Microprocessors and Microsystems 45 (2016) 198–207

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

A runtime fault-tolerant routing algorithm based on region flooding in NoCs

Lu Wang[a,b,*], Sheng Ma[a,b], Zhiying Wang[a,b]

[a] State Key Laboratory of High Performance Computing, National University of Defense Technology, China
[b] College of Computer, National University of Defense Technology, Changsha, China

ARTICLE INFO

Article history:
Received 7 April 2015
Revised 9 May 2016
Accepted 12 May 2016
Available online 14 May 2016

Keywords:
Reliability
Network-on-chip
Fault tolerance
Communication protocol
Region flooding algorithm

ABSTRACT

Aggressive scaling of the CMOS process technology allows the fabrication of highly integrated chips, and enables the design of the network-on-chip (NoC). However, it also leads to widespread reliability problems. A reliable NoC system must operate normally even in the face of a lot of transistor failures. Aiming towards permanent faults on communication links, we introduce a fault-tolerant MPI-like communication protocol. It detects the link failure if there exist unresponsive requests and automatically starts the new path exploration. The region flooding algorithm is proposed to search for a fault-free path and reroute packets to avoid system stalls. The experimental result shows our approach significantly reduces the latency compared with the basic flooding algorithm. The maximum latency reduction is 25% under the bit complement traffic pattern. Also, it brings only 2% fault tolerance loss.

1. Introduction

The Moore's Law scaling is continuing to yield even higher transistor density with each succeeding process generation, leading the design of the network-on-chip (NoC). Unfortunately, the deep sub-micro CMOS process technology is marred by increasing susceptibility to wear out [1]. Widespread reliability challenges are expected in nearest fabrication technologies. Building a fault-tolerant NoC system should be concerned as a necessity. Actually, traditional fault-tolerant algorithms such as using repetitive structures [2–4] are infeasible for the NoC due to area restrictions [5]. Several solutions have been proposed to design a reliable NoC system [6–8], especially for transient or permanent faults on links or routers. These approaches generally provide reliability from four different hierarchies: link control, router control, network interface control and end to end control.

In this article, we mainly address permanent and hard errors on links which result in flit dropping from the hierarchy of end to end control. The error control generally involves three basic steps: detection, containment, and recovery [9]. However, previous works mostly focus only on one step of them. We try to establish an integrated hardware-software framework involving the runtime detection of faulty links, containment of link failure and reconfiguration of healthy routes. Hence, this paper proposes a fault-tolerant MPI-like communication protocol. It detects link failure if there are unresponsive requests and automatically starts the new path exploration.

A key issue to be solved is providing a fault-tolerant routing algorithm, which is discussed in several works [10–13]. A good fault-tolerant routing algorithm means low route latencies and minimal extra consumption. However, most proposed algorithms [10–12] have special restrictions on the number of faulty links as well as their locations.

Watchter et al. [13] have coped with this problem recently. To provide high scalability, they adopted a typical MPI-like protocol for core-to-core communication. The source node detects link failure through unresponsive requests and broadcasts seek packets through the entire mesh NoC to obtain an alternative healthy route. This approach takes advantage of the path redundancy between a pair of nodes and successfully combines high performance of hardware with high flexibility of software.

Although broadcasting seek packets to all other nodes provides complete reachability, it also brings unnecessary packet transmissions. Actually, in most cases, we can find an alternative path within the minimum rectangle defined by source and destination nodes. Based on this observation, we introduce a region flooding algorithm to efficiently search for a fault-free path. It makes use of the NoC's regular structure to direct a search following the minimal path to the destination and dramatically improves network efficiency by limiting the search area.

Generally, a better fault tolerance means a worse performance. Although Wachter et al.'s methodology [13] can perform the best

* Corresponding author.
E-mail addresses: wwanglu1991@gmail.com, 734809187@qq.com (L. Wang), masheng@nudt.edu.cn (S. Ma), zywang@nudt.edu.cn (Z. Wang).

http://dx.doi.org/10.1016/j.micpro.2016.05.004
0141-9331/

fault tolerance, our region flooding approach significantly optimizes the latency with little fault tolerance loss. By analyzing the synthesized influence of fault tolerance and latency, we conclude that our approach is suitable for NoCs, especially for large systems with low fault rates.

Our contributions concentrate on two aspects. First of all, our design arrives at an optimal trade-off between fault tolerance and performance which has not been discussed by previous work [10–17]. Next, our work simultaneously addresses fault detection, containment and recovery.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 presents a fault-tolerant MPI-like communication protocol. Section 4 proposes the design of the NoC architecture and the network interface. Section 5 introduces our approach of searching for a fault-tolerant route and also discusses the deadlock avoidance mechanism and router pipeline. The simulation as well as evaluation are presented in Section 6. After that, we draw a conclusion in Section 7.

2. Related work

Fault-tolerant routing algorithms have been discussed for many years. Previous works mostly focus on adaptive fault-tolerant routing algorithms for mesh networks. Chien and Kim [10] proposed the fault-tolerant planar adaptive routing (PAR) algorithm for n-dimensional meshes. Their algorithms can tolerate rectangle faults with no overlapping of f-rings. Su and Shin [11] proposed an adaptive fault-tolerant routing algorithm for n-dimensional meshes. Their algorithms can tolerate a disconnected rectangular block in an n-dimensional mesh. However, these algorithms can only be used in special topologies or special region shapes such as L, T or +.

Recently, some topology-agnostic fault-tolerant routing algorithms are discussed [14–17]. Dumitras et al. [15] proposed a probabilistic flooding scheme. Costas et al. [16] introduced a deadlock free hybrid routing algorithm, utilizing load-balancing routing on fault-free paths to support high performance and providing pre-reconfigured escape path in the vicinity of faults. Aisopos et al. [17] obtained an alternative path by broadcasting reconfiguration flags upon any number of concurrent network faults in any location. Watcher et al. [13] presented a novel fault-tolerant communication protocol that takes advantage of intrinsic redundancy of the NoC to provide alternative paths between any source-target pair, even in the presence of multiple faults. Different from these algorithms, our approach restrains the search in a rectangular area rather than the entire NoC, which dramatically decreases the latency and improves network efficiency.

There is also some work on designing fault-tolerant message passing libraries as a fault recovery method [18–21]. For instance, Batchu et al. [20] tested unresponsive processes by implementing self-checking threads which use heartbeat messages to monitor the MPI/FT progress. Aulwes and Daniel [21] proposed fault-tolerant mechanisms for the MPI such as the checksum, message retransmission, and automatic message re-routing. The timeout seeking mechanism in our proposed fault-tolerant MPI-like communication has similar idea with these work.

3. Fault-tolerant MPI-like communication protocol

3.1. Basic communication protocol

The basic communication protocol between nodes in this work is message passing. Two MPI-like primitives are adopted: *MPI_Send()* and *MPI_Receive()*. The communication protocol in our approach derives from the non-blocking synchronous communication mode. Fig. 1 illustrates the communication procedure between two tasks. Particularly, the task A and B are mapped to the node A and B respectively.

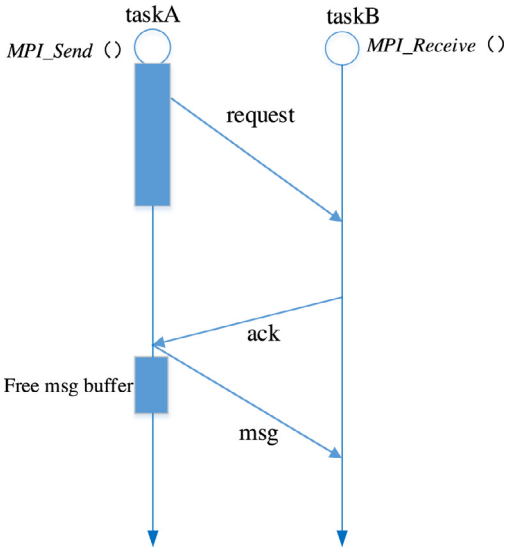

Fig. 1. Basic MPI-like communication protocol between two nodes.

During one point to point communication, task A executes the function *MPI_Send ()* on the source node. First, the node A sends a *request* message through the NoC. Being a control message, the request message has a single flit. After sending a *request* message, some information such as the destination, the tag, the starting address and the size of the data message should be written to '*msg buffer*' which is a dedicated memory space or register stacks in the node A. At the same time, the computation could be carried on in the task A. Only after an acknowledgment from the destination has been received, the node A will free the corresponding '*msg buffer*' and send the data message to the node B.

On the destination node, the task B executes the function *MPI_Receive()* and waits for a request message. After receiving the expected *request* message, it will send an acknowledgment (*ack*) to the source node. Once there exist faulty links in the communication path which cause interruptions for the request or acknowledgement messages, the data message will not be transferred normally and the system will stall. So we improve this communication protocol by adding fault tolerant designs to ensure service qualities.

3.2. Fault tolerant communication protocol

Fig. 2 describes the proposed fault-tolerant communication protocol. It illustrates two fault scenarios, the link failure from A to B when sending a *request* message and link failure from B to A when sending an acknowledgment message. In order to tolerate these faults and provide a reliable service, a timeout seeking mechanism has been added. We suppose that there are faulty links in the original path if the task A cannot receive an acknowledgment from the task B during a scheduled time. Particularly, the scheduled time *d_time* is set according to the average latency in the fault-free circumstances. After that, a seek message will be triggered to find a fault-free path between source and destination nodes. When the node B receives the *seek* message, it will return a *track* message which contains a recording of the new fault-free route. The node A then updates the route table and delivers the data message with the new healthy route. However, new faults may appear in this path. In order to deal with the occasion where new faults appear during the current data transmission, the node B should send a seek packet to the node A if it does not receive expected packets