The CSI Journal on
Computer Science and Engineering
Vol. 6, No. 2 & 4 (b), 2008
Pages 36-45
Short Paper

Deadlock-Free Path-Based Fault-Tolerant Multicast Communications on 2-D Mesh Networks-on-Chip

Majed ValadBeigi Farshad Safaei

Department of Electrical & Computer Engineering, Shahid Beheshti University, Tehran, Iran

Abstract

The move towards nanoscale Integrated Circuits (ICs) increases performance and capacity, but poses process variation and reliability challenges which may cause several faults on routers in Networks-on-Chip (NoCs). While utilizing healthy routers in an NoCs is desirable, faulty regions with different shapes are formed gathering faulty routers. Fault regions can be used to lead the fault-tolerant routing algorithms to perform data transmission between healthy routers. Besides, the increasing number of collective communication-based services with a mass interest and the parallel increasing demand for service quality are paving the way toward end-to-end quality of service guarantees. The collective communication services embrace *multicast* (the same message is sent from a source node to an arbitrary number of destination nodes). This paper presents the fault-tolerant mesh-based wormhole-switched NoCs which supports deadlock-free path-based multicast routing. This technique fills the gap of not having optimum multicast fault-tolerant wormhole-switched approaches and also maintains minimal routing path which tolerates both convex and concave fault regions while keeping the area and power overhead at a proper level. Multicast messages are injected to the network by sending a message header beforehand. The message header contains destination addresses to set-up multicast path connecting a source with multiple destination nodes. Moreover, the fault information is kept locally and each fault-free processor needs to know the status of the link incident on it only. Performance of the proposed methodology is simulated under different network conditions and show that it degrades gracefully in the presence of various fault patterns.

Keywords: Networks-on-Chip, Fault-Tolerance, Multicast Communications, Multicast Path-Based Routing, Wormhole Message Passing, Faulty Patterns, Performance Evaluation.

1. Introduction

Networks-on-Chip (NoCs) is a challenging research topic, providing a scalable solution for multiprocessors on chip [1]. The lack of scalability in bus-based systems and large area overhead of point-to-point dedicated links on one hand, and the scalability and performance of switch-based networks and packet-based communication in the internet and traditional parallel computing on the other hand, have motivated researchers to propose packet-switched NoCs architectures to overcome complex on-chip communication problems [2].

Although the concept of NoCs is inspired from the networks in parallel processors, they have some specific properties which differ from the traditional networks. Compared to traditional networks, *power consumption* and *fault-tolerance* are becoming increasingly important constraints in NoCs design [2]. With the shrinking feature size, the reliability becomes more difficult to achieve due to smaller voltage supplies and higher amount of computing elements per area unit. It results in smaller noise immunity and increased risk of cross-talk and other errors [2-4]. Moreover, critical leakage currents, high field effects, and process variation will lead to more transient and permanent failures of signals, logic values and devices and interconnects [5].

As communication infrastructure is an important issue of today's Systems-on-Chip (SoCs) and Chip-Multiprocessors (CMPs), fault-tolerance has to be taken into account along with design of this part of chips. Having the same concept as traditional interconnection networks, the resources used in traditional networks in order to achieve fault-tolerance are not easily available in VLSI chips. Routing algorithm, as one

of the most important aspects of NoCs, has been exploited by many researchers to guarantee reliable operation of the network [6-8], by bypassing the faulty nodes and links.

Services in terms of efficient routing and scheduling are critical with respect to the performance of NoCs-based multicore processor systems. Historically, the first-generation multicomputer supported unicast communication only (single PE sends a message to single PE unit). Nowadays, multicomputers have been developed towards *collective communication* services. The collective communication services embrace *multicast* (the same message is sent from a source node to an arbitrary number of destination nodes).

In *path-based multicast* scheme, a source node prepares a message for delivery to a set of destinations by first sorting the addresses of the destinations in the order in which they are to be delivered and then placing this sorted list in the header flits of message. When the header enters a router with address α, the router checks to see if α is the next address in the header. If so, the address α is removed from the message header and the data flits are forwarded both to the local processor at this node as well as to the next node on the path. Otherwise, the message is forwarded only to the next node on the path. In this way, the message is eventually delivered to every destination in the header [9].

In this paper, we study the problem of fault-tolerant multicast communication for wormhole-switched NoCs. This is an important problem since multicast communication is a natural communication primitive to handle synchronizations, invalidations and updates of cache lines in NoCs applications and also because the NoCs performance is closely related to the robustness of the fault-tolerant routing algorithms of such networks.

Regarding the complexity, having no guarantee Quality-of-Service and lack of fault-tolerant multicast communications in NoCs, we propose, implement and evaluate fault-tolerant wormhole-switched NoCs which supports deadlock-free path-based multicast routing that leads to easy handling of faults and also preventing static fault patterns in NoCs with 2-D mesh topology. The rest of the paper is organized as follows. Section 2 describes the necessary background information that is used in the paper. In Section 3, we review previous relevant works. Section 4 presents the brief of the main contribution of the paper.

Our fault-tolerant routing methodology which is coupled with multicast communication is introduced in Section 5. In Section 6, the experimental results with different fault regions and other related parameters are presented. Finally, in Section 7 we conclude by summarizing our main contribution and future works.

2. Preliminaries

This section reviews some definitions and background of the mesh and describes the concept of fault models and fault patterns used in this paper.

2.1. The Mesh Networks

The topology of a network defines how the nodes are interconnected and is generally modeled as a graph in which the vertices represent the nodes and the edges denote the communication lines. The NoC concept has been proposed to overcome the problems of traditional wire/bus-based interconnection [3]. The NoC uses interconnection networks to connect intellectual properties (IPs) and to cope with the increased size and complexity of IPs.

Besides, different topologies of interconnection networks can provide different features in NoCs. Among different topologies the 2-D mesh is one of the most popular topology, in which the routers of NoCs are connected as a 2-D array and each IP is connected to an individual router. The mesh networks, the tree networks, the star networks and the simple loop networks frequently appear in various applications of networks. Many interconnection networks are constructed by Cartesian product [10].

Definition 1 [10]: Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs. The Cartesian product of G_1 and G_2, denoted by $G_1 \times G_2$, is the graph with vertex set $V_1 \times V_2$ such that (u_1, v_1) is joined to $(u_2, v_2)k$ times if and only if either $u_1 = u_2$ and v_1 is joined to $v_2 k$ times in G_2 or $v_1 = v_2$ and u_1 is joined to $u_2 k$ times in G_1.

Definition 2 [10]: The topological structure of a 2-D mesh network, $M(m_1, m_2)$ is defined as the Cartesian product of two paths $P_{m_1} \times P_{m_2}$, where P_{m_i} is the path graph with m_i vertices.

Due to its modularity and symmetry in terms of link length, the mesh topology is one of the most desirable topologies regarding to characteristics; the number of links per node does not change if additional nodes are added to the mesh. Therefore, the mesh topology offers very good scalability. Additionally, the mesh's low cost is because this network topology consists of fewer links per node than most other architectures.

2.2. The Fault Model

Fault-tolerance is defined as the ability of a system to continue operation despite the presence of faults [8]. In this sense, fault-tolerance is closely related to concepts such as reliability, availability and dependability as it may serve to provide these features.

Faults in a network take many forms, such as hardware faults, software bugs, malicious sniffing or removal of packets. The first step in dealing with errors is to understand the nature of component failures and then to develop simple models that allow us to reason about the failure and the methods for handling it. Classification of faults by nature is either *random* or *systematic* faults. Random faults are usually hardware faults affecting the system components, which occur with a certain probability, while systematic faults such as software failures are faults which are not random, whether a component has it or not [8]. We assume that such permanent failures are detected and contained on a node or link boundary. Thus, faults are assumed to be fail-stop [11], meaning that we do not consider Byzantine (i.e., malicious) faults [8]. In the contexts of fault-tolerant routing, these are common assumptions [8, 11-13].

Faults can also be classified by their duration as *transient* and *permanent* faults [8]. Transient faults persist in the system for only a short duration while permanent faults

remain in the system until it is repaired and may be either *dynamic* or *static*. In a dynamic fault model, once a new fault is found, actions are taken in order to appropriately handle the faulty component which allows the system to reconfigure at the hardware level and preserves the original network topology. A static system provides a fault-tolerant routing algorithm that will bypass any failed node.

2.3. The Fault Pattern

To simplify the routing algorithm, adjacent faulty nodes are coalesced into *fault regions*, which may lead to different patterns of failed components. To analyze the performance of fault-tolerant routing algorithms, it is important to identify and quantify the fault regions, which may occur in the network. The shapes of such fault regions are often restricted by the fault model in use. Furthermore, the fault model may impose additional restrictions on the locations of the faults. For instance faults may not be allowed on the edges of the mesh and there may be a minimum distance between separate fault regions. A routing algorithm applying such a fault model is generally to tolerate all fault combinations conforming to the fault model, thus, the provided fault-tolerance is defined by the fault model. In the case that a fault combination is not conforming to the fault model in use, healthy nodes are marked as faulty (i.e., disabled) in order to create proper fault regions. Although healthy nodes are disabled, a fault combination is considered to be tolerated as long as all the nodes that are neither faulty nor disabled are connected through valid paths. Fault regions extended by faulty components may form *convex* (also referred to as the block fault model) or *concave* shaped fault patterns [8, 11-14].

Definition 3 [8, 11-14]**:** A convex region is defined as a region $\mathcal{R}$ in which a line segment connecting any two points in $\mathcal{R}$ lies entirely within $\mathcal{R}$. If we change the "line segment" in the standard convex region definition to "horizontal or vertical line segment", the resulted region is called rectilinear convex segments [12, 13]. Any region that is not convex is a concave region.

Examples of convex regions are |-shape, □-shape and concave regions are L-shape, U-shape, T-shape, H-shape and +-shape. The convex fault model is the simplest fault model in mesh topologies. Under the convex fault model, all fault regions are required to have a rectangular shape in a 2-D mesh and a cuboid shape in a 3-D mesh. Convex faults can be created by marking a node faulty if it has a faulty neighbor/channel in at least two dimensions [11].

A stricter version of the convex fault model is when to requiring all fault regions to form a square [8]. Because a node with at most one faulty or disabled neighbor is not able to determine whether it is part of such a region solely based on the status of its faulty neighbors, such fault regions are more complex to construct and maintain than pure block faults if used with a dynamic fault model [11]. Furthermore, such a fault model is likely to require more nodes to be disabled. The experimental results give some indications on the percentage of healthy nodes that are disabled in a 10×10 mesh when using square fault regions [11]. In particular, when 10% of the routers are faulty, on average, less than 12% of the non-faulty nodes are marked as faulty, and the entire network becomes disabled when about 20% of the nodes are faulty. Although it may appear that such a method is able to tolerate almost 20% of the routers being faulty, it is not likely that the NoC is able to perform satisfactory with such a high proportion of the routers disabled. In order to reduce the number of disabled healthy nodes, concave fault regions may be used.

By not requiring the nodes within such shapes to be disabled, fault-tolerant routing methods supporting concave faults are able to tolerate a wide range of fault patterns without disabling healthy routers. Specifically, such routing algorithms provide additional support for concave fault patterns compared to the convex fault model by also tolerating concave fault regions. Still, such methods may not be able to tolerate arbitrary fault patterns and may put some restrictions on the shape of the concavities.

3. Related Work

In this section, we initially provide a brief survey of fault-tolerant routing algorithms, multicast routings and multicast fault-tolerant routings that have been proposed based on mesh topologies. The survey is by no means exhaustive, but serves to provide a general overview of different approaches. We only discuss routing algorithms that are targeted at networks with lossless flow control that is routing algorithms that take deadlocks into account.

Lots of fault-tolerant routing algorithms based on traditional interconnection networks can be performed against static faults in NoCs. In designing of a fault-tolerant routing algorithm, a suitable *fault model* is one of the most substantial issues. The fault model can reflect the fault situations in a real system. Rectangular blocks (i.e., convex faults) are the most common approach to model node failures and to facilitate routings in 2-D meshes.

A labeling scheme [14] has been proposed to construct a fault region with rectangular shapes. However, rectangular fault patterns include many healthy routers. In order to reduce healthy routers in rectangular fault regions, many related works are proposed in the literature.

Boppana and Chalasani [15] proposed a method to tolerate block faults in 2-D meshes (with dimension-order routing). This method is able to tolerate overlapping *f-rings*/*chains* when using four virtual channels and non-overlapping f-rings when using two virtual channels. Fault regions are also used by Boura and Das [16] who provide support for fully adaptive fault-tolerant routing in *n*-D meshes using three virtual channels. This method is based on an adaptive routing algorithm previously proposed by the same authors [17]. Because rectangular fault regions are used, they also mark healthy routers as faulty in order to establish fault regions. However, their method also has a reactivation mechanism, enabling some nodes that have been marked as faulty to be reactivated if they are directly connected to a node outside the fault region.

Chalasani and Boppana [18] extended their method to support solid fault regions using four virtual channels. On the other hand, this new method does not support overlapping f-rings or faults on the edges of the mesh. Such a support is provided by Kim and Han [19], however, still using the same number of virtual channels (i.e., four). Chen and Chiu [20], later, managed to reduce the number of virtual channels to

three, still tolerating overlapping solid faults and faults on the edges. Wu [21] proposed convex regions with T-shape, L-shape and +-shape fault regions.

Besides, there are some multicast routing algorithms based on traditional interconnection networks that can be performed in NoCs. A path-based multicast routing algorithm has been proposed in [22, 23]. In this algorithm, multicast messages are transmitted on one of the two sub networks H_u, H_l and, each is defining a Hamiltonian path in the network. The dual-path algorithm in [23] uses at most two copies of the multicast message. The multipath algorithm attempts to reduce long latencies by setting up to four copies ($2n$ for *n*-dimensional) of the original multicast message. As per the multipath routing algorithm, all the destinations of the multicast message are grouped to four disjoint subsets. Each subset of destinations is serviced by one copy of the multicast messages [24].

The column-path algorithm partitions the set of destinations of a multicast message into at most 2*k* subsets (*k* is the number of columns in the mesh), such that there are at most two messages directed to each column. If a column of the mesh has one or more destinations in the same row or in rows above that of the source, then one copy of the message is sent to service all those destinations. Similarly, if a column has one or more destinations in the rows below that of the source, then one copy of the message is sent to service all those destinations. One copy of the message is sent to a column if all destinations in that column are either below or above the source node; otherwise, two messages are sent to that column [25]. Moreover, Panda et al. [26] have proposed a general technique to provide multicast routing capability using the existing unicast routing method.

However, there are few results that exist on multicast fault-tolerant routing. Boppana and Chalasani proposed a fault-tolerant routing of multicast messages in mesh-based wormhole-switched interconnection networks and this method can tolerate multiple convex faults [27]. Also, Chen and Wu [9] presented a dead-lock free path-based fault-tolerant multicast algorithm in 2-D meshes that consider faulty block model.

4. Contribution

NoCs are being proposed as a global communication platform for future SoCs applications. Among the merits are: scalability, reusability and etc. Buses (a single bus, segmented buses or a hierarchy of buses) do not scale with the system size in bandwidth and clocking frequency. However, it is very efficient in broadcasting. Although a network allows many more concurrent transactions, implementing multicast may be inefficient if not addressed properly. The normal multicasting scheme transfers multicast messages by routers in the same way as unicast messages. This scheme does not guarantee Quality-of-Service (QoS) and multicast fault-tolerant routing. Besides, according to the previous section few relevant work exist on multicast fault-tolerant routing. Based on this observation, a fault-tolerant routing algorithm supporting multicast scheme is proposed with arbitrary fault patterns. The proposed algorithm is more efficient than previous work and also supports minimal routing path. It also tolerates both convex and concave fault regions while keeping the area and power overhead at a proper level.

5. A Multicast Fault-Tolerant Routing

In this section we present and evaluate a fault-tolerant wormhole-switched on NoCs which supports deadlock-free path-based multicast routing and it is able to tolerate a reasonable number of convex and concave fault patterns without disabling healthy routers. In order to achieve this scope in a simple manner, thereby reducing the costs due to additional virtual channels and increased router complexity, a static fault model [8] was chosen.

Our discussion in this section is primarily on the concept of message passing after that the idea of *hole* is described. Then, the explanation of our methodology algorithm and finally a brief description on deadlock-freedom of the proposed algorithm are presented.

5.1. The Concept of Wormhole Message Passing

The concept of wormhole message passing mechanism with multiple destinations was used in [28] to perform fast multicast operations with Hamilton-path-based routing. Under this mechanism, the header of a worm consists of multiple destinations. The *sender* node creates these destinations as an ordered list according to their intended order of traversal. As soon as the worm is injected into the network, it is routed based on the address in the leading header flit corresponding to the first destination. Once the worm reaches the router of the first destination node, the flit containing this address is removed by the router. Now the worm is routed to the node whose address is contained in the next header flit (second destination in the ordered list). While the flits are being forwarded by the router of the first destination node to its adjacent router, they are also copied flit-by-flit to the system buffer of this node. This process is carried out in each intermediate destination node of the ordered list. When the message reaches the last destination, it is not routed any further and gets completely consumed by the last destination node [26].

5.2. The Idea of Hole

Let us define some notation to illustrate how the proposed method works.

- $\mathcal{F}_i$: The set of healthy nodes which contains any fault regions.
- $\mathcal{H}_i$: The set of healthy nodes inside $\mathcal{F}_i$, in which nodes inside $\mathcal{F}_i$ consist of faulty nodes and $\mathcal{H}_i$.
- $\mathcal{N}$: A semi-faulty node. A node is semi-faulty if has two or more faulty or semi-faulty neighbors. $\mathcal{N}$ is the set of semi-faulty nodes in the network.

A fault labeling mechanism is considered to specify nodes status for routing decisions. Hence, each router is labeled as *F* (first), *H* (hole), or *E* (entrance) node. Fault labeling scheme leads to a general fault model which can circumvent faulty nodes.

• $\mathcal{N}^f$: The set of first destination nodes in ordered list inside $\mathcal{F}_i$;

$$\mathcal{N}^f = \{n : \exists d_1, d_2 \ni d_1 \neq d_2^-, (n)^{d_1}, (n)^{d_2} \in \mathcal{F}_i\} \quad (1)$$

where d_1, d_2^- are two orthogonal directions of the node n. The above expression implies that there are two directions d_1, d_2 which are not in the same dimension and the neighbor of the node n on these two directions is an element of $\mathcal{F}_i$.

• $\mathcal{N}^e$: the set of entrance-nodes;

$$\mathcal{N}^e = \{n : \exists d \ni (n)^d \in \mathcal{X}, n \notin \mathcal{H}_i\} \quad (2)$$

Each node in $\mathcal{N}^e$ has only one semi-faulty neighbor.

• $\mathcal{N}^h$: the set of hole-nodes;

$$\mathcal{N}^h = \{n : n \in \mathcal{H}_i, n \notin \mathcal{N}^f\} \quad (3)$$

• *Class*: A set of one entrance-node, one first destination node in ordered list and zero or more hole-node (s).

Classes may contain some nodes in common. Thus, we give each class a unique ID in form of (x, y, z), which (x, y) specifies coordinates of the entrance-node and z denotes dimension of the class. A message will enter in a class if its first destination in ordered list belong to the class and a message will depart a class if its first destination in ordered list are outside the class. Deadlocks are avoided inside the classes by using this scheme. Also, first destination nodes in ordered list have higher priority than hole or entrance-nodes. Thus, a node labeled as first destination node in ordered list at a time, it cannot be labeled as hole or entrance-node any after.

5.3. The MEFH Methodology

In this section, we propose a multicast fault-tolerant routing methodology, namely *MEFH*, which routes messages successfully to all destinations in the presence of more relaxed fault regions than those in [15, 27] without any additional virtual channels, as long as the network is not partitioned by fault regions. Further, it can be applied to any fault-tolerant routing algorithm that employs the concept of f-ring. This algorithm in the worst case is so similar to the fault- tolerant Hamiltonian path routing which is expressed in [28]. Note that we have mostly focused on the first destination node in ordered list which can easily be applied to all other nodes according to the concept of wormhole message passing.

5.3.1. Set-up the Classes

Node fault labeling technique is required as pre-processing task before the restart of the system, after the failure of one or more nodes or links. In this process, the internal nodes of the fault region are detected, and their addresses and locations are sent to the entrance-nodes to convert fault regions into rectangular shaped regions to facilitate message routing.

The classes can be easily formed in two approaches: top-to-bottom and bottom-to-top. In the first approach, the entrance-node (*E*) sends the message for establishing the class along the direction that its neighbor is an element of $\mathcal{X}$ (i.e., the set of semi-faulty nodes) and poses its address into the message header. All receiving nodes will include their own addresses to the header and forward the message along the opposite direction. This procedure is done in recursive manner until the message reaches the first destination node (*F*) in the ordered list and this node sends an ACK message back to the entrance-node (*E*).

In the second approach, the role of the first destination node in the ordered list and entrance- nodes is switched. Both of the approaches can be employed, but the complexity of the first approach is less, since in the first approach, the node *E* knows the direction of its class and sends massage on the same direction, but in the second approach, *F* will find the appropriate direction possibly after a number of retrials.

In the following treatments, we outline the fundamental steps that take place if the first approach is used.

Step 1: The entrance-node (*E*) generates a class creating message which contains class number. As mentioned before, the class number contains a triple vector which keeps the location details of *E* and the direction of sending the message. For example, If *E* is located at (0, 2) and sends message along the Y-dimension, the class number will be (0, 2, Y). Then, the message is transmitted to neighboring node x ($x \in \mathcal{X}$). Step 1 is repeated till all the entrance- nodes which belong to $\mathcal{N}^e$ succeed to transmit their related messages.

Step 2: In this step each node that has received the generated messages from Step 1, looks whether it is the first node or not. If it is, it must record the message and transmit an ACK message to *E* on the direction in which it received the message. Otherwise, the message will be forwarded on the normal way.

Figure 1. illustrates how the fault labeling scheme works.

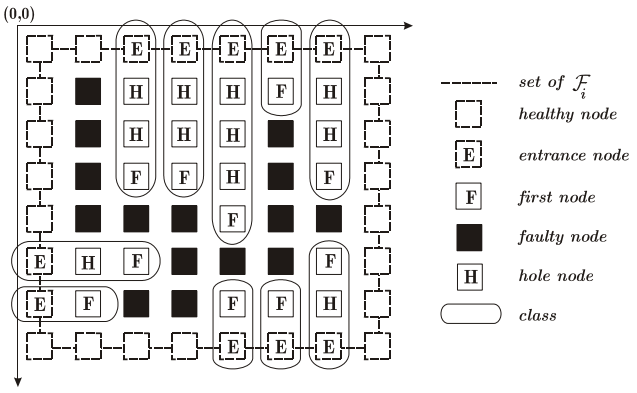

Figure 1. Application of the node fault labeling scheme in an 8×8 2-D mesh (note that the first node in figure means first destination node in order list)

Step 3: All nodes that receive the ACK message ought to maintain some essential history information regarding the path that the message has taken. This information comprised the addresses and the class number within the message. Carrying additional information usually imposes some overhead, unless header has enough bits to accommodate the additional information. When ACK reaches *E*, it is not stored and discarded. Only the information of all existing nodes of each-class is recorded in nodes of type *E*.

5.3.2. Constructing an Appropriate Multicast-Path

In this section a method is defined which is about how the destination address is considered in the message header to find a proper path to send the multicast messages. Here is the description of the method while we assume all destinations in just one list.

The list of destinations in a multidestination worm is ordered by the sender node. The ordering depends on two parameters:

1) Multicast-Labeling number of the node
2) Subnetwork priority

Consider a multidestination worm with n destinations from a source s with a destination list $\{d_1, d_2, ..., d_{n-1}, d_n\}$, the ordered destination list is $\{d'_1, d'_2, ..., d'_{n-1}, d'_n\}$ and is given by the following definition.

Definition 4: The Multicast-Labeling assignment function l for $R \times C$ mesh can be expressed in terms of x and y coordinates of nodes as

$$l(x,y) = \begin{cases} y \times C + x & \text{if } y \text{ is even} \\ y \times C + C - x - 1 & \text{if } y \text{ is odd} \end{cases} \quad (4)$$

Definition 5: The sub network priority $w(x)$ for $R \times C$ mesh is given by maximum of number of destinations in each subnetwork which x is expressed as the number of subnetwork.

Initially we divided the network to n^2 subnetworks (n is the dimension of network) and we obtain the number of nodes for each subnetwork. Then, we find in which subnetwork our source is located. Later on, we update the set of destinations in a way that all the existing nodes (in the chosen subnetwork where the source is located) can be placed in an ascending manner by considering their multicast-labeling number. After locating all existing nodes in the set, we will consider the latest chosen destination as the new source. Then the set, in which the closet node to the new source exists, is chosen and put through the previous process. After reaching the latest node of the destination set, the updated set from destination is chosen and is sent with the message. The message starts routing and meets every node according to the ordering destination sets.

Remark: If the distance between the current source node $source = d_s$ and d_k, d_l are equal in which d_k, d_l are in two different subnetworks the sub network will be chosen whose w is more than the other. An example of the method is illustrated in Figure 2.

5.3.3. Routing Rules

Here we assume that the network supports a routing scheme $\Re$. Even when there are faults in the network, each message is routed using the $\Re$ algorithm as much as possible. When a message arrives at a node, the next hop for that message is specified by the ordered destination set which is explained before. If that hop is on a faulty link, then the message is blocked by the fault. The routing logic is enhanced to handle such situations so that the message is routed around faults. Once the message is routed around faults, the $\Re$ algorithm is used to route the message until it reaches all of its destinations.

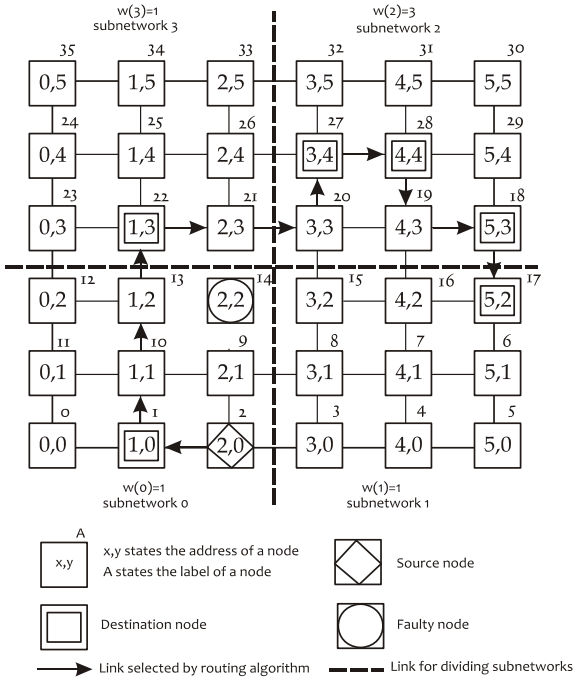

Figure 2. Examples of the proposed algorithm. There is one faulty node. The path of multicast message from (2,0) to (1,0), (1,3), (3,4), (4,4), (5,3), (5,2)

Here we assume that in each routing rules the source node is denoted as n_s and the first destination node in ordered list as n_d. Note that all the coming rules will be extended to all destinations in the ordered list by applying the steps of which we explained before. Considering the following rules, a proper routing algorithm can circumvent any fault patterns without disabling healthy nodes:

a) If all fault patterns are convex, all messages will be routed by the basic fault-tolerant routing algorithm.

b) If there exists some fault regions but n_s and n_d are not inside $\mathcal{F}_i$, the message will be routed by the basic fault-tolerant routing algorithm.

c) If n_d is inside $\mathcal{F}_i$ and n_s is not, a misrouting mechanism will be applied. Nodes outside $\mathcal{F}_i$ have no knowledge about nodes inside $\mathcal{F}_i$ and cannot find the proper *E* node. As a result, a misrouting mechanism can be used to reach destination class entrance-node. Misrouting without any threshold, may lead to livelock. The maximum misrouting hops according to the basic fault-tolerant routing algorithm should be defined. When a message passes maximum legal misrouting hops, it would be absorbed by the current node and waits for a period of time to be re-injected into the network. Absorbed messages have greater priority than normal messages for re-entering into the network. In this way, starvation will be avoided. Here, we consider delay in local buffer to re-inject messages equal to the message length.

d) If n_s is inside $\mathcal{F}_i$ and n_d is not, the message will be forwarded to *E* node first, and then will be routed to n_d using the basic fault-tolerant routing algorithm.

e) If both n_s and n_d are inside $\mathcal{F}_i$, one of these situations will be encountered: n_s and n_d belong to a same class, or n_s and n_d belong to different classes. For the first situation, the message will be forwarded to n_d by dimension-order-routing algorithm. For the second situation, the message will be forwarded to its class *E* node. Afterward, it will be misrouted to reach destination class *E* node. Finally, it is forwarded from *E* node to n_d.

5.4. Deadlock Avoidance

Multicast fault tolerant routing is the topic of this paper. Being able to utilize the network without introducing any risk of deadlock and decrease unnecessary costs at the same time are the crucial challenges when designing this kind of routing scheme. Hence, the deadlock-freedom is an essential attribute of any Multicast fault-tolerant routing algorithm. The following part of this section is devoted to proving that the proposed method (i.e., *MEFH* methodology) is deadlock-free. The following theorem proves the acyclicity of the channel dependency graph of the proposed multicast fault-tolerant routing methodology.

Lemma 1: The multicast fault-tolerant routing algorithm based on the MEFH methodology causes no deadlock in 2-D meshes that contains any fault region.

Proof: We initially consider the source node and the first destination node in ordered list here. According to wormhole message passing and path-based multicast scheme, when the message reaches the first destination node address it will be the new source and will be removed from destinations list. Then, the second destination node becomes the first destination node in ordered list. As a result, only the proof for the source and the first destination node is explained here. This approach can be applied to all destinations. Consequently lemma will be proved.

According to the positions of n_s and n_d, the proof falls into one of the following three cases:

Case I: n_s and n_d are inside $\mathcal{F}_i$. In such case, no changes are made on the deadlock-free original fault-tolerant algorithm. Consequently, the original fault-tolerant routing algorithm guarantees deadlock-freedom.

Case II: n_s or n_d is inside $\mathcal{F}_i$. In this case, the routing strategy is divided into two phases. First, is that a part of the course of message transmission is in the class and another part is outside the class. The $n_e \in \mathcal{N}^e$ node is the connecting point of these two phases. According to the routing rules in an attempt to find a path to forward a message outside the class, the original algorithm is deadlock-free. The second phase is that routing is done inside the class. According to the characteristics of the class, we had assumed that classes are isolated and path selection scheme is taken on one dimension; there is no misrouting operation, no cycle of deadlocked messages exists and deadlock-freedom can be assured.

Case III: Both n_s and n_d nodes are inside $\mathcal{F}_i$. If both n_s and n_d are in the same class, the message is delivered explicitly to its destination without producing any misrouting. Hence, in this case no deadlock will occur, otherwise it is handled similar to Case II.

6. Simulation Study

In this section, we analyze how the proposed method influences the network performance for different fault models. Hence, we will first describe the simulation model and then the performance results are shown and evaluated.

A flit-level, listener-based object oriented simulator [29] has been used to illustrate the performance of the presented method. An 8×8 mesh network with point to point and bidirectional links has been employed for all simulation experiments, and only wormhole switching mechanism is applied.

We have fixed the number of destinations at 15. The wormhole switching is considered as the flow control mechanism. When wormhole flow control is applied, a message is not allowed to re-enter the adaptive layer after having been routed in the escape layer. The simulations have been performed using a base message size of 8 flits, and also the width of each flit is 128 bit. We calculated the power consumption of links of each router in 100 nm technology library. In this technology, V_{DD} is set to 1.1V, the clock frequency is set to 200 MHz, and the length of links between two adjacent routers is set to 1 mm for the mesh topology. The message generation rate is also equal for all the nodes. The performance and power consumption of our method is evaluated by terms of average message latency (cycles) and total power consumption (mW) vs. traffic load. Boppana-Chalasani's fault-tolerant routing [15, 18] is used to govern routing on a faulty network. As we have previously discussed, the routing methodology requires additional virtual channels in order to be deadlock-free. Thus, 4 virtual channels with 8 flits deep buffers are required by each variation of the methodology to avoid deadlock phenomenon.

We have also evaluated the fault-tolerance of the H-shape, L-shape, Block-shape, and overlapped fault patterns in a 2-D (i.e., 8×8) mesh topology, as shown in Table 1. For this topology, all the combinations up to six faults (i.e., 10% of the total number of routers) have been exhaustively analyzed. Results for up to 10% faults are shown in Figures 3 and 4 and the confidence intervals are always lower than ±5%.

Table 1. Different variations of the fault combinations that utilize the methodology

Fault pattern	Faulty points set
H-shape	(2,1)(2,2)(2,3)(3,2)(4,1)(4,2)(4,3)
Overlapped	(2,2)(2,3)(3,3)(3,5)(4,3)(4,5)(4,6)
L-shape	(2,1)(2,2)(2,3)(2,4)(3,4)(4,4)(5,4)
Block-shape	(2,2)(2,3)(2,4)(3,2)(3,3)(3,4)(4,2)(4,3)(4,4)

Before explaining both figures and results, we would like to stress the point that simulation studies show that our approach has a good performance in order to support multicast. Moreover, both Figure 3 and Figure 4 show a better performance with 15 destinations compared to unicast fault tolerant routing that depends on one source and one destination.

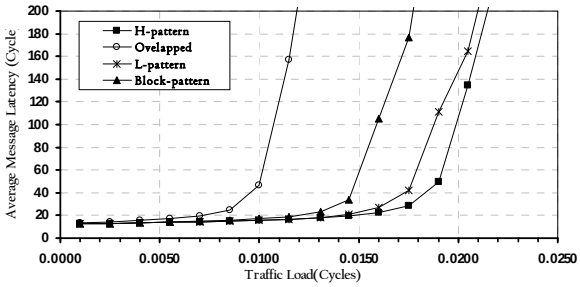

Figure 3. Average message latencies vs. traffic load obtained by the proposed methodology in an 8×8 Mesh with 15 destinations and 4 virtual channels for each fault pattern listed in Table 1

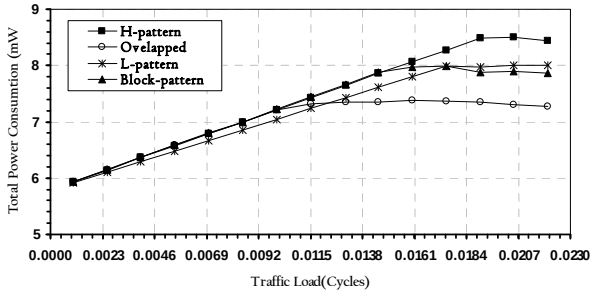

Figure 4. Average power consumption vs. traffic load in an 8×8 Mesh with 15 destinations and 4 virtual channels obtained by the proposed methodology for each fault pattern listed in Table 1

Figure 3 shows the achieved message latency over normalized accepted traffic in an 8×8 mesh with 15 destinations for 4 different shapes of fault patterns presented in Table 1. In this configuration, we used 4 virtual channels per physical channel. The figure reveals that for low traffic load, all four fault patterns have the same latency, but they begin to behave differently around the moderate and saturation regions. At overlapped pattern, congested overlapped nodes between two f-rings lead to increase the average message latency and it can be seen that the worst case has been encountered for overlapped pattern.

Figure 4 compares the power consumption of the network for different variations of the proposed methodology as a function of traffic load by each node in an 8×8 mesh with 15 destinations and 4 virtual channels per physical link. Such curves provide an indication on the degree of performance degradation caused by fault patterns in the network. It is evident that, as the traffic load in the network increases, the power dissipation diagram of each pattern increases almost linearly before reaching a point on the curve. Beyond this point, the dissipated power decreases abruptly. According to the diagram we witness high level of initial consumption of power that is due to the construction of the path. Then the power consumption is reduced and reaches a plateau because the path construction is completed and there is no need for the extra amount of power. Among fault shapes, the Block-pattern is a situation that we just use basic fault- tolerant routing algorithm for H-pattern. This means semi-faulty nodes in H-pattern will be considered faulty. As the figure demonstrates, however, the state of Block-pattern is better than H-pattern under 0~60% of traffic load. Beyond 60% of maximum traffic load, simulation results exhibit meaningful performance degradation for Block-pattern. The degradation reason is related to extra faulty nodes. From simulation studies we can conclude that our proposed methodology obtains proper performance degradation in the network.

7. Conclusions and Future Works

Due to the exponential growth of circuit integration, the reliability of NoCs is concerned with its functionality in the presence of given failure probabilities of their entities. NoCs reliability pertains to interconnection networks whose nodes and/or channels have associated probabilities of being operational. The main contribution of this article is that the proposed approach supports multicast routing which is based on path-based multicast scheme that easily can handle static faults. Moreover, the path used does not need to be reconstructed when a fault occurs. Further, our approach is deadlock-free, it supports the static fault model which is applicable to both convex and concave fault regions. This method fills the gap of not having an optimum multicast fault-tolerant scheme and this is the first attempt to implementing multicast fault-tolerant routing to support fault regions. In a near future we are planning to not only improve it but also to perform more research on the issue and find an optimum solution and also extend the design of path-based fault-tolerant multicast to cover dynamic faults in NoCs with 2-D mesh topology.

Acknowledgement

First Author would like to thank Mr. Shahed ValadBeigi for his useful comments and help.

References

[1] F. A. Samman, T. Hollstein, and M. Glesner, "Adaptive and Deadlock-Free Tree-Based Multicast Routing for Network-on-Chip," IEEE Transactions on Very Large Scale Integration Systems, vol. 18, no. 7, pp. 1067-1080, 2010.

[2] L. Benini and G. De Micheli, "Networks on Chip: a New Paradigm for Systems on Chip Design," *IEEE Transactions on Computers*, vol. 35, no. 1, pp. 70-78, 2002.

[3] G. De Micheli and L. Benini, *Networks on Chips*, Morgan Kaufmann-Publishers, 2006.

[4] C. Constantinescu, "Trends and Challenges in VLSI Circuit Reliability," *IEEE Micro*, vol. 23, no. 4, pp. 14-19, 2003.

[5] T. Dumitras, S. Kerner, and R. Marculescu, "Towards on-Chip Fault-Tolerant Communication," *Proc. Asia and South Pacific Design Automation Conference*, pp. 225-232, 2003.

[6] Y. B. Kim, and Y. Kim, "Fault Tolerant Source Routing for Network-on-chip," *Proc. of the IEEE International*

Symposium on Defect and Fault-Tolerance in VLSI Systems, pp. 12-20, 2007.

[7] M. Ali, M. Welzl, M. Zwicknagl, and S. Hellebrand, "Considerations for Fault-Tolerant Network on Chips," Proc. of the *International Conference on* Microelectronics, pp. 13-15, 2005.

[8] J. Duato, S. Yalamanchili, and L. M. Ni, *Interconnection networks: An Engineering Approach*, Morgan Kaufmann Publishers, 2003.

[9] X. Chen and J. Wu, "Path-Based Fault-Tolerant Multicasting in Mesh-Connected Multicomputers," *The Handbook of Ad Hoc Wireless Networks*, CRC Press, 2002.

[10] L. H. Hsu and C. K. Lin, *Graph Theory and Interconnection Networks*, CRC Press, 2009.

[11] N. A. Nordbotten, *Fault-Tolerant Routing in Interconnection Networks*, Ph.D. Thesis, Faculty of Mathematics and Natural Sciences, The University of Oslo, 2008.

[12] J. Wu and Z. Jiang, "On Constructing the Minimum Orthogonal Convex Polygon for the Fault-Tolerant Routing in 2-D Faulty Meshes," *IEEE Transactions on Reliability*, vol. 54, no. 3, pp. 449-458, 2005.

[13] M. H. Farahabady, F. Safaei, A. Khonsari, and M. Fathy, "Characterization of Spatial Fault Patterns in Interconnection Networks," *Journal of Parallel Computing*, vol. 32, no. 11-12, pp. 886-901, 2006.

[14] A. A. Chien, and J. K. Kim, "Planar Adaptive Routing: Low Cost Adaptive Networks for Multiprocessors," *Proc. of the International Symposium on Computer Architecture*, pp. 268-277, 1992.

[15] R. V. Boppana, and S. Chalasani, "Fault-Tolerant Wormhole Routing Algorithms for Mesh Networks," *IEEE Transactions on Computers*, vol. 44, no. 7, pp. 848-864, 1995.

[16] Y. M. Boura, and C. R. Das, "Fault-Tolerant Routing in Mesh Networks," *Proc. of the International Conference on Parallel Processing*, pp. 106-109, 1995.

[17] Y. M. Boura, and C. R. Das, "Efficient Fully Adaptive Wormhole Routing in n-Dimensional Meshes," *Proc. of the International Conference on Distributed Computing Systems*, pp. 589–596, 1994.

[18] S. Chalasani, and R. V. Boppana, "Communication in Multicomputers with Nonconvex Faults," *IEEE Transactions on Computers*, vol. 46, no. 5, pp. 616–622, 1997.

[19] S. P. Kim, and T. Han, "Fault-Tolerant Wormhole Routing in Mesh with Overlapped Solid Fault Regions," *Parallel Computing*, vol. 23, no. 13, pp. 1937–1962, 1997.

[20] C. L. Chen, and G. M. Chiu, "A Fault-Tolerant Routing Scheme for Meshes with Nonconvex Faults," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 5, pp. 467–475, 2001.

[21] J. Wu, "A Fault-Tolerant and Deadlock-Free Routing Protocol in 2D Meshes Based on Odd-Even Turn Model," *IEEE Transactions on Computers*, vol. 52, no. 9, pp. 1154–1169, 2003.

[22] D. F. Robinson, P. K. Mckinley, and B. H. C. Cheng, "Path Based Multicast Communication in Wormhole Routed Unidirectional Tours Networks," *Journal of Parallel and Distributed computing*, vol. 45, pp. 104-121, 1997.

[23] X. Lin and L. M. Ni, "Deadlock-Free Multicast Wormhole Routing Multicomputer Networks," *Proc. of the International Symposium on Computer Architecture*, pp. 116-124, 1991.

[24] P. Mohapatra and V. Varavithya, "A Hardware Multicast Routing Algorithm for Two Dimensional Meshes," *Proc. of the IEEE Symposium on Parallel and Distributed Processing*, pp. 198–205, 1996.

[25] R. V. Boppana, S. Chalasani, and C. S. Raghavendra, "Resource Deadlock and Performance of Wormhole Multicast Routing Algorithms," IEEE Transactions on Parallel and Distributed Systems, vol. 9, no. 6, pp. 535–549, 1998.

[26] D. K. Panda, S. Singhal, and R. Keshavan, "Multidestination Message Passing in Wormhole *k*-Ary *n*-Cube Networks with Base Routing Conformed Paths," *IEEE Transactions on Parallel and Distributed Systems*, vol. 10, no. 1, pp. 76-96, 1999.

[27] R. V. Boppana and S. Chalasani, "Falut-Tolerant Multicast Comunication in Multicomputers," *Proc. of the IEEE International Conference on Parallel and Distributed Processing*, pp. 118-125, 1995.

[28] X. Lin, H. Xu, and A.-H. Esafahnian, "Performance Evaluation of Multicast Wormhole Routing in 2D-Mesh Multicomputers," *Proc. of the International Conference of Parallel Processing*, pp. 435-442, 1991.

[29] A. Nayebi, S. Meraji, A. Shamaei, and H. Sarbazi-Azad, "XMulator: an Object oOriented XML-Based Simulator," *Proc. of the Asia International Conference on Modeling and Simulation*, pp. 128–132, 2007.

Majed ValadBeigi received the B.Sc. degree in Computer Hardware Engineering from Shahid Rajaee University, Tehran, Iran, in 2008. He is currently M.Sc. student in Computer Architecture Engineering in Department of Electrical and Computer Engineering, Shahid Beheshti University, Tehran, Iran. He is a member of Scorpius simulation team which has participated in many international RoboCup Competitions. He is also an IEEE student member. His main Researches focus on network-on-chips, interconnection networks, and performance evaluation.

E-mail: m.valadbeigi@mail.sbu.ac.ir

Farshad Safaei received the B.Sc., M.Sc., and Ph.D. degrees in Computer Engineering from Iran University of Science and Technology (IUST) in 1994, 1997 and 2007, respectively. He is currently an assistant professor in the Department of Electrical and Computer Engineering, Shahid Beheshti University, Tehran, Iran. His research interests are performance modelling/evaluation, Interconnection networks, computer networks, and high performance computer architecture.

E-mail: f_safaei@sbu.ac.ir

Paper Handling Data:

Submitted: 10.20.2010
Received in revised form: 06.13.2011
Accepted: 06.20.2011
Corresponding author: Dr. Farshad Safaei,
Department of Electrical & Computer Engineering, Shahid Beheshti University, Tehran, Iran.