applications that use repeatedly the same source- destination node pairs with large amounts of data.

In order to use Explicit Path Routing, the Explicit Path is extracted from the Node Stamping information after the first message has been routed from a given source. The Explicit is then stored at the destination node, according to Def.4.1. The acknowledgment then follows the Explicit Path, and reaches the source node with a copy of the Explicit Path. This allows the source node to store its own copy, for future messages.



(a) South Last VN (b) North Last VN (c) Turn Restriction table

Figure 3. Turns restrictions, in dashed, for the South Last VN (a) and North Last VN (b), and the associated table (c). Turns marked *U* are forbidden for both VNs (U Turn are prohibited). North Last VN restrictions are marked *N* and South Last VN restrictions are marked *N*. Other turns, including those which source or destination is Local, are granted (marked *T*).

Because routes are stored in nodes' memory, there is virtually no limit to the Explicit Path storage.

Definition 4.1 (Explicit Path): For a given node s, and considering a destination node d, $R_{s\to d}$ is a valid path from s to d, obtained after a message was received from node d by the node s or a message from s to d was acknowledged. ■

Definition 4.2 (Explicit Path Routing Mode): When an Explicit Path $R_{s\to d}$ exists in node s, a message m from the node s to the node d will store the Explicit Path as a list of output ports R_m, starting from the source node s. Afterwards routers will read directly the route R_m, skipping Hierarchy and Echo Mode, according to the algorithm in Fig.4.

B. Virtual Network Turn Restriction

In [7], the deadlock freedom of each VN was guaranteed by channel numbering, where flits had to use channel (i.e. links) only in ascending order. This provided a solid theoretical background for the deadlock freedom of the proposed algorithms. However, this scheme is relatively expensive to implement, because channel numbers must be compared at each step and the last numbering embedded in the flits.

In this article, we suggest to use turn restrictions as described in Def 4.3. This approach is equivalent to the channel numbering regarding the prohibited turns as shown in Fig.3-a,b, but is much more economical. In effect, it takes only a 4x4 table to check if the Virtual Network forbids a given turn, as shown on Fig.3-c.

Definition 4.3 (Virtual Network Turn Restriction): According to Virtual Network Turn Restrictions table shown in Fig.3-c, the boolean $VN_m(p_{in}, p_{out})$ is set to *true* if and only if the message m can be forwarded from input port p_{in} to output port p_{out} without breaking the restrictions of the VN used by m. ■

C. Presentation and example

The main novelty of the proposed algorithm is that it uses Explicit Path Routing to improves average latency, while guaranteeing deadlock freedom and fault tolerance.

According to Fig.4, the proposed routing algorithm consists of 3 different modes. As we are assuming that the chip failure rate will be acceptable, the Hierarchy Mode is used in most cases, and is based on the Output Hierarchy. If the message is

```
1  p_in: input port of message m
2  p_out: output port to route m to
3  x: current node for message m
4  Q: set of eligible output ports
5  start:if R_m ≠ ∅ (Explicit Path Routing mode)
6           Pop the first element of R_m to p_out
7        else
8           Set Q to P(x) \ V_m(x)
9           if Q ≠ ∅ (Hierarchy mode)
10             Set p_out to Hierarchy(Q, x, d)
11             Add x to the nodes visited by m
12          else  (Echo mode)
13             Set p_out to Echo(m, x)
14          endif
15       endif
16 send: if ¬ VN_m(p_in, p_out)
17          Use Virtual Source
18 end:  Route message to output port p_out
```

Figure 4. Proposed novel routing algorithm with Explicit path routing for a message m from the source node s to the destination d at node x

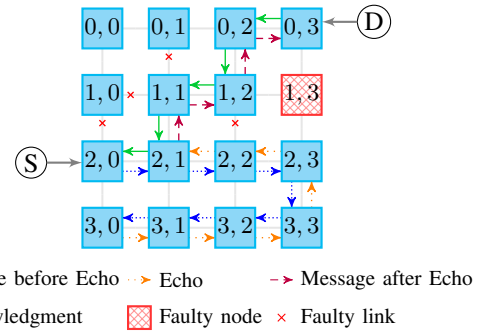


Figure 5. The proposed algorithm requires Echo Mode to route the message from source node S to destination node D, but not for the acknowledgment, thanks to the Explicit Path Routing Mode

trapped, the Echo Mode takes over and guarantees the safety properties stated in Section IV-D. At last, the router enables Explicit Path Routing Mode for messages and Acks, which improves significantly the average latency.

For example, consider the routing of the message from the node S to node D in Fig.5. Based on the hierarchy, the message will be routed to the East straight to the node $(2,3)$. Eventually, the message will be trapped in node $(3,0)$ and start Echo mode. The message returns to the node $(2,1)$, and tries another direction. First, the router starts by examining the hierarchy, as given in Fig.2. As the destination node $(0,3)$ is East North, we have {East, North, South, West}. East has already been used, so it is necessary to shift to the following direction, North. Because the node is fault-free, it can be used to reach the destination node D.

If the interconnect were partitioned, the Echo Mode would eventually return the message to its source node's Local port, allowing the source node to detect the partitioning.

Afterwards, the acknowledgment of the message from S to D is routed using Explicit Path Routing, using the Explicit Path from the acknowledged message, as defined in Def.4.1. Therefore, instead of choosing the East direction at node $(0,2)$ based on the output hierarchy, the acknowledgement directly

uses a shorter route and reaches the source node S. Any subsequent message from the node S to the node D and reverse will follow directly this Explicit Path too, improved substantially the traffic and average latency in case of failures.

D. Algorithm properties

The presented routing algorithm has several important properties, relying on the properties on the Variant C algorithm presented in [7]. First, it does not use any routing table. Second, we claim that the proposed algorithm is safe. In effect, it is deadlock-free, does not generate message cycles and always terminates. Finally, the algorithm guarantees that the destination is always reached if a path exists, otherwise network partitioning is detected.

The described properties have been validated by means of simulation of different sizes of mesh and worst case failure scenarios. Furthermore, the properties have been proved formally, based on [7], but omitted because of space restriction.

V. Experimental results

In [7], a simulation model has been built with a focus on fault tolerance, using SocLib [9] and SystemC libraries.

We simulated the proposed routing algorithm for different sizes of meshes (10x10 and 20x20). Our contribution was compared against the Variant C routing algorithm presented in [7]. For each configuration, we simulated the transfer of 100 messages in 50 interconnects with random non partitioned failure patterns induced during the start-up. We considered *Node* failures sets, *Link* failure sets and *Mixed* sets where nodes and link failures occur at half the nominal rate each.

As a statement, we observed no deadlock, no message lost and no infinite looping, even for 20x20 meshes under 40% failure rate. In Fig.6, we show the performances of different routing algorithms for mesh sizes of 10x10 and 20x20, under different failure sets.

As expected, there is little difference between the Variant C and the proposed algorithm for low failure rates, regarding the average route length (6-a) and traffic per message (6-b). However, one can see that the proposed algorithm clearly improves the average route length and traffic for failure rates starting from 20%. Finally, Fig.6-c shows that the proposed algorithm provides lower latency than Variant C, especially in the 20x20 mesh.

VI. Conclusion

The proposed Explicit Path Routing algorithm does not utilize *routing tables*, which will have a big impact on scalability and power consumption of extremely complex multicore chips consisting of more that 1000 cores.

Deadlock freedom of the solution is guaranteed by the use of 2 Virtual Networks and Turn Prohibition; while the Virtual Source Routing offers sufficient adaptability to support the high fault tolerance. Finally, the Echo Mode guarantees that a path will be discovered, even under complex failure patterns.

In addition, the figures show that the average latency tends to increase with the interconnect size and the failure rate. For

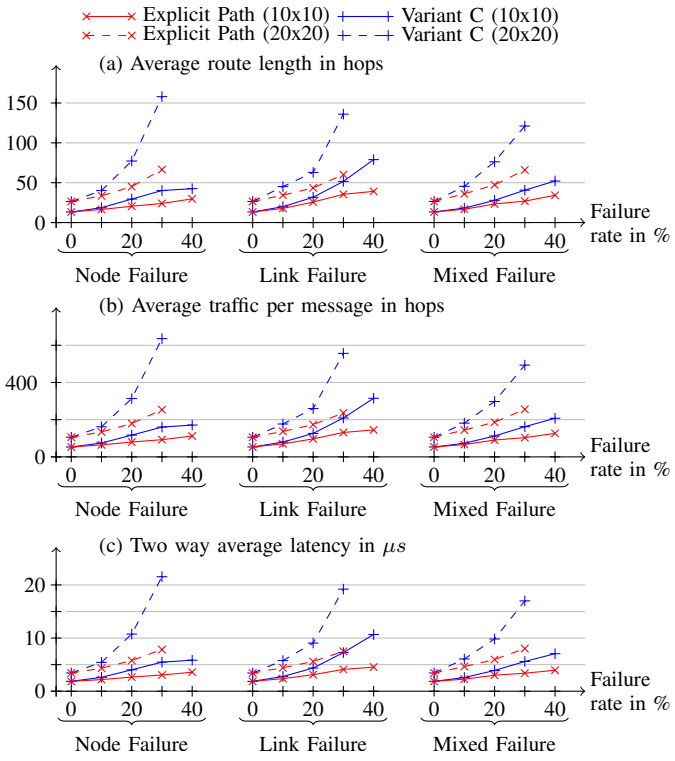


Figure 6. Comparison between the proposed algorithm with Explicit Path Routing Mode and the Variant C algorithm presented in [7] under Node, Link and Mixed failures

streaming applications, which require large messages between few different source - destination node sets, the Explicit Path Routing Mode limits traffic increase in case of failures. Thus, the average latency is improved significantly, based on the proposed deadlock-free fault-tolerant adaptive routing.

References

[1] R. Marculescu, U. Ogras, L.-S. Peh, N. Jerger, and Y. Hoskote, "Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspectives," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 1, p. 3–21, Jan. 2009.

[2] M. Azimi, D. Dai, A. Kumar, A. Mejia, D. Park, R. Saharoy, and A. Vaidya, "Flexible and adaptive On-Chip interconnect for Tera-scale architectures," *Intel Technology Journal*, vol. 13, no. 4, 2009.

[3] T. Dumitras, S. Kerner, and R. Marculescu, "Towards On-Chip Fault-Tolerant Communication," in *Proc. Asia & South Pacific Design Automation Conf. (ASP-DAC)*, January 2003.

[4] Y. B. Kim and Y.-B. Kim, "Fault Tolerant Source Routing for Network-on-chip," *IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems*, p. 12–20, Sept. 2007.

[5] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks* . Elsiever, 2004.

[6] D. Fick, A. DeOrio, G. Chen, V. Bertacco, D. Sylvester, and D. Blaauw, "A Highly Resilient Routing Algorithm for Fault-Tolerant NoCs," in *Design, Automation and Test in Europe*, no. 2.2, April 2009.

[7] F. Chaix, D. Avresky, N. Zergainoh, and M. Nicolaidis, " Fault-tolerant deadlock-free adaptive routing for any set of link and node failures in Multi-Cores systems," in *IEEE International Symposium on Network Computing and Applications*, july 2010.

[8] D.R. Avresky, C.M. Cunningham, and H. Ravichandran, "Fault-tolerant adaptive routing for two-dimensional meshes," *Int. Journal of Computer Systems Science and Engineering*, vol. 14, no. 6, november 1999.

[9] SocLib library. [Online]. Available: http://www.soclib.fr/