PERFORMANCE EVALUATION OF FAULT TOLERANT METHODOLOGIES FOR NETWORK ON CHIP ARCHITECTURE

By

HAIBO ZHU

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

WASHINGTON STATE UNIVERSITY

School of Electrical Engineering and Computer Science

August 2007

To the Faculty of Washington State University:

The members of the committee appointed to examine the thesis of HAIBO ZHU find it satisfactory and recommend that it be accepted.

_____

Chair

_____

_____

ACKNOWLEDGEMENT

I would like to thank my advisor Dr. Partha Pratim Pande for having guided me through this challenging problem. I would cherish in my memory the lively discussions I had with him during my research. It was a great experience working with him.

I would also like to thank my colleagues Mr. Amlan Ganguly, Mr. Brett Feero, and Mr. Souradip Sarkar for their frequent help and discussion during my research. They are always willing to help me to fully understand the problems and then to solve them.

My parents, Mr. Longyuan Zhu and Mrs. Hedi Mei have always given me the courage to pursue the higher knowledge. Without their support none of this work would have been possible.

Last but most importantly I thank my wife Yan for her understanding and encouragement. Without her I could not have proceeded so far. Her unflinching faith in me and curiosity about my work made my research experience even more rewarding.

Performance Evaluation of Fault Tolerant Methodologies for Network on Chip Architecture

Abstract

By Haibo Zhu, M.S.
Washington State University
August 2007

Chair: Partha Pratim Pande

Current SoC designs are appearing with very large numbers of embedded processors. From consumer multimedia to image processing to defense applications, new designs are coming out with very high numbers of embedded processors. The communication requirements of these large MP-SoCs are convened by the emerging network-on-a-chip (NoC) paradigm. In the deep sub-micron (DSM) VLSI processes, it is difficult to guarantee correct fabrication with an acceptable system performance and chip yield without employing design techniques that take into account the intrinsic existence of manufacturing faults. To become a viable alternative IC design methodology the NoC paradigm must address the system-level reliability issues, which is going to be the dominant concern in the DSM and beyond silicon era. By incorporating fault tolerant methodologies in the data communication mechanism it is possible to tolerate permanent manufacturing faults in the NoC interconnect architectures. Performance of two different NoC architectures, namely Mesh and Butterfly Fat Tree (BFT) are explored by incorporating the partially adaptive routing algorithms and spare hardware block respectively. The performance tradeoffs associated with fault tolerant schemes in NoC fabrics, like network

throughput, latency, silicon area overhead and power consumption are explored. With the help of fault tolerant mechanisms, the chip yield can be improved because of higher sustained throughput in presence of faults.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1. System-on-Chip Design Background

The idea of integrating numerous components of a computer system into a single chip has led to the miniaturization of many portable devices and an increase in their computational capabilities. The possibility of this higher degree of integration has led to the concept of System on Chip (SoC). Current SoC designs are appearing with multiple embedded processors, called multiprocessor SoC or MP-SoC. The number of embedded processors is ranging between 8 to 32 in communications and network processing, security processors, storage array networks, and wireless base stations; to over 100 processors in recent platforms in consumer image processing, and high-end network processors. As the complexity of the MP-SoCs increases, communication among the constituent Intellectual Property (IP) blocks becomes the main challenge. Initially, shared-bus architecture was introduced into SoC domain to improve the efficiency of data transferring. However, standing from the point of Modular, Flexible, and Scalable Architecture Model (MFSAM), a shared-bus based system is not suitable for MP-SoC simply because of the long delay in data transfer and excessive energy dissipation when more processors/IPs are added into the system. Therefore, designing high performance interconnection networks to integrate multiple IP blocks in a single die becomes a critical issue. In this context, Network-on-Chip (NoC) [1] is regarded as one of promising solutions to achieve high degree of integration in a single SoC.

1.2. The Network-on-Chip Paradigm

The Network-on-Chip (NoC) design paradigm is viewed as an enabling solution [1] [2] for the integration of exceedingly high number of computational and storage blocks in a single chip. The common characteristic of NoC interconnect architectures is that the functional blocks communicate with each other with the help of intelligent switches and links.

Wormhole switching [3] [4] is the most commonly used data transmission mechanism adopted for NoCs. In wormhole switching, the packets are divided into fixed length flow control units (flits) and the input and output buffers are expected to store only a few flits. As a result, the buffer space requirement in the switches can be small compared to that generally required for packet switching. Thus, using a wormhole switching technique, the switches will be small and compact. The first flit, i.e., header flit, of a packet contains routing information. Header flit decoding enables the switches to establish the path and subsequent flits simply follow this path in a pipelined fashion. As a result, each incoming data flit of a message packet is simply forwarded along the same output channel as the preceding data flit and no packet reordering is required at destinations. If a certain flit faces a busy channel, subsequent flits also have to wait at their current locations. One drawback of this simple wormhole switching method is that the transmission of distinct messages cannot be interleaved or multiplexed over a physical channel. Messages must cross the channel in their entirety before the channel can be used by another message. This will decrease channel utilization if a flit from a given packet is blocked in a buffer. By introducing virtual channels [5] in the input and output ports, we can increase channel utility considerably. If a flit belonging to a

particular packet is blocked in one of the virtual channels, then flits of alternate packets can use the other virtual channel buffers and, ultimately, the physical channel. The canonical architecture of a switch having virtual channels is shown in Figure 1.1.

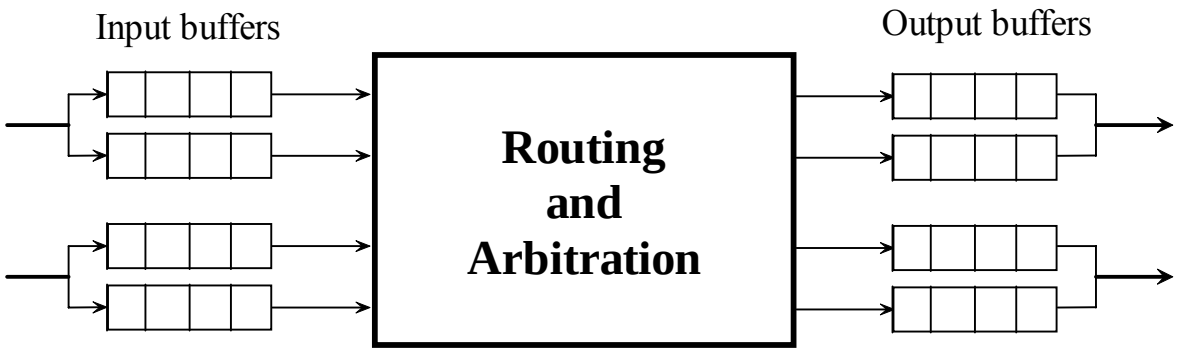

Figure 1.1: Virtual-channel switch

1.3. NoC Topologies

A few NoC interconnect architectures have been proposed by different research groups. The characteristic of several well known NoC topologies are discussed in the following subsections.

1.3.1. Mesh

A Mesh interconnect architecture, first proposed in [6], is so-called Chip-Level Integration of Communicating Heterogeneous Elements or CLICHÉ. In a Mesh network, it consists of $m \times n$ mesh of switches interconnecting processing node (IPs) placed along with the switches. Each switch is connected to four neighboring switches and one IP block except those on edges. Consequently the number of switches equals the number of IPs. The architecture of a Mesh-based NoC consisting of 16 functional IP blocks is shown in Figure 1.2.

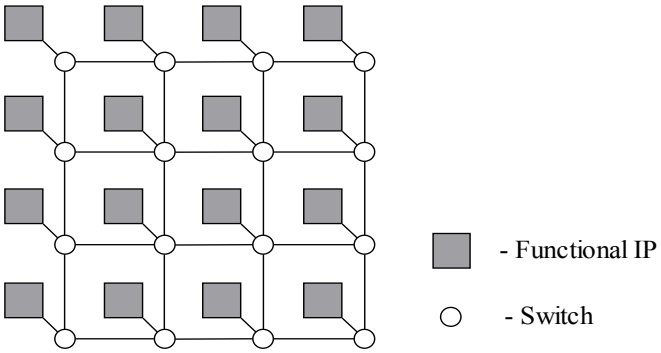

Figure 1.2: Mesh network

1.3.2. Folded Torus

In [5], the authors proposed a 2-D Torus as a NoC architecture. The Torus architecture is basically the same as the Mesh; the only difference is that the switches at the edge are connected to the switches at the opposite edge through a wrap-around channel. Each switch has five ports, one connected to the local functional IP and the others connected to the closest neighboring switches. The long end-around connections can yield excessive delays (Figure 1.3 (a)). However, this can be avoided by folding the Torus as shown in Figure 1.3 (b). This alternative Torus is called Folded Torus.

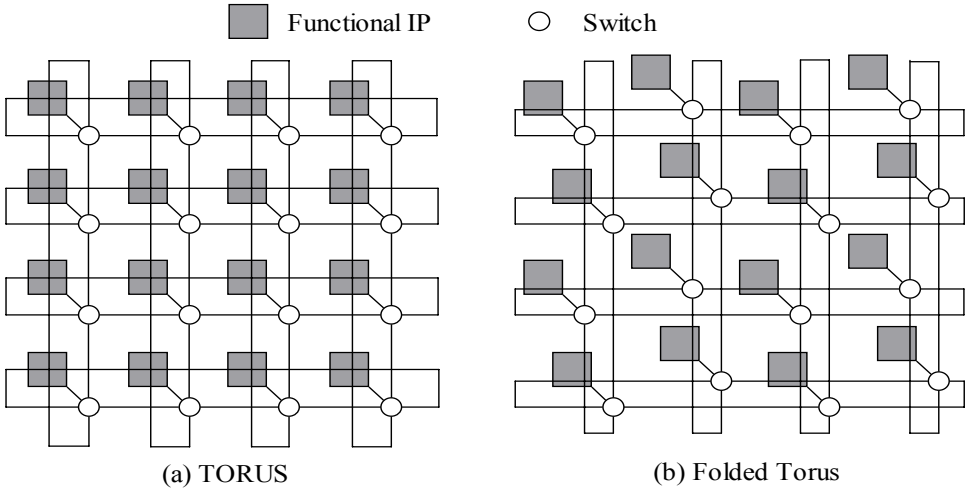

Figure 1.3: Torus and folded Torus

1.3.3. SPIN

Guerrier and Greiner have proposed a generic template call SPIN [7] (Scalable, Programmable, Integrated Network) for on-chip packet switched interconnections. In SPIN architecture shown in Figure 1.4, a fat tree topology is used to integrate functional IPs, where the switches reside at the vertices and functional IP reside at the leaves. Each node has four children and parent is replicated four times at any level of the tree. As a consequence, the network size grows as $(N \log N)/8$ and the number of switches converges to $\frac{3N}{4}$ for large N which denotes the number of functional IP blocks integrated.

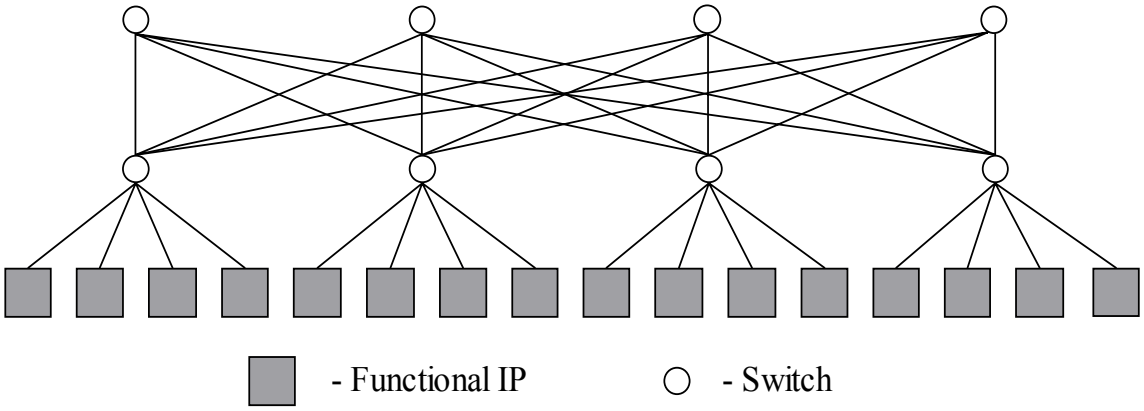

Figure 1.4: SPIN

1.3.4. BFT

The Butterfly Fat Tree (BFT) is proposed in [8], which is shown in Figure 1.5 for the case of 16 IP blocks. In this architecture, the IPs are placed on the leaves and switches are placed at the vertices. A pair of coordinates is used to label each node, *(l, p)*, where l denotes a node's level and p denotes its position within that level. In general, at the lowest level, there are N functional IPs with addresses ranging from 0 to *(N-1)*. The pair *(0, N-1)* denotes the locations

of IPs at that lowest level. Each switch, denoted by *S(l, p)*, has four child ports and two parent ports. The IPs are connected to *N/4* switches at the first level. In the j^{th} level of the tree, there are $N/2^{j+1}$ switches. The number of switches in the BFT architecture converges to a constant independent of the number of levels, *N/2* as proved in [8].

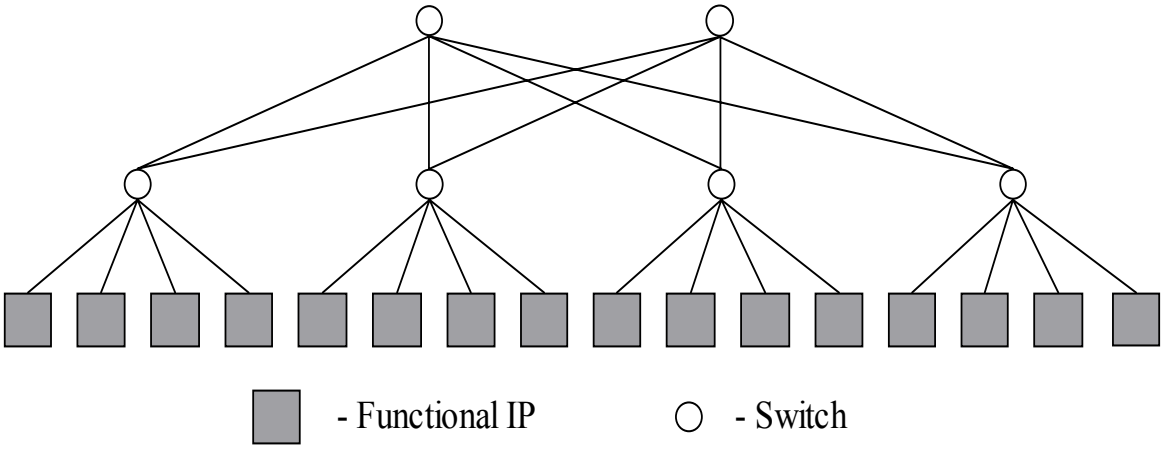

Figure 1.5: BFT network

1.4. Fault Tolerance in NoC

For deep sub-micron (DSM) VLSI processes, it is difficult to guarantee correct fabrication with an acceptable yield without employing design techniques that take into account existence of manufacturing defects [9]. Moreover, the life-time reliability of DSM devices is likely to be compromised by effects such as electromigration and material ageing [10]. In order to improve the reliability of MP-SoCs, their interconnect infrastructures must be designed such that fabrication and life-time faults can be tolerated. These irrecoverable faults influence the behavior of NoC fabrics and consequently degrade the system performance. NoCs can be designed to handle negative effects of permanent faults by adopting different fault tolerant design methodologies. These mechanisms depend either on modification of data

routing schemes or the overall architecture.

In the NoC environment initially deterministic routing mechanism was employed due to the ease of implementation. The primary limitation of deterministic routing is that it establishes a fixed path between a pair of source and destination nodes. Consequently it demonstrates poor performance in presence of faults situated on the routing path as it fails to establish alternate routes. A certain level of performance can be maintained in presence of faulty components if fault tolerant mechanism such as adaptive routing algorithm is adopted. One of the principal characteristics of fault tolerant mechanism is the ability to establish alternate routing paths in presence of faults.

In a tree based NoC, adaptive routing schemes are unable to maintain the system performance because of the change of architecture due to the broken links/nodes. As a consequence, tree based NoCs should be designed in which redundant links and spare nodes are added to the basic tree structure, so that connectivity is maintained in the presence of a certain number of faults.

1.5. Yield Enhancement

The yield of an integrated circuit is the fraction of IC chips that meet a specific set of functional requirements out of the total number of chips manufactured [11]. In the DSM era, yield-loss can be caused by any imperfections in the fabrication process either introduced during the processing through physical steps, or through the chip design and manufacturing, i.e., manufacturing defects. Moreover, the life-time reliability of DSM devices is likely to be compromised by effects such as electromigration and material ageing which consequently

reduces the yield. Continuous advances in manufacturing technology have reduced the defect density in a chip. But with scaling and increase in the chip area the overall defect density increases and this causes the reduced fabrication yield [12]. Thus, the development and use of yield enhancement techniques at the design stage is justifiable.

Yield enhancement techniques are aimed at making the integrated circuit fault tolerant, i.e., less sensitive to manufacturing defects or permanent faults at the design stage [12]. These techniques include incorporating redundancy into the design, modifying the floorplan, and incorporating fault tolerant mechanisms. By incorporating fault tolerant design techniques in NoC design flow a certain level of performance can be maintained even in the presence of faults. Consequently it will help to enhance the yield of the fabricated chip.

1.6. Contributions

The principal contributions of this work are as follows:

1. Evaluating performance of partially adaptive routing methodologies to achieve fault tolerance in Mesh-based NoC architectures.

2. Establishing performance benchmark for partially adaptive routing methods compared to stochastic algorithms, like random walk.

3. Design of a fault tolerant Butterfly Fat Tree (BFT) based NoC architecture and its performance evaluation

4. Quantifying enhancement of yield for NoC-based SoC by adopting fault tolerant design methodologies.

1.7. Thesis Organization

The thesis is organized in six chapters. The 1st chapter introduces the research problem as well as the background. Related work is presented in the 2nd chapter. In the third chapter, the fault tolerance in Mesh-based NoC is introduced, followed by the system performance evaluation and comparison, which are made by incorporating all the fault tolerant mechanisms onto a Mesh-based NoC. In fourth chapter, the fault tolerance in BFT based NoC is presented and the system performance is characterized. The chip yield enhancement by incorporating fault tolerant methodologies is explored in chapter five. Finally the last chapter summarizes the important conclusions and points out the direction of future research.

CHAPTER 2

RELATED WORK

With technology scaling, fault tolerance of the communication infrastructure is becoming a key challenge for designing NoCs. Though NoC research has gained significant momentum, the aspect of fault tolerance is not addressed adequately. Initial NoC research primarily concentrated on deterministic routing algorithms, but to make the system fault tolerant, we need to adopt more complex routing mechanisms.

In [13], stochastic communication paradigm is proposed to achieve fault tolerance in NoC architectures. The IPs communicate using a probabilistic broadcast: data packet is forwarded to a randomly chosen neighboring router until the entire network becomes aware of it. Even though this approach spreads the packet with an exponentially fast broadcast speed, it requires significant energy consumption in order to achieve a higher system performance by increasing the probability of redundant transmission. Furthermore, a packet keeps propagating to the rest area of the network even if it reaches the destination unless the parameter, Time-To-Live (TTL) goes to zero. As a consequence, valuable network resources are kept busy in sending a successfully received message repeatedly instead of useful new information. In [14], the authors inherited the randomized gossip protocol from [13] to address fault tolerance in NoCs. Performance of different stochastic routing algorithms, viz. directed flooding, probabilistic flooding and random walk were investigated in NoC scenario. The principal limitation of these algorithms is that they can only sustain a very low traffic injection rate. This

arises due to the fact that multiple copies of a single message are injected into the network to improve successful data arrival rate. Design of a low latency router supporting adaptivity for on-chip interconnects is described in [15]. But the authors have not quantified the performance of the NoC with increasing number of faults.

The turn model is a well known partially adaptive routing algorithm widely investigated for multi-processor SoC environments [16] [17]. West-first routing, north-last routing and negative-first routing are three basic types of turn models. Compared to fully adaptive routing algorithms, turn model algorithm is a partially adaptive algorithm because two turns out of eight are forbidden in order to avoid deadlock. In [18], the authors combined deterministic x-y routing and adaptive routing in a single router. The routing scheme switches from deterministic to adaptive routing depending on the network congestion. The partially adaptive routing algorithm adopted in [18] is the odd-even turn model [19]. The odd-even turn model prohibits some types of turns based on the locations of nodes in order to make itself deadlock free. More specifically, a packet is not allowed to make east-to-north or east-to-south turns at nodes located on even columns, and north-to-west or south-to-west turns at nodes located on odd columns [19]. The performance evaluation in [19] shows that the negative-first and odd-even turn models have very competitive performance depending on the traffic scenario. The above algorithms can be applied to achieve fault tolerance in Mesh-based NoCs.

Guerrier and Greiner have proposed a generic interconnect template called SPIN (Scalable, Programmable, Integrated Network) for on-chip packet switched interconnections, where a fat-tree architecture is used to integrate IP blocks. Based on SPIN, the Butterfly Fat Tree (BFT) topology for NoC is introduced in [8]. The authors describe an interconnect

architecture based on the BFT topology for a networked SoC, as well as the associated design of the required switches and addressing mechanisms. In [20], the authors developed a general k-fault-tolerant tree based structure for multiprocessor architecture network where spare nodes and spare links are employed. The authors introduced an efficient reconfiguration by using shared links when faults occur. According to the paper, the processing node is able to switch to a fault free shared link, aiming at creating a new network, in order to get rid of obstruction of a faulty link. The number of shared links and speed of reconfiguration are optimized in this paper. Izadi and Özgüner present a real-time fault-tolerant design for an l-level k-ary tree multiprocessor network in [21]. The authors suggested to cluster neighboring processing nodes and assigned each cluster a spare node connecting to every regular node in this cluster. Beyond this, spare nodes of neighboring clusters are also suggested and they are connected using inter-cluster spare links. The spared nodes inside/outside clusters are capable to replace the faulty nodes by reconfiguring the tree based network when a fault is found in the network.

In this work, our aim is to evaluate the performance of partially adaptive routing algorithms, such as the negative-first routing algorithm and odd-even turn model algorithm compared to a stochastic method like random walk and deterministic routing algorithm such as dimension order routing or x-y routing in presence of permanent faults when applied to Mesh-based NoC architectures. The system performance characteristics including network throughput, latency, silicon area overhear, and power dissipation are considered in this work. We also demonstrate how a tree-based NoC, i.e. Butterfly Fat Tree (BFT), can be made tolerant to permanent faults by modifying the interconnect architecture. All these fault tolerant design methodologies will significantly improve the yield of the system as they improve the

system's reliability when faults occur. We quantify the yield improvement of different NoC topologies in presence of permanent faults.

CHAPTER 3

FAULT TOLERANCE IN MESH-BASED NOC

The common characteristic of NoC architectures is that the constituent IP cores communicate with each other through intelligent switches. Generally wormhole switching is adopted [22]. As shown in Figure 3.1, one of the widely known NoC topologies, the Mesh based network architecture, is used in which each switch is connected to four neighboring switches and one IP block except those on edges. We analyze the performance of a Mesh-based NoC in presence of permanent faults when different deterministic/adaptive and stochastic routing algorithms are adopted in this chapter.

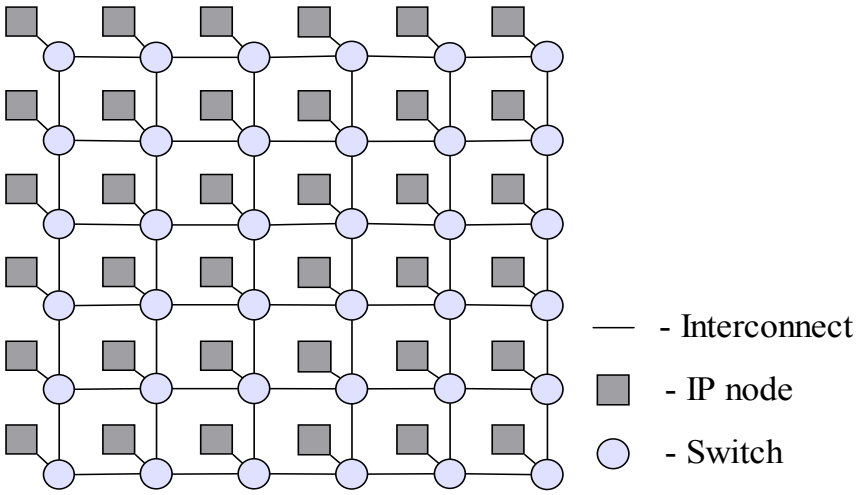

Figure 3.1: Mesh network

3.1. Routing Algorithms

Routing algorithms establish the path followed by each message or packet. Routing algorithms can be implemented in different ways. The routing algorithm can be either

deterministic or adaptive. Deterministic routing algorithms always supply the same path between a given source and destination pair. Adaptive routing algorithms use information about network traffic and/or link status to avoid congested or faulty regions of the network [2]. In addition to the deterministic and adaptive routing algorithms, another type of routing algorithm is on-chip stochastic communication, which spreads more than one copy of the message through the entire NoC, assuming at least one will ultimately reach the destination.

Fault tolerance is the ability of the network to function in the presence of component failure [2]. A fault tolerant algorithm distinguishes from deterministic one according to the fact that it can provide an alternative path or make a redundant copy so that the message wouldn't be blocked by a faulty component. On the other hand, if deterministic routing algorithm is adopted in a NoC environment, the effects of faults are rapidly propagated through the switches to other messages that compete for input/output buffers since block message span multiple switch nodes. Thus, fault tolerant algorithms not only bring the message to destination successfully in presence of faults, but also make rooms for other competing messages waiting for the occupied buffer space.

In this chapter, performance of different routing algorithms, in presence of permanent faults when applied to a Mesh-based NoC architecture is explored.

3.1.1. X-Y Routing

The x-y algorithm is a simple deterministic routing methodology used in Mesh networks [22]. The basic idea of this x-y routing algorithm is that it routes data packets by crossing dimensions in strictly increasing or decreasing order, reducing the offset to zero in one

dimension before routing in the next one. This algorithm for a 2-D Mesh network is described in the following:

Algorithm: X-Y Routing for 2-D Mesh Network

Inputs: Coordinates of current node (*Xcurrent*, *Ycurrent*) and destination node (*Xdest*, *Ydest*)

Output: Selected output channel

Procedure:

Xoffset := *Xdest* − *Xcurrent*;

Yoffset :=*Ydest* −*Ycurrent*;

if *Xoffset* < 0 **then**

Channel := X−;

endif

if *Xoffset* > 0 **then**

Channel := X+;

endif

if *Xoffset* = 0 **and** *Yoffset* < 0 **then**

Channel := Y−;

endif

if *Xoffset* = 0 **and** *Yoffset* > 0 **then**

Channel := Y+;

endif

if *Xoffset* = 0 **and** *Yoffset* = 0 **then**

Channel := *Internal*;

endif

Though the x-y routing algorithm is easy to implement and has low overhead, it cannot maintain the desired level of performance in presence of faults. On the other hand fully adaptive algorithm, such as adaptive fault-tolerant wormhole routing algorithm [23] [24] [25],

perform better in presence of faults, but due to complexity in implementation has higher overhead in terms of silicon area and energy consumption. Consequently we investigate the applicability of partially adaptive algorithms to achieve a certain degree of fault tolerance in NoC communication fabrics. Turn models and odd-even routing are well established partially adaptive routing algorithms used in parallel computing domain [16] [17] [19]. Additionally, random walk is proposed as a suitable stochastic routing algorithm for NoC architectures [14].

3.1.2. Negative-First Algorithm

The turn models [16] [17] can be used to develop partially adaptive routing algorithms for Mesh and Torus networks. In a 2-D Mesh network, there are four directions, eight 90-degree turns, and two abstract cycles of four turns as shown in Figure 3.2. Turn model routing algorithm prohibits one turn from each cycle to prevent deadlock. According to the prohibited turns, the turn model routing algorithm is classified into three basic types of adaptive algorithms, namely, west-first, north-last and negative-first. West-first routing algorithm routes a packet in the west direction first, if necessary, and then adaptively south, east, and north. North-last routing algorithm routes a packet first adaptively west, south, and east, and then north. Negative-first routing algorithm routes a packet first adaptively west and south (negative x or y according to coordinates), and then adaptively east and north (positive x or y). These turn models can be applied to handle switch or link failures in a NoC. Algorithms examples of west-first and north-last are shown in the following Figure 3.3 (a) and (b).

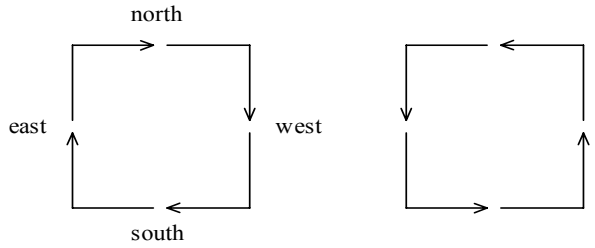

Figure 3.2: Possible turns in a 2-D Mesh network

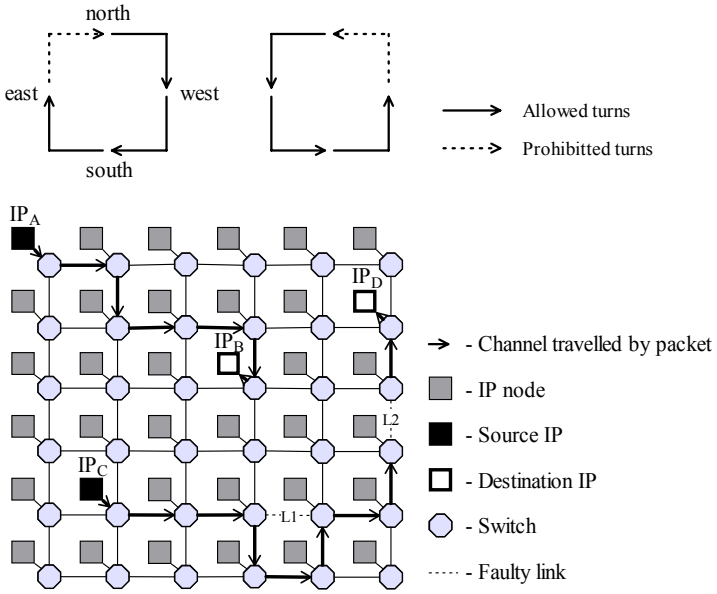

Figure 3.3 (a): Routing sample for west-first algorithm

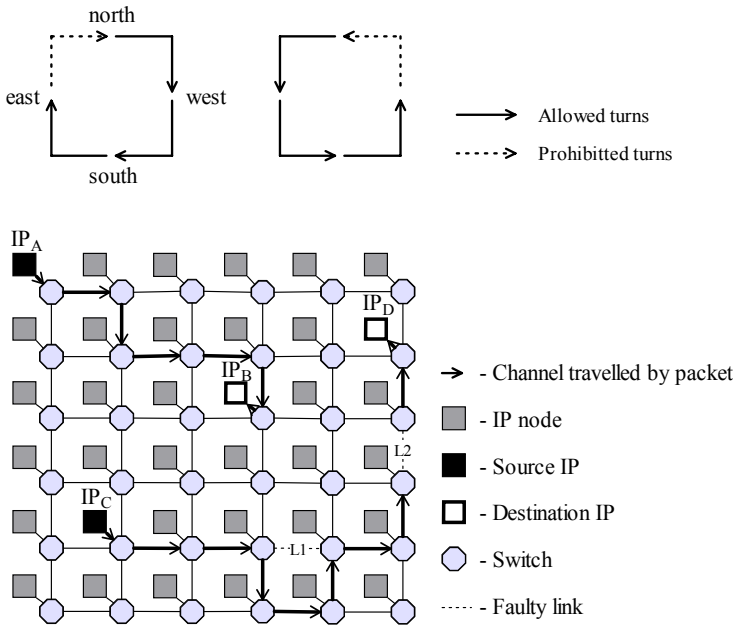

Figure 3.3 (b): Routing sample for north-last algorithm

The negative-first algorithm is one of the turn models used in 2-D Meshes, in which a packet is routed in the negative direction in each dimensions in the first phase, and then it is routed in the positive direction in the second phase [16]. Specifically, the forwarded message first finishes offsets directing to west or north, then turns to east or south. The fault-tolerant version of this negative-first algorithm routes adaptively in the negative direction, even further west or south than the destination. The effectiveness of this routing methodology in presence of a fault in the NoC interconnect architecture is explained with the help of Figure 3.4 (a) and (b).

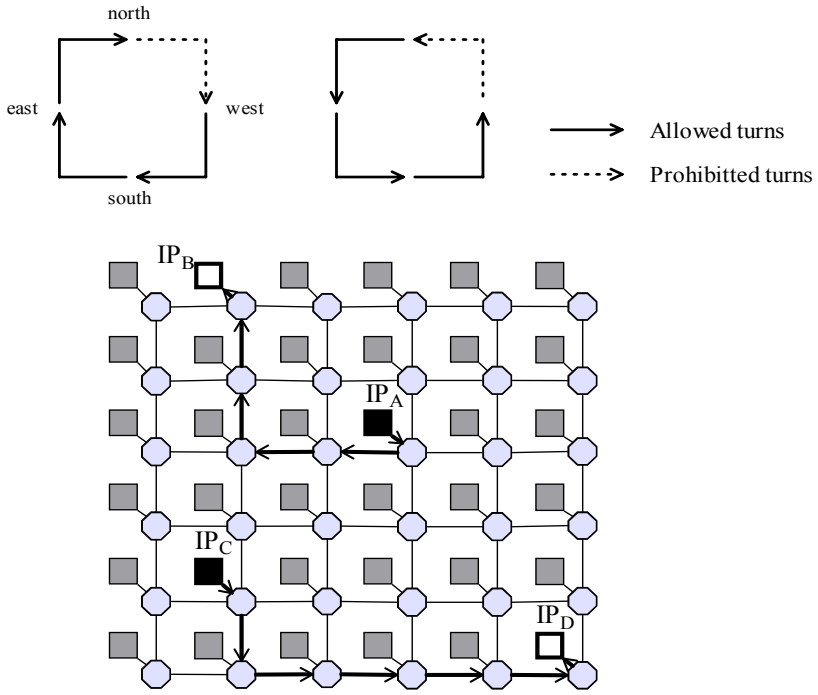

Figure 3.4 (a): Negative-first routing in a Mesh-based NoC fault-free case

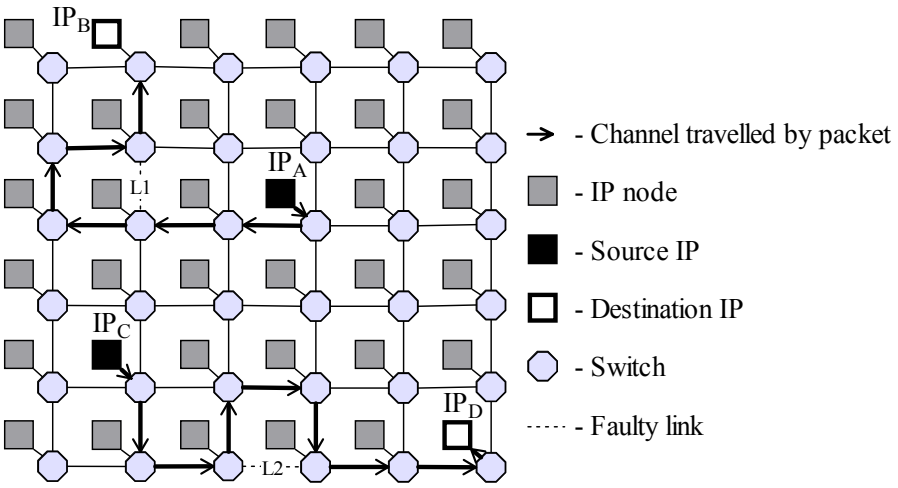

Figure 3.4 (b): Negative-first routing in a Mesh-based NoC – switches S_1, S_2 are faulty

In Figure 3.4, source IP_A is trying to send a packet to destination IP_B. Two situations are presented in Figure 3.4 (a) and (b) respectively. When the network is fault-free, the packet is first routed in x direction before being routed to y direction. When one link L_1 is faulty in the path, if normal deterministic x-y routing [16] was adopted, there is no path for the packet to move towards the destination. For negative-first routing, the packet advances further along the negative direction and then turns back, therefore avoiding the faulty link.

If negative-first routing is adopted then as indicated in Figure 3.4, the packet is routed adaptively even further south and west than the destination to avoid the faulty switch, and then turn back to north or east to reach the destination. The exception occurs when a packet being routed along the edge of the Mesh in the negative direction encounters a faulty switch. As shown in Figure 3.4 (b), suppose source IP_C is trying to communicate to the destination IP_D, and the link L_2 is faulty. In this case the packet is routed one hop perpendicular to the edge, then one hop toward the destination, and one hop back to the edge. The fault-tolerant routing algorithm resulting from the modification of the negative-first routing algorithm is summarized as follows.

Let the coordinates of the current node be $(X_{current}, Y_{current})$ and the coordinates of the destination node be (X_{dest}, Y_{dest}). The distance between the current node and the destination node is expressed as $X_{offset} = X_{dest} - X_{current}$ and $Y_{offset} = Y_{dest} - Y_{current}$

1. If either $X_{offset} < 0$ or $Y_{offset} < 0$ then route the packet west and south to the destination or further west and south than the destination, avoiding routing the packet to a negative edge for as long as possible. If a faulty node on a negative edge blocks the path along the edge, route the packet one hop perpendicular to the edge.

2. If both X_{offset} and Y_{offset} are greater than zero then route the packet east and north to the destination, avoiding routing the packet as far east or north as the destination for as long as possible. If a faulty node on a negative edge of the Mesh blocks the path to a destination on the edge, route the packet one hop perpendicular to the edge, two hops toward the destination, and one hop back to the edge.

The principal advantage of the negative-first routing algorithm is that by allowing the packets to be routed further west and south than the destination, more paths to the destination are created. This increases the probability that the packets can be routed around a faulty switch or link.

3.1.3. Odd-even Turn Model Algorithm

Odd-even turn model was first proposed by Chiu [19] and is an extension of Glass and Ni's turn model [16]. The odd-even turn model facilitates deadlock-free routing in Mesh network of a NoC [27] [28]. In a two dimensional Mesh of size $m \times n$ every node is identified by a two element vector (x, y), $0 \leq x \leq m-1$, and $0 \leq y \leq n-1$, where x and y are the coordinates in

the two dimensions. The nodes having the same *x* dimension belong to the same column and those having the same *y* dimension constitute the same row. A row channel and a column channel refer to channels in the *x*-dimension and *y*-dimension respectively. A turn is the common point where the tail node of either the row or the column channels meet and the particular node (at which they meet) are referred to as the *turning node*.

Deadlocks in routing usually occur as a result of packets waiting to for each other to form a cycle. Many of the routing algorithms prohibit deadlock by avoiding certain turns, whereas the *odd-even* routing is based on restricting the locations at which certain turns can be taken so that cyclic dependency never occurs and hence deadlock is avoided. This model allows all types of turns.

Definition 1: In a 2-Dimensional Mesh network, a particular column is called an even (or odd) column if the x-coordinate of the column is even (or odd).

The odd-even turn model is based on the following routing rules:

Rule 1: *East-north* and *north-west* turns are not allowed at any nodes located in the even column and odd column respectively.

Rule 2: *East-south* and *south-west* turns are not allowed at any nodes located in the even column and odd column respectively.

The odd and even columns defined above can also be interchanged. Deadlock freedom is achieved as the rightmost column of the waiting path cannot result, if the rules are followed. The detail of odd-even turn model is explained with the help of Figure 3.5. In the figure, the odd-even routing algorithm has been illustrated. First, we have considered a case when there is no fault in the network. Then we have considered a network with faults and again show how the

data can be routed in the latter case.

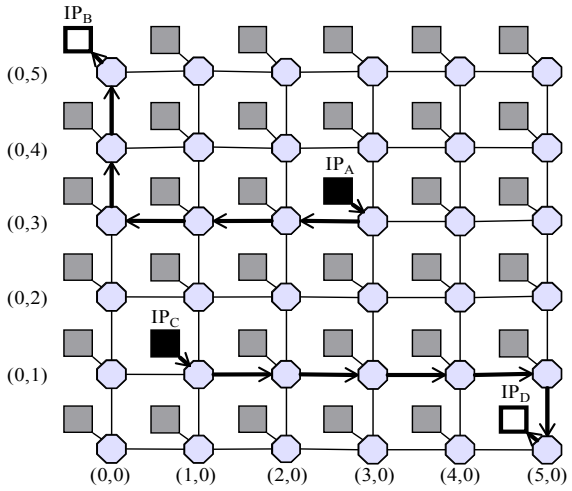

Figure 3.5 (a): A sample for Odd-even turn algorithm - fault-free case

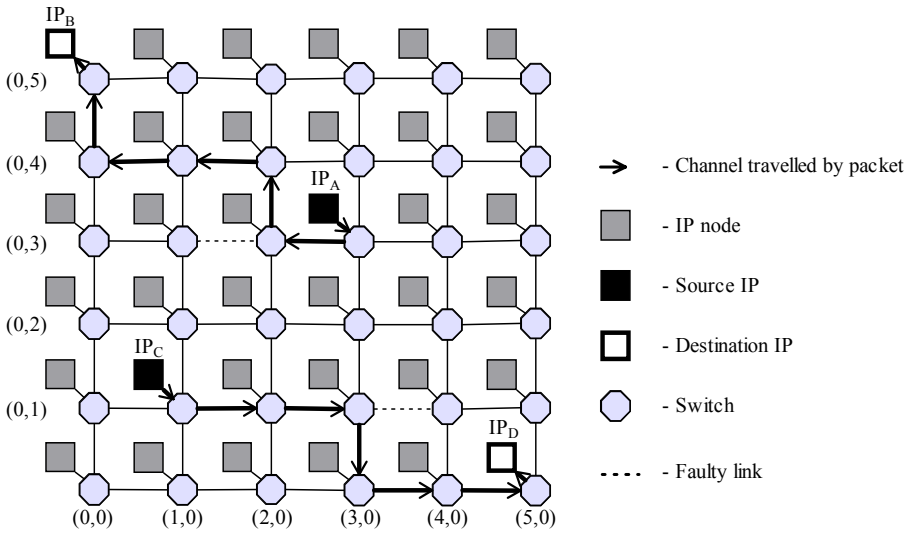

Figure 3.5 (b): A sample for Odd-even turn algorithm - in presents of faults

3.1.4. N Random Walk

In [14], the authors introduced three different fault tolerant algorithms, viz. probabilistic flooding, directed flooding and N random walk. In this work it is shown that N random walk can provide a better performance than the two flooding algorithms. N random walk allows injection of a fixed number of copies (N) of a message into the network. By

using random walk, each node forwards the messages to one of its outgoing channels, and meanwhile makes them follow non-deterministic paths to destination. The selection of outgoing channel is determined by a set of random probabilities P_N, P_S, P_W and P_E, where sum of all probabilities is 1. The probabilities are calculated as follows: first, the Manhattan distance between the destination and the current node as well as its neighboring nodes is calculated. The Manhattan Distance for current IP node is given by (3.1).

$$D_c = \left|X_{destination} - X_{current}\right| + \left|Y_{destination} - Y_{current}\right| \tag{3.1}$$

A multiplicative factor M_x is set to 1 for any direction where D_x (x denotes either the current or any neighboring node) is greater than D_c. For the remaining nodes (where $D_x \leq D_c$), the multiplicative factor M_x is equal to min (D_x, 4). Then the multiplicative factors are normalized to obtain the probabilities P_N, P_S, P_W and P_E. After that, a random number is generated in order to choose one outgoing channel based on the probabilities computed in the previous step. Through the procedure described above, each message is likely to be forwarded towards the same destination, but will take a different path to reach destination [14].

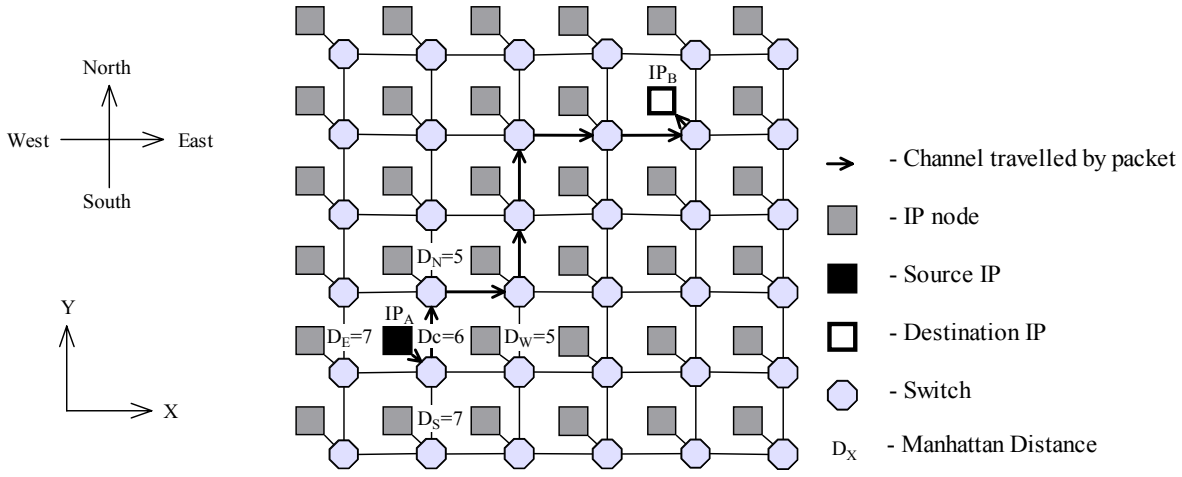

Figure 3.6: N=1 random walk

A Mesh-based network as well as the coordinates is shown in Figure 3.6. The same coordinate is used in the following chapters. Suppose, source IP_A wants to communicate with the destination IP_B. First, the Manhattan distances D_X of IP_A and its neighboring nodes with respect to IP_B are computed, as shown in Figure 3.6. Based on D_X, the multiplicative factors are set to 1 for D_E and D_S, and $\min(D_x, 4)$ for D_N and D_W. Then multiplicative factors are normalized to create the probabilities P_N, P_S, P_W and P_E. In this case, the IP_A delivers the message in north direction as the first step because both north and west directions have larger probability than the other directions. Eventually, the message follows the paths shown in Figure 3.6 from source to destination.

3.2. Experimental Results

We evaluate the performance of the routing algorithms discussed above when applied to a Mesh-based NoC. We characterize the performance of the NoC under consideration in terms of network throughput, energy dissipation, latency and silicon area overhead in presence of permanent faults.

To study the system performance characteristics mentioned above, we consider a system consisting of 64 IP blocks mapped onto Mesh-based NoC architecture as shown in Figure 3.1. Messages were injected with a uniform traffic pattern (in each cycle, all IP cores can generate messages with the same probability). Faults are generated randomly in the network. Faults are manifested by making a particular inter-switch link or node unavailable permanently for data routing. Different routing algorithms described above are applied on the Mesh network. Simulations were performed using 90nm standard cell library. The parameters

used for the purpose of simulations are listed in Table 3.1.

Table 3.1: Simulation parameters

Architecture	Message Length (Flits)	Buffer Depth (Flits)	Number of ports
MESH	16	2	5

3.2.1. Network Throughput

To evaluate the capability of fault tolerance for each routing scheme, we initially considered a very low injection load and measured the achievable throughput in presence of 1% fault rate by incorporating x-y, negative-first, odd-even turn model and 1-random walk algorithms onto a Mesh-based NoC. Figure 3.7 shows the throughput characteristics of the NoC by varying the injection load only up to 0.3%.

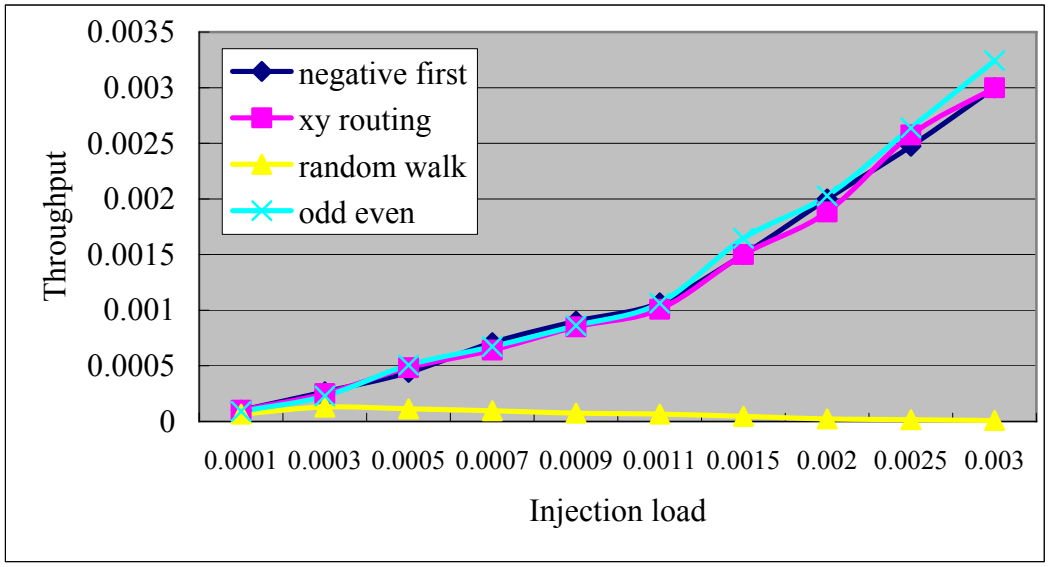

Figure 3.7: Throughput profile when varying injection load

It is evident that for a Mesh-based network, x-y routing, odd-even turn model and negative-first algorithms show an increasing trend, while for 1-random walk throughput has a decreasing trend to zero. This brings out the limitation of N-random walk algorithm. With N=1,

it cannot even sustain an injection load as low as 0.3% with a 1% fault rate. On the contrary, Figure 3.8 shows the performance of the NoC with a 5% fault rate by incorporating x-y routing, odd-even turn model and negative-first algorithm on a Mesh based network. It is evident that both odd-even turn model and negative-first outperforms the x-y routing algorithm. It is worth noting that all of these algorithms are able to sustain a much better throughput profile than the random walk.

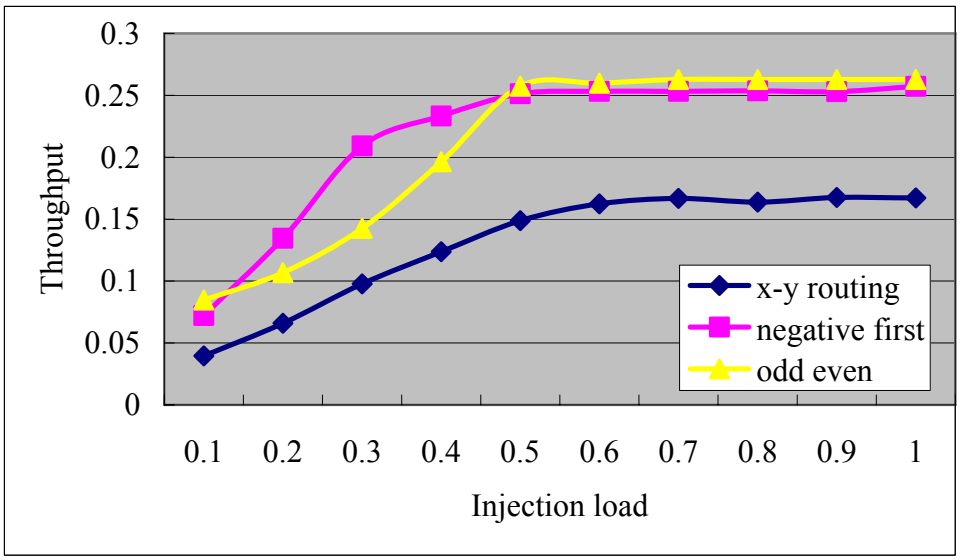

Figure 3.8: Throughput comparison for various routing algorithms with 5% fault rate

We also compared the performance of the N random walk with all the other algorithms in terms of the successful data arrival rate as a relevant metric as suggested in [14]. We considered very low injection load of 0.05%. As shown in Figure 3.9, in a Mesh-based network, negative-first can provide a better performance in presence of faults than odd-even turn model, x-y routing and N random walk (for $N < 64$). It has almost identical successful data arrival rate in presence of 5% fault rate as the random walk when N=64. Odd-even turn model is comparable with N=16 random walk and outperforms x-y routing.

We showed that the negative-first routing algorithm outperforms the odd-even turn

model, x-y routing and N random walk (when N<64) in presence of faults in terms of successful data arrival rate. For a more complete characterization, in the next sub-section we explore the other performance metrics, such as energy dissipation, latency and silicon area overhead.

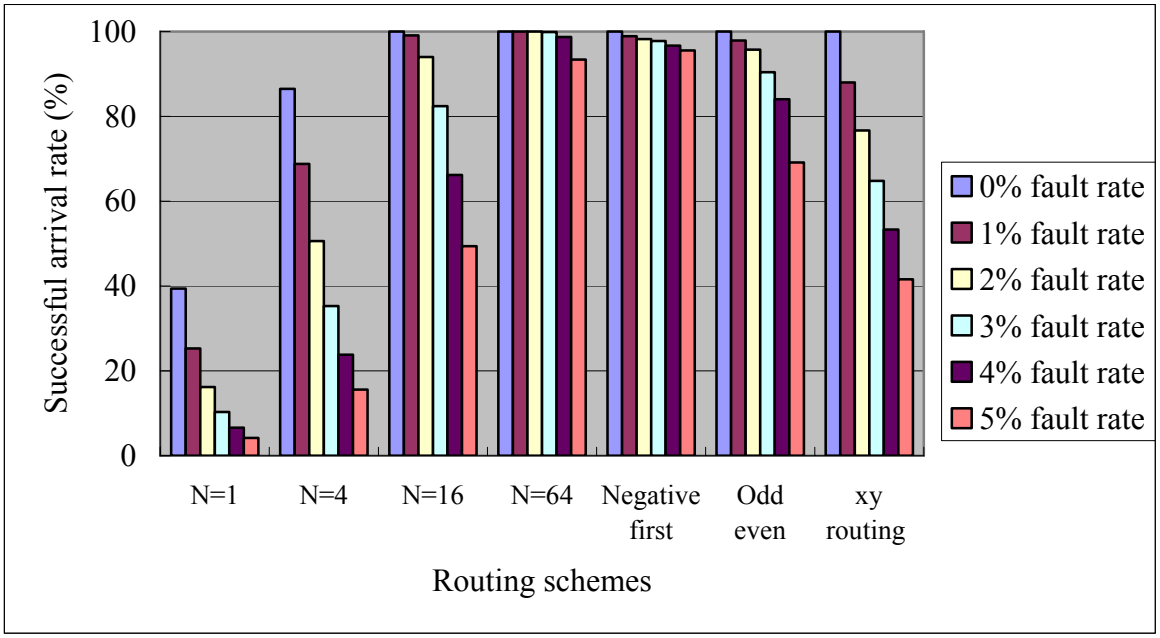

Figure 3.9: Performance with increasing fault rate

3.2.2. Energy Dissipation

When flits travel on the interconnection network, both the inter-switch wires and the logic gates in the switches toggle and this will result in energy dissipation. The flits from the source nodes need to traverse multiple hops consisting of switches and interconnect to reach destinations. Consequently, we determined the energy dissipated in each interconnect and switch hop. The energy per flit per hop is given by:

$$E_{hop} = E_{switch} + E_{\text{int}erconnect} \tag{3.2}$$

The energy dissipated in transporting a message consisting of n flits through h hops can be calculated as:

$$E_{message} = n\sum_{j=1}^{h} E_{hop,j} \tag{3.3}$$

Let p be the total number of messages transported, and let $E_{message_i}$ be the energy dissipated by the i[th] message, where i ranges from 1 to p. The average energy per bit E_{bit} is then calculated according to the following equation:

$$\overline{E_{bit}} = \frac{\sum_{i=1}^{p} E_{message_i}}{p \cdot n} = \frac{\sum_{i=1}^{p} (n_i \sum_{j=1}^{h_i} E_{hop,j})}{p \cdot n} \tag{3.4}$$

In order to quantify the energy dissipation profile for a NoC interconnect architecture, we determine the energy dissipated in each switch, by running Synopsys™ Prime Power on the gate-level netlist of the switch blocks in the 90 nm technology node. To determine interconnect energy, the capacitance of each interconnect stage is calculated taking into account the specific layout of the topology.

Figure 3.10 shows the average energy dissipated per hop for different routing algorithms as discussed above. It is evident that the negative-first routing algorithm dissipates almost the same energy as x-y routing. N random walk dissipates much more energy with increasing N, the number of copies of messages in the network. Though the N=64 random walk algorithm helps sustaining a higher successful data arrival rate in presence of faults, it comes at the cost of significantly higher energy dissipation.

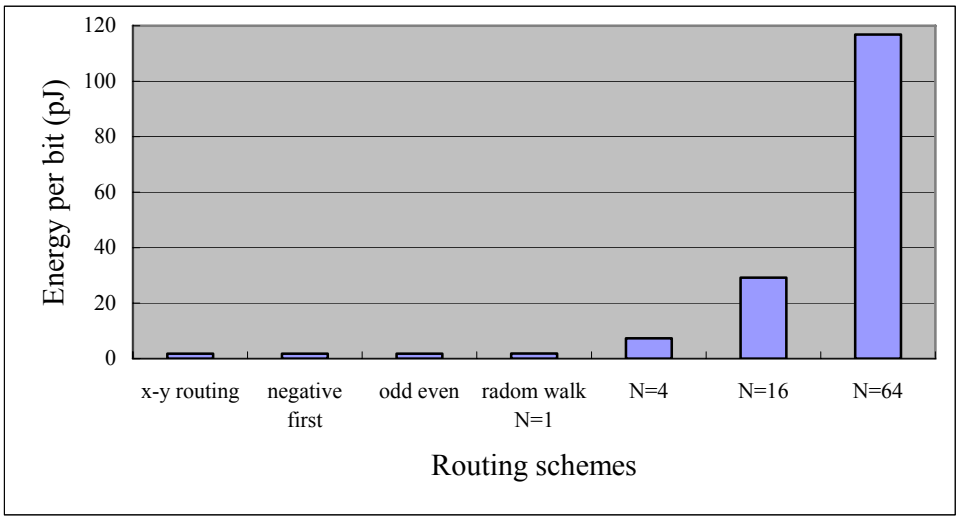

Figure 3.10: Energy dissipation using different routing schemes

3.2.3. Latency

Message latency is the time elapsed between the time a message is generated at its source node and the time the message is delivered at its destination node [22]. It is directly related with the average path length, which is given by the number of hops each message is traversing between a pair of source and destination nodes.

Figure 3.11 shows the average path length for each algorithm in presence of 2% permanent fault, assuming 0.05% injection load. It is evident that negative-first algorithm has the lowest average path length in a Mesh-based network. The average path length for the x-y routing is more than that of negative-first. If there is a fault in the path between any particular pair of source and destination nodes, x-y routing can not route the packet successfully. For the random walk algorithm the average path length decreases with increasing N, as the probability of successful arrival increases with it. But even with N=64, the average path length is more than that for the negative-first algorithm. This happens due to the fact that random walk does not guarantee forwarding packets in the optimum direction. One point worth noting here is that

the injection load was assumed to be very low to get any meaningful results for N-random walk. Consequently this is the best case situation for random walk algorithm. If we increase the injection load then N-random walk will have significantly higher average path length than the negative-first algorithm.

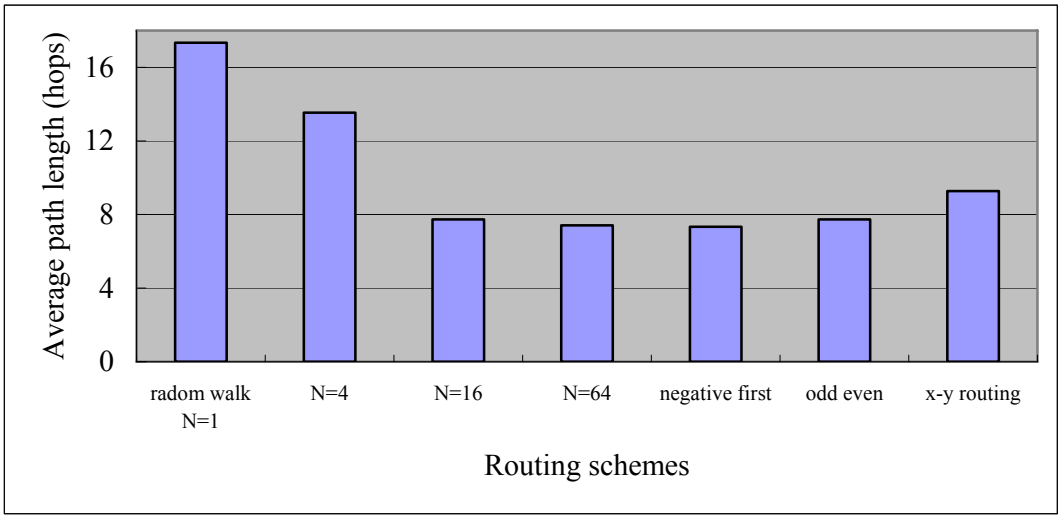

Figure 3.11: Average path length

3.2.4. Area Overhead

We designed and synthesized the routing blocks incorporating deterministic x-y routing, N random walk, odd-even turn model and also the negative-first routing and synthesized using 90 nm standard cell libraries in Synopsys Design Analyzer for a Mesh-based network. We express the silicon area overhead required for a single switch for the different routing schemes discussed in terms of equivalent two-input NAND gates in Table 3.2.

Table 3.2: Silicon area overhead of the routing schemes

Routing Method	Silicon Area (2-input NAND gates)
N=1 random walk	291
x-y routing	86

Negative-first	96
Odd-even turn model	152

From the table, it is evident that x-y routing and negative-first have very comparable area overhead while N random walk needs much more area even for N=1.

3.3 Conclusions

Deterministic routing algorithm like x-y routing is unable to sustain high throughput when increasing the number of faults in NoC. N random walk is not suitable for tolerating permanent faults in a NoC environment because it quickly saturate the network with the higher injection rates. By incorporating fault tolerant algorithms other than N random walk in NoC data communication it is possible to improve the system throughput in presence of a certain number of permanent faults. Negative-first and odd-even turn model have the competitive throughput with 5% fault rate in NoC, while odd-even has a poor data successful arrival rate than that of negative-first. All fault tolerant algorithms have higher power dissipation than x-y routing algorithm because of the additional hardware block rerouting messages in presence of faults. It is also observed that fault tolerant algorithms have higher silicon area overhead due to the same reason as power dissipation.

CHAPTER 4

FAULT TOLERANCE IN BFT BASED NOC

A tree based network is another commonly interconnect architecture used for NoC. Till date a couple of tree-based NoCs have been proposed; (1) the generic fat-tree based architecture (SPIN) [7] and (2) The butterfly fat tree (BFT)-based NoC [8]. Typically the nodes of the tree correspond to switches or functional blocks while the edges represent communication links. In this chapter we explore design methodologies to achieve fault tolerance in Tree-based NoCs. We consider only the BFT-based architecture for detailed analysis.

4.1. Network Architecture of Fault Tolerant BFT

In BFT architecture each switch is connected to four children and two parents. There is more than one shortest path between a pair of source and destination nodes in the Butterfly Fat Tree. More precisely, a message can take any one of the two up links from a switch, if the destination is not in the same sub-tree [4]. For instance, it selects an up-link randomly, if that link is blocked, it tries the other, and if both are blocked, it waits. But there is no redundancy for down links. From the BFT architecture it is evident that if a particular switch is faulty then the IP blocks connected to the switch do not have any other path to communicate with the rest of the network. Consequently instead of depending on the modification of the routing algorithm, we propose to modify the overall architecture to incorporate fault tolerance in a

BFT-based NoC as shown in Figure 4.1.

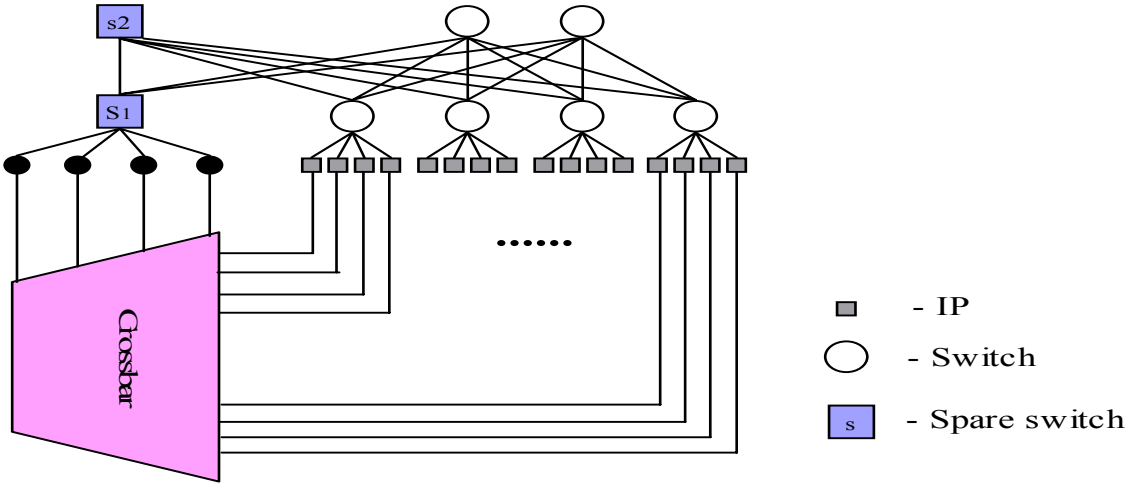

Figure 4.1: Fault tolerant BFT

A fault tolerant BFT adds two spare switches and one crossbar to a block of 16 IPs as shown in Figure 4.1. The spare switches have the same structure as the regular ones. The crossbar is connected to spare switch *S1* at the one side and connected to 16 IP blocks at the other side. Therefore, 4 different channels from the spare switch to any of the 4 IPs can be created with the help of the crossbar simultaneously. The spare hardware block consisting of the two additional switch blocks, *S1, S2* and the crossbar is added to sub network of 16 IP blocks as shown in Figure 4.1. Consequently, a fault tolerant BFT with *N* IP nodes has 5*N/16* switches at the first level; and at the second level of the network, there are 3*N/16* switches. At the jth level of the network, there are $5N/2^{j+3}$ switches ($j \geq 3$). In a total, there are N/16 crossbars in the fault tolerant BFT where N indicates the number of IP blocks in system. It is worth noting that by adding spare switches and crossbar we can reconfigure the network resources so that the overall BFT interconnection structure remains unchanged in presence of faults.

4.2. Design of Crossbar

A crossbar is used in fault tolerant BFT NoC to maximize the utilization of the spare switch *S1*. Basically there are two portions when designing a crossbar. One is to create channels from spare switch *S1* to IPs and the other is the opposite way. More specifically, the first part is multiplexer to support the packets going from spare switch through crossbar to IPs according to the destination IP address. The second part works like a demultiplexer, which transmits data packets from IPs to spare switch, mainly controlled by the available links connected to the spare switch *S1*. In this case, the crossbar is unable to create more than five channels due to the number of incoming links for the spare switch.

4.3. LCA based Routing Algorithm

The LCA (Least Common Ancestor) [4] [29] based routing algorithm is the most commonly used methodology for message transfer in tree based NoC, e.g., BFT NoCs. The idea of LCA is that data packets are first sent upwards until the common ancestor for both source and destination nodes is reached. Then from common ancestor, the packets follow a particular downward link.

In this case, the first step in the implementation of LCA routing logic involves the bit-wise comparison (XOR) of the source and destination addresses taking the most significant, i.e, M=(log2N – 2l) bits, where N is the number of functional IP blocks in the system and l denotes the level number of the switch. Subsequently, the result of the comparison is checked, i.e., whether any “1” results from the bit-wise XOR operation. The basic structure of the hardware block implementing the LCA algorithm is shown in Figure

4.2.

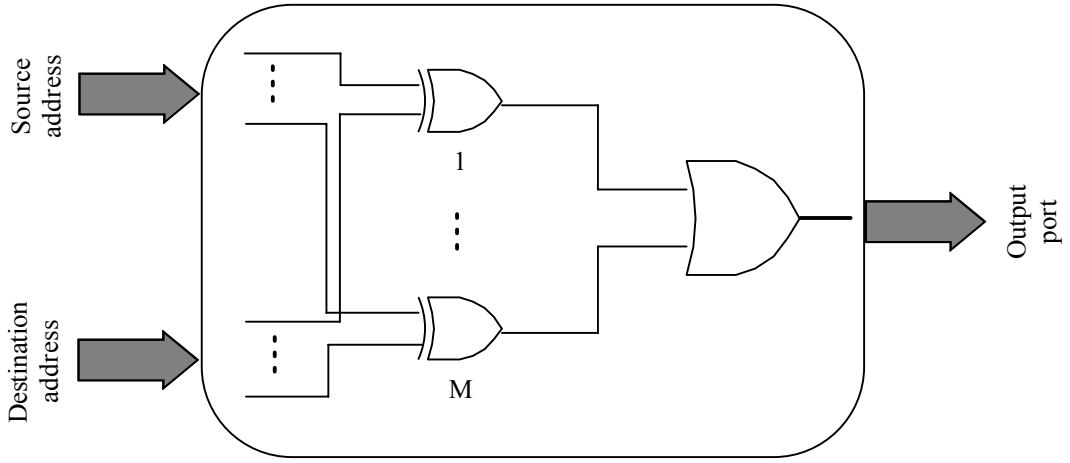

Figure 4.2: Block diagram of the LCA routing

The effectiveness of this routing methodology in presence of a fault in the BFT as well as the fault tolerant BFT NoC interconnect architecture is explained with the help of Figure 4.3 (a) and (b) respectively.

When the network has a fault situated on the path from source IP_C to destination IP_D, there are two possibilities as shown in Figure 4.3. In a regular BFT NoC as shown in Figure 4.3 (a), the message packet is blocked by the faulty link and fails to reach the destination eventually. While with the help of spare switch in a fault tolerant BFT in Figure 4.3 (b), IP_C is able to forward the packet to the spare switch *S2*. Then from spare switch *S2* to *S1*, the packet is sent to the destination IP_D through the crossbar successfully.

The system characteristics including throughput, latency, silicon area overhead, and power dissipation for these two architectures are discussed in the next subsection.

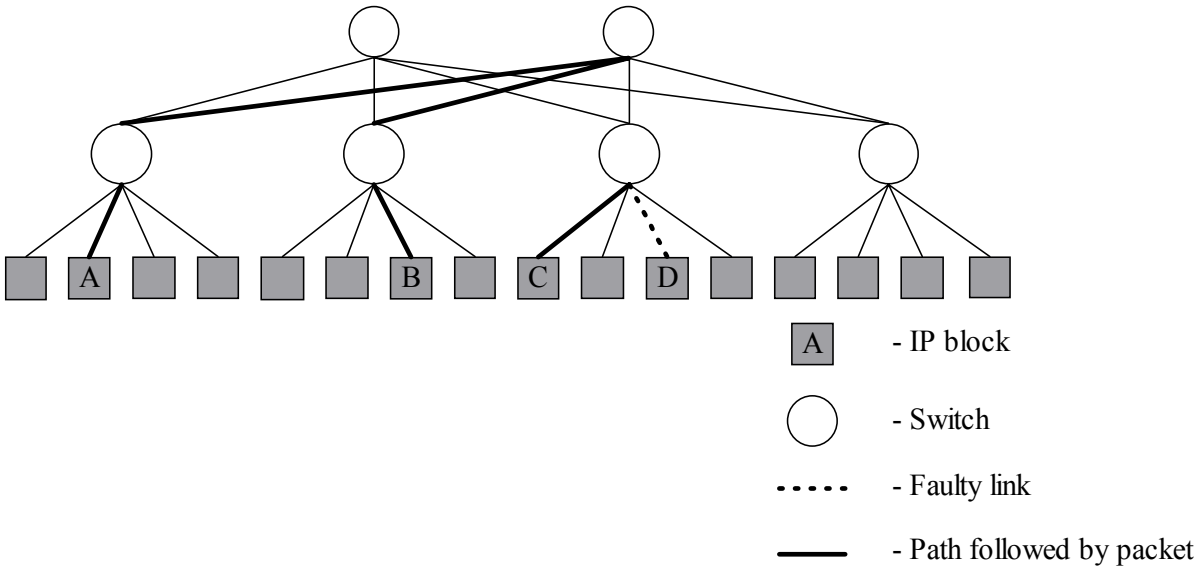

Figure 4.3 (a): LCA routing sample in BFT

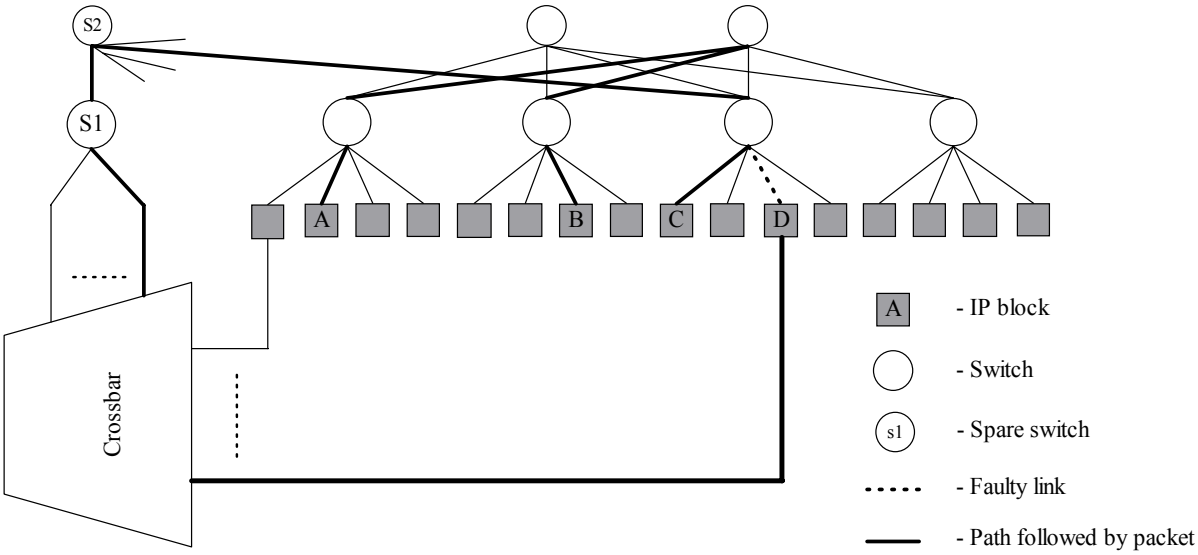

Figure 4.3 (b): LCA routing sample in fault tolerant BFT

4.4. Experimental Results

As we demonstrated in Mesh-based NoC, we also evaluated BFT based NoC in terms of throughput, latency, area overhead, and power dissipation in presence of faults. We consider the same system size of 64 IP blocks. To study the system performance

characteristics mentioned above, messages were injected with a uniform traffic pattern. Faults are generated randomly in the network. LCA routing algorithms is adopted by both regular and fault tolerant BFT. Simulations were performed using 90nm standard cell library from CMP [30]. The parameters used for the purpose of simulations are listed in Table 4.1.

Table 4.1: Simulation parameters

Architecture	Message Length (Flits)	Buffer Depth (Flits)	Number of ports	
			Child ports	Parent ports
BFT	16	2	4	2
Fault Tolerant BFT	16	2	5	3

4.4.1. Throughput

Figure 4.4 shows the performance of the NoC with a 5% fault rate mapped onto both regular and fault tolerant BFT based NoC. It is observed that fault tolerant BFT outperforms the regular BFT in terms of throughput. In presence of faults, even though the regular BFT has alternative path while going up, it is unable to forward packets to destination if a down link is faulty. On the contrary, the fault tolerant BFT can utilize the spare switch as a backup when transferring packets to lower level switches. Consequently, the higher throughput for fault tolerant BFT in presence of faults is reasonable.

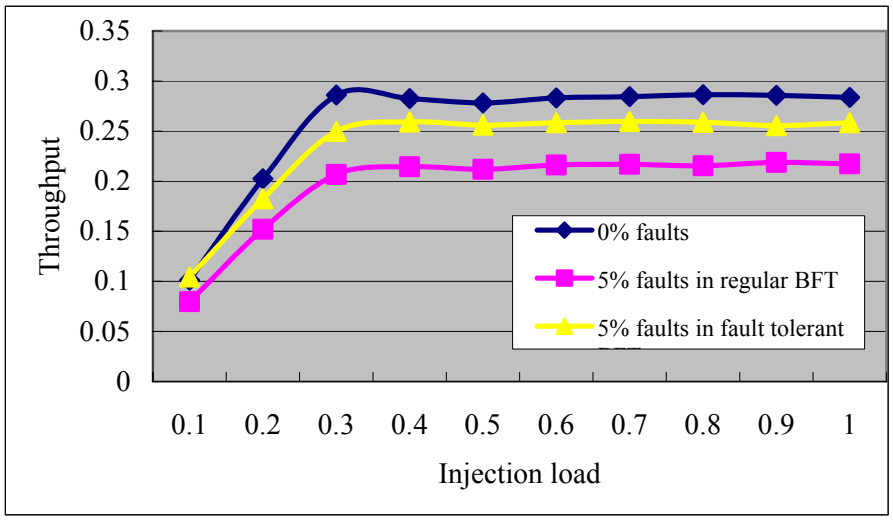

Figure 4.4: Network throughput for BFT based NoC

4.4.2. Energy Dissipation

Following equation (3.4) we determine the energy dissipation profile of the NoC. We determine the energy dissipated in each switch, by running Synopsys™ Prime Power on the gate-level netlist of the switch blocks in the 90 nm technology node. To determine interconnect energy, the capacitance of each interconnect stage is calculated taking into account the specific layout of the topology. We present the average energy dissipated per bit for two different BFT architectures in Table 4.2. It is evident that the fault tolerant BFT dissipates higher energy than that of regular one, which is due to the necessary data rerouting through spare hardware block.

Table 4.2: Energy dissipation and area overhead for BFT based NoC

	Regular BFT	Fault tolerant BFT
Energy dissipation (pJ/bit)	2.20	2.26
Area overhead (gates)	67	219
Average path length (hops)	5. 423	5. 934

4.4.3. Area Overhead

We designed and synthesized the routing blocks as well as the crossbar for the BFT based networks and synthesized using 90 nm standard cell libraries in Synopsys Design Analyzer. We express the average silicon area overhead required for a single switch for the different topologies discussed in terms of equivalent two-input NAND gates in Table 4.2. It is evident that the fault tolerant BFT suffered from the additional hardware block. Thus it consumes much more silicon area than for a regular BFT even though it can sustain a higher throughput.

4.4.4. Latency

The average path length for each network structure in presence of 2% permanent fault, assuming 0.05% injection load is listed in table 4.2. It is evident that both of these two BFT based networks have a shorter average path length than that of Mesh-based network due to the less inter-switch links and switches inside the network. A fault tolerant BFT needs to forward data packet to spare switches in order to reach destination in presence of faults. Hence it has a longer average path length than regular BFT which is expected.

4.5 Conclusion

By incorporating spare hardware block consisting of the two additional switch blocks, *S1, S2* and the crossbar into a sub network of 16 IP blocks, the fault tolerant BFT is able to improve the ability of tolerating faults. As verified through experimental results, the proposed BFT design maintains a higher throughput compared to the regular BFT. It requires almost

same power dissipation for each bit transferred. It is also observed that such a BFT structure would consume more area due to added spare hardware block. And a higher average path length is expected in presence of faults for a fault tolerant BFT because rerouting through spare switches and crossbar is necessary.

CHAPTER 5

YIELD ENHANCEMENT

The yield of an integrated circuit is the fraction of IC chips that meet a specific set of functional requirements out of the total number of chips manufactured. In the DSM era, yield-loss can be caused by imperfections in the fabrication process. Moreover, the life-time reliability of DSM devices is likely to be compromised by effects such as electromigration and material ageing which consequently reduces the yield.

Incorporating hardware redundancy to address fault tolerance provides a very practical solution to the low yield problem [31]. Applicability of the hardware redundancy-based method for yield enhancement has been demonstrated for high-density memory chips or multi-processor platforms. With the increasing density of integration in a single chip, it becomes very difficult to enhance yield by means of restructuring hardware. Moreover, hardware redundant components need extra area and power. Proportional to number of processing nodes in a single chip, the extra cost of area and power of redundant components is increasing significantly.

In this work, we intend to quantify the effects of the fault tolerant design methods discussed earlier on the yield of NoC-based chips. In this context, it is possible to improve chip yield if system performance, i.e. network throughput, is sustained at a certain level in presence of faults. Fault tolerant mechanisms for NoC architecture discussed in previous chapters are able to establish alternate routing paths with faults existing in system.

Consequently, a chip incorporating fault tolerant mechanisms demonstrates a certain capability of maintaining system performance. Therefore, chip yield can be improved by incorporating such mechanisms.

For many consumer applications, whether a product provides acceptable quality or not depends on the system performance it produces and the requirements of the customers. Therefore, a chip identified as faulty in the classical testing flow might still be acceptable if any fault tolerant algorithm is adopted such that system performance is maintained higher than the acceptable value specified by the users or system developers.

5.1. Yield Calculation

The manufacturing yield is defined as the fraction (or percentage) of acceptable chips among all chips that are fabricated. During the manufacturing process, defects and faults such as shorts or interconnection open have the most significant impact on yield loss. In the past, the spatial Poisson distribution was used to describe the defect and fault distribution. It assumes a uniform distribution of defects and faults over the wafer resulting in:

$$Prob(Number\ of\ Faults\ on\ Chip = K) = e^{-\lambda}\frac{\lambda^k}{k!} \tag{5.1}$$

Therefore, $Chip\ Yield = Prob(Number\ of\ Faults = 0) = e^{-\lambda}$ (5.2)

where λ is the average number of faults per chip and α is the clustering parameter, typically, $0.5 \le \alpha \le 5$. This is a very simplistic model and does not fit empirical manufacturing data. In practice, both defects and faults are more clustered than predicted by the pure Poisson distribution. A general yield model [32] for VLSI manufacturing, which is given by equation (5.3), is commonly employed to estimate the yield sensitivity of chips to random failure

mechanisms.

$$p(k) = Prob(number\ of\ \ faults\ on\ chip = k)$$
$$= \frac{\Gamma(\alpha + k)}{k!\Gamma(\alpha)} \frac{(Ad/\alpha)^k}{(1 + Ad/\alpha)^{\alpha+k}} \tag{5.3}$$

where $\Gamma(x)$ is the Gamma function, defined as (5.4):

$$\Gamma(x) = \int_0^\infty e^{-y} y^{x-1} dy \tag{5.4}$$

The mean $E(x)$, and variance $\sigma^2(x)$, of x are given by:

$$E(x) = Ad \tag{5.5}$$

$$\sigma^2(x) = Ad(1 + Ad/\alpha) \tag{5.6}$$

The values of $E(x)$ and $\sigma^2(x)$ for the number of defects on a chip are obtained either by experimental measurements or by process simulation. Substitution in the above equations then leads to the determination of yield parameters, Ad and α. The yield is obtained as the probability $p(0)$, of no defect on a chip. Thus, substituting $k = 0$ in equation (5.3), we can get:

$$Y = (1 + Ad/\alpha)^{-\alpha} \tag{5.7}$$

When $\alpha \to \infty$, equation (5.3) gives the Poisson density function with mean Ad, which corresponds to the unclustered distribution of defects:

For unclustered defects: $p(x) = \frac{(Ad)^x e^{-Ad}}{x!}$ (5.8)

On substituting $x = 0$ this gives the yield, which can also be obtained by substituting $\alpha = \infty$ in equation (5.7), as

$$Y_{Poisson} = e^{-Ad} \tag{5.9}$$

Equation (5.8) and (5.9) are simplified models for unclustered distribution of defects, as given in (5.1) and (5.2) previously. In this work, equation (5.7) could be accepted as a general model for chip yield estimation, in which, Ad and α are average faults per chip and

clustering parameter respectively.

Equation (5.7) is not sufficient in some scenario especially for yield enhancement situation. The reason behind this is that (5.7) only takes average number of faults on chip and clustering parameter into consideration. However, for some yield enhancing methodologies, area cost is a very important factor because those methodologies, e.g. redundant components, used to increase the yield would cost more area. Thus, even though introduced methodologies might provide with a higher yield, we may end up with fewer operational chips per wafer because of the larger chip area. Therefore, effective yield [33] is considered as a more suitable feature to evaluate yield improvement. The definition of effective yield is given by (5.10):

$$Y_{effective} = Y_{chip} \frac{Area \quad of \quad Chip \quad without \quad Modification}{Area \quad of \quad Chip \quad with \quad Modification} \tag{5.10}$$

5.2. Experimental Results for Yield Enhancement

Deterministic routing algorithm such as x-y routing algorithm and partially adaptive routing algorithm like negative-first and odd-even turn model algorithms are two different types of algorithms we applied on Mesh based NoC. For a BFT base NoC, we considered a modified architecture. We evaluate the performance of the routing mechanisms discussed above when mapped onto a system consisting of 64 IP blocks. System throughput, which reflects the sustainable data rate of the on-chip network, is regarded as the most important performance metric. To quantify the effects of the fault-tolerant design methods described in the previous chapters on the system yield, we consider the variation of the throughput in presence of faults. We assume that the system is usable if the throughput is degraded by 10% compared to the fault-free case in presence of faults. We first studied the variation of

throughput as function of the number of faults considering the deterministic and partially adaptive routing algorithms. Figure 5.1 shows the throughput degradation of the Mesh-based NoC in presents of faults. It is evident that by using negative-first routing, the throughput degradation overall is below 10% compared to 18% degradation for odd-even turn model, and 44% for x-y routing algorithm when 8 faults are generated in system. In other words, the negative-first algorithm is able to tolerate up to 8 faults in a 64-IP Mesh-based NoC. Odd-even turn model is able to tolerate up to 6 faults.

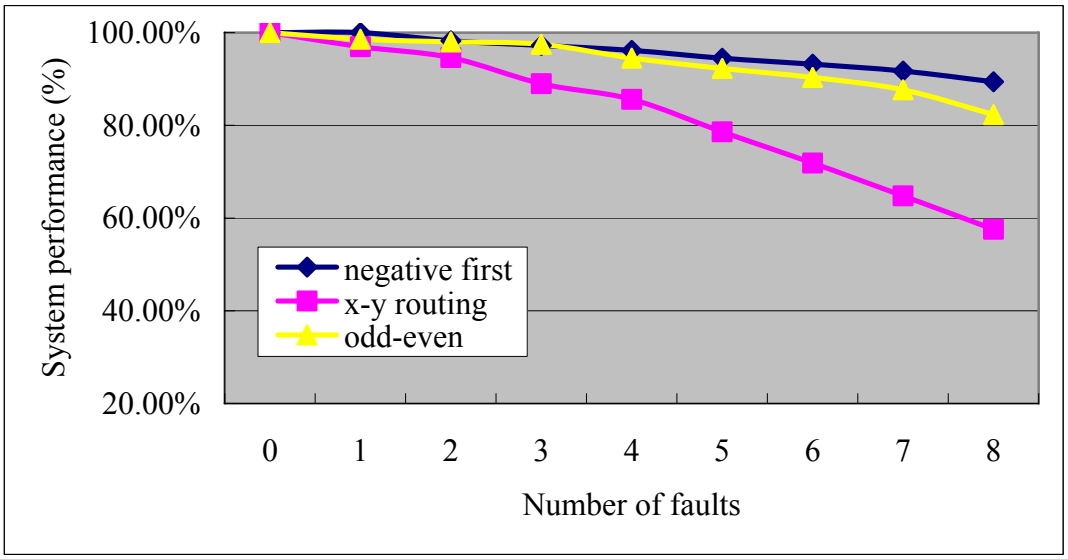

Figure 5.1: System throughput degradation for different routing algorithms in presence of faults

According to equation (5.7), we can calculate the yield estimation when we select $\alpha = 2$ as a typical clustering parameter. In this case, we ignore the instance where all of the NoC based systems using different schemes can provide a desire throughput. We only consider the situation where throughput degradation is larger than 10% to explore the ability of yield enhancement by incorporating fault tolerant algorithms. The yield estimation is shown in Figure 5.2. It is evident that by incorporating the negative-first algorithm the Mesh-based NoC has the highest yield with increasing the number of faults. A NoC can have

a higher yield if odd-even turn model is applied than x-y routing. The chip yield drops very fast with increasing number of faults, if x-y routing is adopted. As a result, for a Mesh based NoC, fault tolerant algorithms such as negative-first and odd-even turn model are able to enhance chip yield in comparison to deterministic routing like x-y routing.

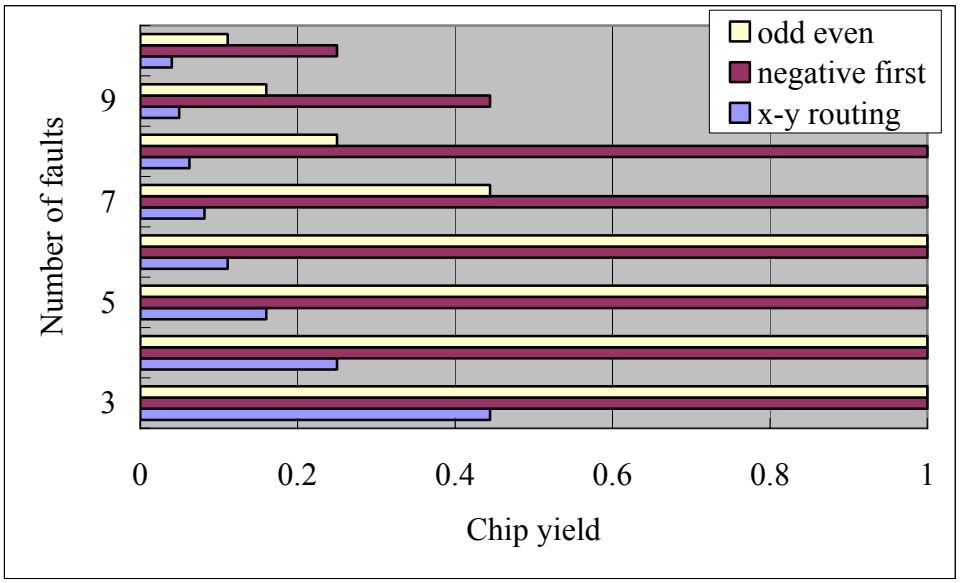

Figure 5.2: Yield estimation for different routing schemes for a Mesh-based NoC

It is observed that fault tolerant algorithms such as negative-first and odd-even cost more area than that of x-y routing. The silicon area costs of algorithms are given in chapter 3 (Table 3.2). With the higher area overhead, it is unable to yield the same number of chips on the identical wafers if fault tolerant algorithms are adopted compared to the x-y routing. So to evaluate these two types of algorithms fairly, we need to take the area overhead into account. Thus, effective yield (equation (5.10)) is considered, which is shown in Figure 5.3.

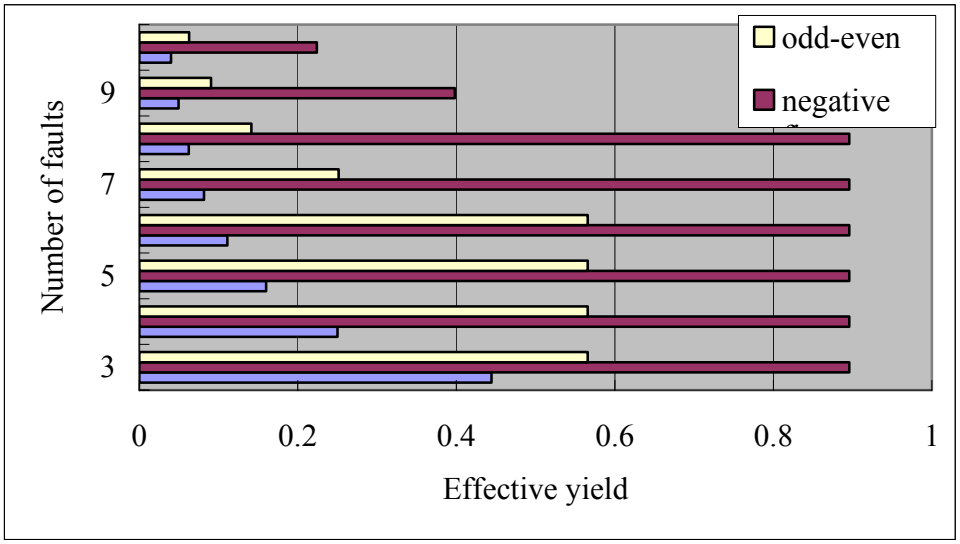

Figure 5.3: Effective yield for different routing schemes for a Mesh-based NoC

It is evident that negative-first and odd-even turn model can provide with higher effective yields in presence of faults though they consume more area than the x-y routing. The area overhead for fault tolerant algorithms reveals negative impact on the effective yield compared to the chip yield shown in Figure 5.2. In this case, negative-first and odd-even turn model algorithms demonstrate a lower effective yield with comparison to chip yield because of the higher area overhead requirements. To sum up, a Mesh-base NoC provides a better overall effective yield in presence of higher number of faults if fault tolerant mechanisms are incorporated.

The system performance in presence of fault for a BFT based network is shown in Figure 5.4. As shown in the figure, the system performance degradation for a regular BFT is over 10% when there are more than 2 faults found in the network. Compared with a regular BFT, a fault tolerant BFT network can keep throughput degradation within 10% when the number of faults is up to 6. The system yield and effective yield for both regular and fault tolerant BFT NoC are shown in Figure 5.5 and 5.6

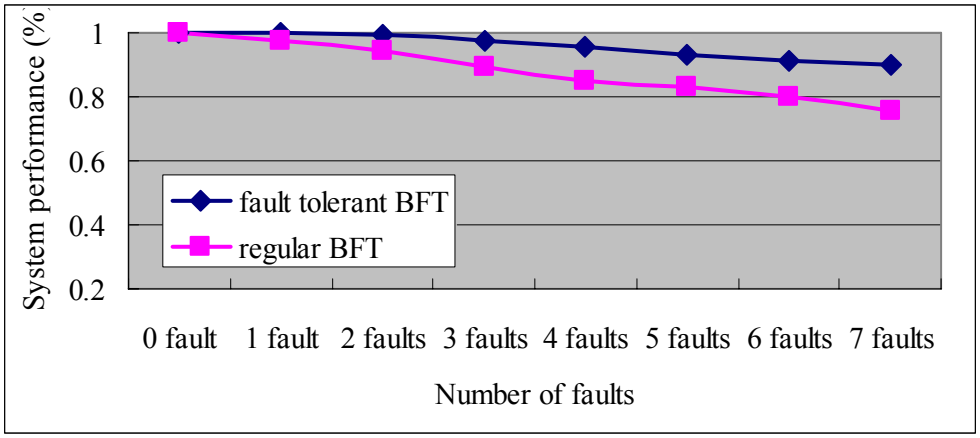

Figure 5.4: System performance for a BFT based network

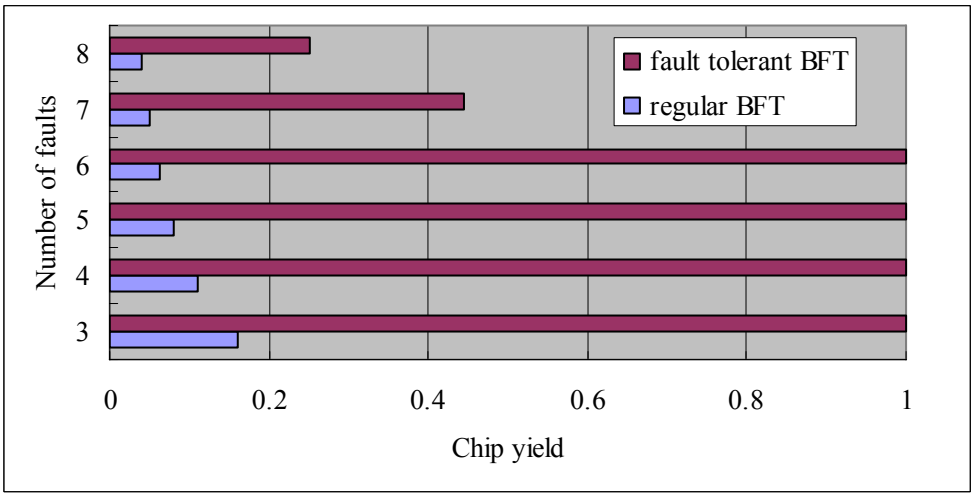

Figure 5.5: Yield for both regular and fault tolerant BFT network

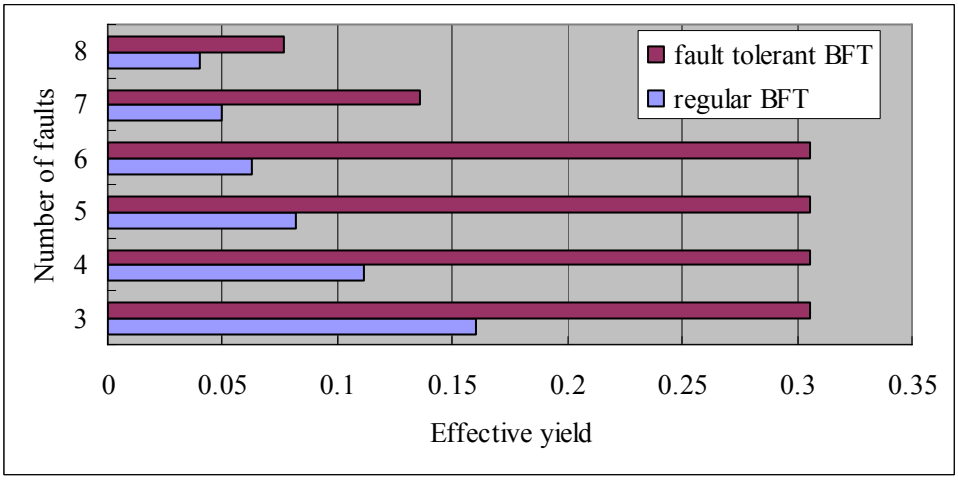

Figure 5.6: Effective yield for both regular and fault tolerant BFT network

It is evident that fault tolerant BFT is able to improve chip yield compared to regular BFT. The fault tolerant BFT can provide 100% chip yield for up to 6 faults in the NoC. However for the regular BFT the chip yield degrades very fast with increasing number of faults. The fault tolerant BFT suffers from the area overhead due to the spare hardware block so that it has a lower effective yield with comparison to chip yield shown in Figure 5.5.

5.3. Conclusion

In this chapter, we have discussed how the fault tolerant design methodologies considered in this work enhances system yield. It is demonstrated through experimental results that in case of a Mesh-based NoC, negative-first algorithm provides the highest yield in presence of faults. We have also demonstrated how a fault tolerant BFT-based NoC achieves better yield compared to a regular BFT.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1. Conclusions

In the DSM era, yield-loss can be caused by any imperfections in the fabrication process. The life-time reliability of DSM devices is likely to be compromised by effects such as electromigration and material ageing. Fault tolerant methodologies should be adopted at the design stage to improve system reliability and to enhance chip yield for NoC architectures.

In this work, we demonstrated that by incorporating fault tolerant mechanisms in the NoC communication fabric a certain number of permanent faults can be tolerated and hence overall system yield is improved. We have shown that in case of a Mesh-based NoC partially adaptive routing mechanisms like negative-first and odd-even turn model outperform stochastic routing mechanism such as random walk. Negative-first outperforms odd-even turn model with a higher capability of tolerating permanent faults. Both of these two algorithms are able to achieve higher throughput compared to deterministic routing scheme such as x-y routing in presence of faults. We also demonstrate that by adding spare hardware block in a regular BFT-based NoC its fault tolerance capability can be enhanced. The fault tolerant BFT outperforms the regular one in terms of network throughput because it not only has redundant uplinks but also downlinks through the spare hardware block. The above fault tolerant methodologies improve the performance of NoCs in presence of faults. Hence by

incorporating these methodologies in NoC architectures chip yield can be enhanced. We have demonstrated how the yield of Mesh and BFT-based NoC can be increased in presence of faults with the help of the fault tolerant design methodologies described in this work.

6.2. Future Work

In the following possible extension of this work in future is elaborated.

6.2.1. Fully Adaptive Routing Algorithm

Partially adaptive routing algorithms such as negative-first and odd-even turn model can provide certain level of fault tolerance as we demonstrated in this work. But these algorithms are not suitable when the number of faults become large. One reason is because the partially adaptive routing algorithms only require the status information (faulty or not) of neighboring link or switch. Therefore, one possible future work for this research topic is to explore performance of fully adaptive routing algorithm in NoC environment. Though fully adaptive routing algorithm requires more power and area overhead than existing partially adaptive ones, it can tolerate more faults and make NoCs more reliable. One of the suitable algorithms is Adaptive Fault Tolerant Wormhole Routing Algorithm (AFTRouting) [23] [24] which is explained as follow.

Adaptive fault tolerant wormhole routing algorithm (AFTRouting) [23] [24] [25] is based on a convex fault model in 2D Meshes. With the algorithm, a normal routing message, when blocked by faulty routers, would detour along some special polygons around the fault region. In 2D Mesh network, two processing node (PN) (X_0, Y_0) and (X_1, Y_1) are called

4-neighbors if $|(X_0 - X_1) + (Y_0 - Y_1)| = 1$, 8-neighbors if $\max\{|X_0 - X_1|, |Y_0 - Y_1|\} \le 1$. A sample is shown in the following figure. Assume every PN in 2D Mesh communicates with its neighbors through its input ports $\{W_{i1}, E_{i1}, N_{i1}, S_{i1}\}$ and output ports $\{W_{i2}, E_{i2}, N_{i2}, S_{i2}\}$ in a virtual network VM_i, i=1, 2 respectively.

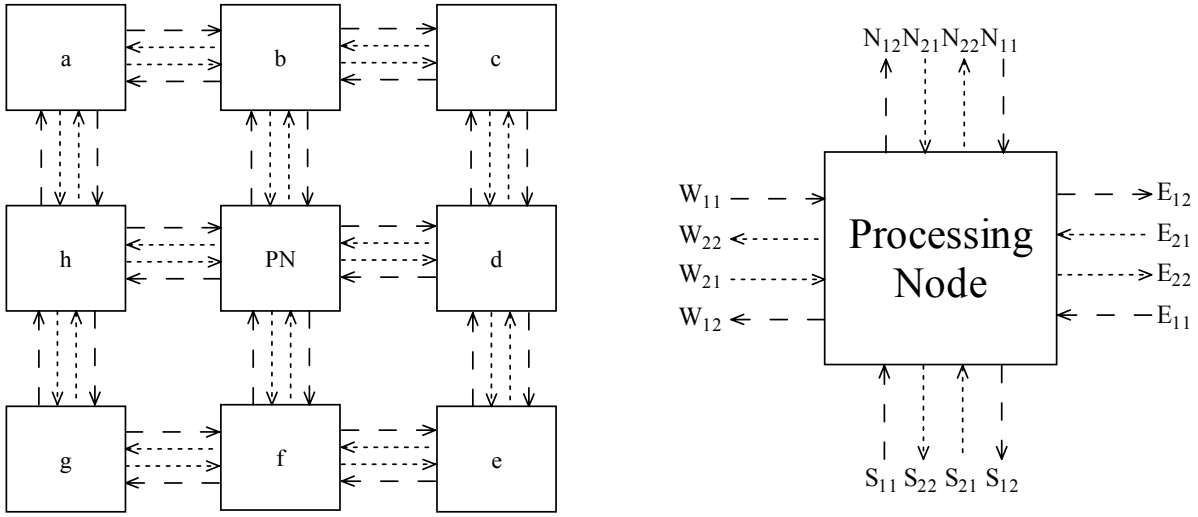

Figure 6.1: 8 neighbors and the channels of a PN

The message routings according to current PN (X_c, Y_c) and destination (X_d, Y_d) are divided into four types: X^+, X^-, Y^+, and Y^- routings, where X^+ routing if $X_c < X_d$; X^- routing if $X_c > X_d$; Y^+ routing if $X_c = X_d$ and $Y_c < Y_d$; Y^- routing if $X_c = X_d$ and $Y_c > Y_d$. In algorithm AFTRouting, when there exists an outgoing channel in a shortest path from current node to destination, which is not blocked by fault regions, a normal adaptive routing channel is used to route messages according to the Table 6.1. When a normal minimal adaptive routing is blocked by a fault-region, a flag is been set and the distance from current PN to the destination is stored as d_{flag}. The message is misrouted according AFTRouting rules along the f-polygon of the fault-region until $d_c < d_{flag}$, i.e., the routing header moves at least one hop closer to the destination.

Table 6.1: Strategy for channel selection of normal message routing

Routing type	Channels in qualified virtual networks	Channels in denoted networks
X^+	in VM_1	in VM_1
X^-	in VM_1 or VM_2	in VM_2
Y^+	in VM_1 or VM_2	in VM_2
Y^-	in VM_1 or VM_2	in VM_2

If the f-polygon of the fault-region is f-ring, the misrouting channels are determined by Table 6.2. Otherwise if it is blocked by an f-chain, the message will be misrouting along the fault-region according to Table 6.3.

Table 6.2: Strategy for channel selection of misrouting along f-rings

Routing Type	Routing Channel
X^+ routing	in VM_1
X^- routing	in VM_2
Y^+ routing	in VM_2
Y^- routing	in VM_1

Table 6.3: Channel selection strategy for misrouting along f-chains

Routing Type	Port of f-chain head	Position of current PE	Channel
X^+ routing	S_{12}		in VM_1
	W_{12}	$d_y > c_y$	in VM_1
		$d_y < c_y$	in VM_2
	N_{12}		in VM_2
X^- routing	E_{12}	$d_y > c_y$	in VM_2
		$d_y < c_y$	in VM_1
	S_{12}		in VM_2
	N_{12}		in VM_1
Y^+ routing	E_{12}		in VM_2
	W_{12}		in VM_1
Y^- routing	E_{12}		in VM_1
	W_{12}		in VM_2

This proposed algorithm can tolerate convex fault regions regardless of possible overlapping of the boundaries of different fault regions. Only two virtual channels per physical channel are required.

The future work regarding to the fully adaptive routing algorithm is to evaluate the performance of the AFTRouting algorithm on the NoC platform and make it suitable for NoC environment when making trade off among network throughput in presence of faults, energy dissipation, latency and area overhead.

6.2.2. Three Dimensional NoC

Three dimensional (3D) Network on Chip (NoC) has recently attracted researchers' attention. 3D NoC's are capable of achieving better system throughput and lower latency compared to the corresponding 2D implementations. To fully exploit the performance benefits of 3D architectures, it is imperative to address system reliability issues in the design phase. The fault tolerant algorithms investigated in this work, like, the negative-first and the odd-even routing algorithms can be incorporated into 3D NoCs also, with some modifications to take into account the effects arising out of one additional dimension. One of the possible future directions will be to evaluate the performance of the fault tolerant design methods investigated in this thesis for 3D NoC architectures.

6.3. Summary

The Network-on-Chip (NoC) design paradigm is viewed as an enabling solution for the integration of exceedingly high number of computational and storage blocks in a single chip. For DSM VLSI processes, it is difficult to guarantee correct fabrication with an acceptable yield without employing design techniques that take into account existence of manufacturing defects. By incorporating fault tolerant methodologies for NoC architecture, higher system

performance and yield enhancement can be achieved while keeping overheads within acceptable limits.

BIBLIOGRAPHY

[1] L. Benini, G. De Micheli, “Networks on Chips: A New SoC Paradigm”, Computer, Volume: 35 Issue: 1, Jan. 2002. pp. 70 - 78.

[2] P. P. Pande, C. Grecu, A. Ivanov, R. Saleh, G. De Micheli, “Design, Synthesis and Test of Networks on Chip”, IEEE Design and Test of Computers, Vol. 22, No. 5, 2005. pp. 404 - 413.

[3] R. I. Greenberg, Hyeong-Cheol Oh, “Universal wormhole routing”, IEEE Transactions on Parallel and Distributed Systems, Volume 8, Issue 3, March 1997. pp. 254 – 262.

[4] R. I. Greenberg, Lee Guan, “An Improved Analytical Model for Wormhole Routed Networks with Application to Butterfly Fat-Trees”, Proceedings of International Conference on Parallel Processing, 1997. pp. 44 - 48.

[5] W. J. Dally, and B. Towles, “Route Packets, Not Wires: On-Chip Interconnection Networks”, Proceedings of Design Automation Conference (DAC), Las Vegas, Nevada, USA, June 18-22, 2001. pp. 683 - 689.

[6] S. Kumar, A. Jantsch, J. P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, A. Hemani, “A Network on Chip Architecture and Design Methodology,” Proceedings of IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Pittsburgh, USA, 2002. pp. 117 - 124.

[7] P. Guerrier, A. Greiner, ”A Generic Architecture for On-Chip Packet-switched Interconnections”, Proceedings of Design, Automation and Test in Europe (DATE), Paris, France, March 27-30, 2000. pp. 250 - 256.

[8] P. P. Pande, C. Grecu, A. Ivanov, R. Saleh, “Design of a Switch for Network on Chip Applications”, Proceedings of International Symposium on Circuits and Systems (ISCAS), Volume 5, Bangkok, May 2003. pp. 217 - 220.

[9] C. Grecu, A. Ivanov, R. Saleh, P. P. Pande, “NoC Interconnect Yield Improvement Using Crosspoint Redundancy”, 21st IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2006. DFT Oct. 2006. pp. 457 - 465.

[10] D. A. Hodges, H. G. Jackson and R. A. Saleh, “Analysis and Design of Digital Integrated Circuit: In Deep Submicron Technology”, Third Edition, 2004.

[11] T. L. Michalka, R. C. Varshney, J. D. Meindl, “A Discussion of Yield Modeling with Defect Clustering, Circuit Repair, and Circuit Redundancy”, IEEE Transactions on Semiconductor Manufacturing, Volume 3, Issue 3, Aug. 1990. pp. 116 - 127.

[12] I. Koren and Z. Koren, “Defect Tolerant VLSI Circuits: Techniques and Yield Analysis”, Proceedings of the IEEE, Vol. 86, Sept. 1998. pp. 1817 - 1836.

[13] T. Dumitras, S. Kerner, R. Marculescu, “Towards on-chip Fault-tolerant Communication”, Proceedings of the 40th Design Automation Conference DAC 2003, 21-24 Jan, 2003. pp: 225 - 232.

[14] M. Pirretti, G. M. Link, R. R. Brooks, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, “Fault Tolerant Algorithms for Network-on-Chip Interconnect”, IEEE Computer Society Annual Symposium on VLSI, 2004. 19-20 Feb. 2004. pp. 46 - 51.

[15] J. Kim, D. Park, T. Theocharides, N. Vijaykrishnan, C. R. Das, “A Low Latency Router Supporting Adaptivity for On-Chip Interconnects”, 42nd Design Automation Conference DAC 2005. pp. 559 - 564.

[16] C. J. Glass and L. M. Ni, “The Turn Model for Adaptive Routing”, 19th Annual International Symposium on Computer Architecture, May 1992. pp: 278 - 287.

[17] C. J. Glass and L. M. Ni, “Adaptive Routing in Mesh-connected Networks”, 12th International Conference on Distributed Computing Systems, Jun. 1992. pp: 12 - 19.

[18] J. Hu, R. Marculescu, “DyAD - Smart Routing for Networks-on-Chip”, 41st Proceeding of Design Automation Conference, DAC 2004. pp. 260 - 263.

[19] Ge-Ming Chiu, “The Odd-even Turn Model for Adaptive Routing”, IEEE Transactions on Parallel and Distributed Systems, Volume 11, Issue 7, July 2000. pp. 729 - 738.

[20] S. Dutt, J. P. Hayes, “On Designing and Reconfiguring k-fault-tolerant Tree Architectures”, IEEE Transactions on Computers, Volume 39, Issue 4, April 1990. pp. 490 - 503.

[21] B. A. Izadi, Füsun Özgüner, “Two-Stage Fault-Tolerant k-ary Tree Multiprocessors”, Proceedings of the 2000 International Conference on Parallel and Distributed Processing and Applications, June 2000. pp. 527 - 533.

[22] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, R. Saleh, “Performance Evaluation and Design Trade-offs for Network on Chip Interconnect Architectures”, IEEE Transactions on Computers, vol. 54, no. 8, August 2005. pp. 1025 - 1040.

[23] J. Zhou, F. C. M. Lau, “Fault-tolerant Wormhole Routing in 2D Meshes”, International Symposium on Parallel Architectures, Algorithms and Networks, 2000. 7-9 Dec. 2000. pp. 94 - 101.

[24] J. Zhou, F. C. M. Lau, “Adaptive Fault-tolerant Wormhole Routing in 2D Meshes”, 15th

International Parallel and Distributed Processing Symposium., 23-27 April 2001.

[25] Jipeng Zhou, F. C. M. Lau, “Adaptive Fault-tolerant Wormhole Routing with Two Virtual Channels in 2D Meshes”, 7th International Symposium on Parallel Architectures, Algorithms and Networks, 2004. 10-12 May 2004. pp. 142 - 148.

[27] Jie Wu, “A Fault-tolerant and Deadlock-free Routing Protocol in 2D Meshes Based on Odd-even Turn Model”, IEEE Transactions on Computers, Volume 52, Issue 9, Sept. 2003. pp. 1154 - 1169.

[28] Jie Wu, Dajin Wang, “Fault-tolerant and Deadlock-free Routing in 2-D Meshes Using Rectilinear-monotone Polygonal Fault Blocks”, International Conference on Parallel Processing, 18-21 Aug. 2002. pp. 247 - 254.

[29] J. Duato, S. Yalamanchili, L. Ni, “Interconnection Networks – An Engineering Approach”, Morgan Kaufmann, 2002.

[30] http://cmp.imag.fr/index.php

[31] I. Koren and D.K. Pradhan, “Yield and Performance Enhancement through Redundancy in VLSI and WSI Multi-processor Systems”, Proc. of IEEE, Special Issue on Fault-Tolerance in VLSI, Vol. 74, No. 5. May 1986. pp. 699 - 711.

[32] M. L. Bushnell, and V. D. Agrawal, “Essentials of Electronic Testing for Digital, Memory & Mixed-Signal VLSI Circuits”, Springer, 2000.

[33] http://www.ecs.umass.edu/ece/koren/yield/

[34] Feihui Li, C. Nicopoulos, T. Richardson, Yuan Xie, V. Narayanan, M. Kandemir, “Design and Management of 3D Chip Multiprocessors Using Network-in-Memory”, 33rd International Symposium on Computer Architecture, 2006. ISCA '06. 2006. pp. 130 - 141

Appendix A

Publications

Following is a list of publications published in reputed conferences during the course of this research.

Conference:

1. Haibo Zhu, Partha Pratim Pande, and Cristian Grecu, "Performance Evaluation of Adaptive Routing Algorithms for achieving Fault Tolerance in NoC Fabrics", Accepted by 18th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP 2007), March 2007.

2. Partha Pratim Pande, Haibo Zhu, Amlan Ganguly, Cristian Grecu, "Crosstalk-aware Energy Reduction in NoC Communication Fabrics", IEEE International SOC Conference, SOCC 2006, 24th-27th September, 2006.

3. Partha Pratim Pande, Haibo Zhu, Amlan Ganguly, Cristian Grecu, "Energy Reduction through Crosstalk Avoidance Coding in NoC Paradigm", 9th Euromicro Conference on Digital System Design, DSD 2006, 30th August-1st September 2006.

Journal:

1. Partha Pratim Pande, Haibo Zhu, Amlan Ganguly, Cristian Grecu, "Energy Reduction through Crosstalk Avoidance Coding in Networks on Chip", Submitted to Journal of Systems Architecture, January 2007.