Institute of Computer Engineering and Computer Architecture
Stuttgart University, Pfaffenwaldring 47, 70569 Stuttgart

Master Thesis Nr. 3239

Fault Tolerant Routing Algorithm for Fully- and Partially-defetiveNoC Switches

Seyyed Mahdi Najmabadi

Study Program:	M.Sc. Information Technology(INFOTECH)
Examiner:	Prof. Dr. Hans-Joachim Wunderlich
Supervisor:	M.Sc. Atefe Dalirsani
Start Date:	1.9.2011
Submission Date:	1.3.2012
CR Classification:	B.4.3, B.4.4

Acknowledgements

This research project would not have been possible without the support of many people. The author wishes to express his gratitude to his supervisor, M.Sc. Atefe Dalirsani who was abundantly helpful and offered invaluable assistance, support and guidance. Deepest gratitude are also due to the members of the supervisory committee, Prof. Dr. Wunderlich without whose knowledge and assistance this study would not have been successful.
The author wishes to express his love and gratitude to his beloved families; for their understanding and endless love, through the duration of his studies.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Stuttgart, February 27, 2012 _____

Seyyed Mahdi Najmabadi

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation and Objectives

In the recent years, Network-on-chip (NoC) has emerged as a new communication infrastructure to decrease communication complexity of the current SoCs [11]. Like any other digital system, NoC may become defect either during production or after production.
The switches occupy bigger area on the NoC compared to the links. Therefore, the probability of having fault in the switches is more than the links. Consequently, dealing with the faulty switches is the main concern for designing fault-tolerant routing algorithms [2][3]. On the other hand, recently a new structural test method has been proposed [11] which can diagnose which port of defective switch is faulty, and guarantees the healthiness of the other ports. So there is no need to disable the defective switch. These two reasons, which are similar to each other from routing algorithm point of view, motivated us to modify the available algorithms in order to use the full capacity of the existing routing paths, and try to utilize all of the healthy ports and links to achieve higher performance, less congestion and more throughput. In this thesis, the capabilities of available fault tolerant routing algorithms will be examined to find out which routing algorithm is able to deal with fully and partially defective switches and defective links. The target is to develop an efficient deadlock and live-lock free fault tolerant routing algorithm with the following properties:

1. Designed for mesh topology NoCs,
2. Deal with defective switches,
3. Deal with partially defective switches,
4. Deal with faulty links efficiently.

1.2 Networks-on-Chip Concepts

In an NoC-based multi core system, processing elements are connected to each other through NoC communication network. The Network-on-Chip(NoC) consists of switches, links and network interfaces. Every processing element is connected to the network through network interfaces. In the design of the NoC the most essential decisions are choosing a network topology, switching technique, and a routing algorithm.

1.2.1 Network Topology

The shape of the wiring layout used to link switches is called the topology of the network. Lots of topologies have been proposed for NoCs so far, such as mesh, torus, star and octagon [27]. Several researchers have suggested that a mesh architecture for NoC will be more efficient in terms of latency, power consumption and ease of implementation, in comparison with the other topologies [28]. Due to the same reason, mesh topology has been used in this thesis.

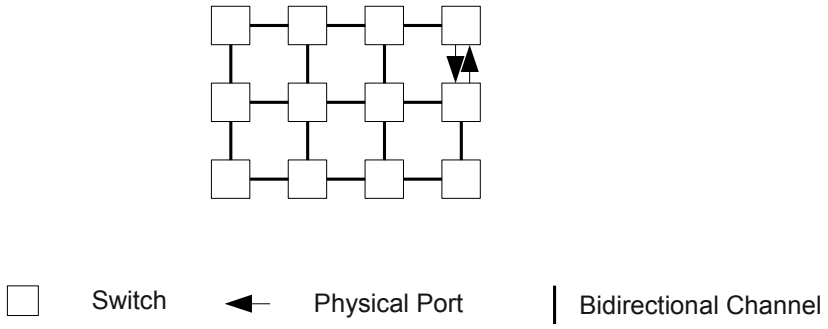

Figure 1.1: Example of 4 × 3 NoC mesh topology.

A 2-dimensional mesh network has m × n switches, where m and n are the radix of dimension. Two neighboring switches are connected by a bidirectional channel, which consists of two unidirectional physical ports. Figure 1.1 shows an example of 4 × 3 NoC mesh topology.

1.2.2 Switching Technique

The switching technique determines how and when the packet moves forward from a switch. The switching techniques are categorized into three classes, called store-and-forward switching, virtual cut-through switching and wormhole switching.

Store-and-Forward Switching:
Store-and-forward is the simplest switching technique. Packets are forwarded in one piece. The entire packet has to be stored in one switch buffer completely before it is forwarded to the next switch. So, the buffer has to be large enough to store the largest possible packets in the network. In this method an incoming packet is completely stored in one switch and then is forwarded to the next switch. Once the switch forwards a packet the communication can not be stopped [12].

Virtual Cut-Through Switching:
Virtual cut-through is an improved version of store-and-forward switching technique. Part of the packet which contains the addressing information is called header. A switch can begin to forward the packet to the next switch as soon as the addressing information has arrived and the next switch gives a permission. The header contains the required addressing information. This technique needs as much buffer as store-and-forward but forwarding can be started before the whole packet arrives, so it has less latency [12].

Wormhole Switching:
In wormhole switching, packets are divided into small and equal sized bit groups which are called flits (flow control digit or flow control unit). Likewise for virtual cut-through, forwarding begins as soon as the addressing information arrives. Addressing information is usually stored in the first flit of the packet which is called header flit. The routing path chosen for the header flit to be forwarded is reserved and the remaining flits of the packet are sent through the same path. The header flit is forwarded only if it gets the permission of the next switch. Once the permission is given the rest of flits must not wait for individual permission. Wormhole switching requires less memory than the two other switching techniques, because there is no need to store the entire packet into one switch. Due to the pipeline behavior of the wormhole switching, the latency is less than the other methods. But this behavior of wormhole switching makes it susceptible to deadlock because packets are allowed to hold as many resources as their flits are, while requesting other resources [12].
Due to high cost of on-chip buffers and pipeline behavior, wormhole switching is employed in many high-performance networks and almost all NoC architectures proposed up to now.

1.2.3 Routing Algorithm

On chip communication must be fast and reliable, but it must not be complicated and costly to implement. Considering on chip restrictions and application requirement, routing algorithm for network-on-chips have been introduced. There are many kind of different algorithms for different systems to choose. Every system has its

own requirements for the routing algorithm.
Routing is the process of selecting a possible path for packet transmission over a given network topology.
There are two main questions which must be answered before choosing a routing algorithm. The first question is the place *where* the routing decision must be made. There are two possibilities for the place where the decision can be made: The routing decision can be made at the source upon packet sending, and the routing information is stored in the packet header. Based on the destination address, the source determines the routing path (switch by switch) through which the packet traverses the network toward the destination. Therefore, no more routing decision has to be made at intermediate switches.
In another manner, the decision making is done in intermediate switches and each switch upon receiving packet can decide whether it should be delivered to the local processor or neighboring switches.
The second question in choosing a routing algorithm is *how* the routing decision must be made. In other words, how the routing algorithm must be implemented. Table-based routing and algorithmic routing are two main possibilities for implementing a routing algorithm.
In table-based routing, the destination address of the packet is used as an index in the table, and the outcome is the address of the next switch where the packet must be forwarded to. The routing table is used in every switch of the network to determine the best routing path. Although the decision making in table based approach is fast, there is a disadvantage for this approach. The main disadvantage is that as the size of the network is increased the table size also grows rapidly, consequently the switch area, energy and latency are increased [24].
On the other hand, in algorithmic routing, the decision about where to forward the packet is made by a routing algorithm. Destination address, network topology and in some advanced routing algorithms network traffic and congestion are the parameters which are considered to make the routing decision.
Routing algorithms can be also categorized into three classes, deterministic routing, adaptive routing and stochastic routing.
Deterministic routing algorithms always use the same routing path between certain sources and destinations. In other words, for every pair of source and destination, there is a unique routing path, which will never change. For instance XY routing algorithm is deterministic [21].
A routing algorithm is denoted adaptive if the routing between a certain source and destination changes according to the network state such as defective switches or traffic congestion. For example FADyAD is an adaptive routing algorithm [22].
A stochastic routing selects the path randomly (probabilistic) and is based on a coincidence and an assumption that every packet sooner or later reaches its destination [15]. Deflection routing which is presented in [19] is an example of stochastic routing algorithm.

Common Challenges of Routing Algorithms

The main challenges in the design of NoC routing algorithms are deadlock and live-lock.
Ensuring that a routing algorithm is deadlock-free is one of the major issues. Deadlock occurs when packets are allowed to hold some resources while requesting others. A common way to avoid deadlock is using virtual channel [17]. Because of the area overhead, it is not used in this work. But, in a more sophisticated way the routing algorithm itself can be designed so that a deadlock never happens. It will be described in details in chapter 3.
Live-lock occurs when a packet moves around its destination without ever reaching it. Live-lock may occur in stochastic routing algorithms [19]. The common solution to avoid live-lock is using a counter in the packet. The counter counts how many switches are traveled by the packet. When the counter reaches a predetermined value, the packet will be dropped from the network [16]. The main problem of this solution is that, for the dropped packet energy is consumed, but it doesn't reach the destination and must be resent. Besides, for non-adaptive rouitng algorithms, the packet does not find any alternative path to be routed to the destination. Therefore, the live-lock won't be avoided.

State-of-the-Art Routing Algorithms

Several routing algorithms that require no virtual channel have been proposed for mesh topology. Although virtual channels make the routing algorithm more flexible, this achievement is not for free due to the big area overhead which is required for the implementation of virtual channels.
XY routing is one of the most popular deterministic routing algorithms, due to its simplicity, and fast decision making. The algorithm routes the packet first in x dimension (row) until the packet reaches to the correct column where the destination is located, and then routes the packet in y dimension (column) until the packet reaches to its destination. The main problem of XY routing algorithm is that it provides no adaptiveness [23].
Turn model algorithm is another approach that avoids the deadlock by prohibiting the minimum number of turns. All turning limitations are required to prevent cyclic dependencies that can cause deadlock. For instance, west-first routing algorithm prevents all turns to the west. So, the packets which their destination is in the west side must be first forwarded as far as necessary to the west and then forwarded to the south or north. The main problem of these algorithms is that the adaptiveness that is provided is highly uneven [23]. The odd- even (OE) turn model [23] is a newer version of turn model algorithms, which classifies columns as either odd or even and prevents different turns in the odd and even columns, unlike the turn model which

prohibits certain turns in order to achieve deadlock free algorithm. The degree of routing adaptiveness provided by this method is more even [23].

State-of-the-Art Fault Tolerant Routing Algorithms

The routing algorithm which doesn't lost its functionality in presence of fault is called fault tolerant routing algorithm. In the network-on-chip, the faults may occur in the switches, the links and in the network interface. Faults in the network interface can be handled by processing element, and the fault tolerant routing algorithm is dealing with the faults in the switches and in the links. The task of a fault tolerant routing algorithm is to guarantee packet transmission between as much processing elements as possible, even in the NoC with a certain number of defective switches and links.
There has been significant works on fault tolerant routing algorithms for NoCs since 1995 [1]. Several fault tolerant routing algorithms have been developed which consider both faulty links and faulty switches. Some of these algorithms are implemented without virtual channels.
One of them is table-based algorithm such as the method presented in [4]. The routing algorithm which is presented in this work reconfigures network routing table in an offline process. Each switch contains a routing table, which lists an output port for each destination in the network. But due to area overhead, table-based algorithms generally are not recommended for the large NoCs.
One of the common solutions for fault tolerant routing algorithms is the region based algorithms, which have been widely proposed since the mid of 90s [1][5][6][7][8][9]. The main idea is considering a rectangular area around the faulty switches and faulty links and routes the packets around this region. In order to avoid deadlock, some especial turns are prohibited. The main drawback is that a lot of healthy switches have to be deactivated to guarantee deadlock free.
One of the region based fault tolerant routing algorithms that uses the odd-even turn model has been presented in [5]. The algorithm can provide high routing control, but it does not handle the defective switches in the network boundaries, and lots of healthy switches have to be deactivated.
In 2010, a new region based algorithm has been proposed, which reduces the number of deactivated healthy switches [10]. Although it still deactivates a number of healthy switches to achieve deadlock free, the main drawback is that a link failure is modeled as two faulty switches. This point motivated us to provide an efficient solution for faulty links. Another motivation for this thesis is that a new structural test method has been proposed [11] which can diagnose which port of the defective switch is faulty.

1.3 Thesis Organization

This thesis is composed of six chapters. The current chapter states the motivation, objectives and background on the subject. Chapter 2 discusses the proposed algorithm. In chapter 3 we prove that the algorithm is deadlock-free. Chapter 4 is intended to provide the experimental results and algorithm evaluation. And chapter 5 is the conclusion, where we will summarize our work within this thesis and mention the future work. The last part of this thesis comprises the reference.

Chapter 2

The Fault Tolerant Routing Algorithm

In this section, the fault tolerant routing algorithm is described in details. The proposed algorithm is an enhancement of the available region-based approach for NoCs [1]. The main problem of region-based algorithm is that the faulty link is represented as two faulty switches [8]. This solution deactivates two partial defective switches. On the other hand, a new structural method has been proposed recently[11] which can diagnose which port of the defective switch is faulty. In our algorithm a switch with one faulty port is not deactivated. In other words, we support partial-defective switches.
The main idea of the routing algorithm is to create a rectangular shape faulty region to achieve deadlock free and reduce routing difficulty. The rectangular shape faulty region is constructed by deactivating healthy switches based on the state of the neighboring switches in the configuration phase. The detail of switch deactivation will be discussed in section 2.2.

2.1 Notations and Assumptions

The faulty switches are detected in production phase, and with diagnosis, the faulty port of the faulty switch is distinguished [11]. This information is stored in the state registers in each switch and will be used in configuration phase later on.
A switch is called *faulty* if it is not diagnosable which port is faulty. A switch is called *semi-faulty* if it only has a single faulty port. According to these definitions, the faulty switches can not be considered as source, destination and intermediate switches. A semi-faulty switch might be source, destination, or intermediate switch, it might be deactivated too, it depends on the state of its neighboring switches, which is described in details.
The initial labeling is done based on the provided information by production test and diagnosis. The initial labeling will be changed during configuration phase to

the labels required for the routing: *ring, deactivated, faulty, unsafe, active, chain* and *s-chain*.
The routing algorithm is divided into two main phases, configuration phase and packet routing phase, which will be described separately.

2.2 Configuration Phase

2.2.1 Configuration

A configuration phase is required before routing algorithm able to route packets. The configuration phase is divided into six steps: 1) switch deactivation, 2) Ring construction, 3) Chain and s-chain construction, 4) Reference switch assignment, 5) Unsafe switches and 6) Highway concept. In the following these six steps are described.

1- Switch Deactivation

The switches which are labeled faulty remain faulty in the configuration phase. But the label of semi-faulty switches will be changed to either deactivated or unsafe based on the location. Some of the non-faulty, and semi-faulty switches are deactivated in this step, in order to construct rectangular shape faulty regions.
As it is shown in listing 2.1, a non-faulty switch is deactivated if there is at least one faulty or deactivated neighbor switch in both row and column or if its two neighbor switches in row are faulty [3], and a semi-faulty switch is deactivated if it has two or more faulty, deactivated or semi-faulty neighboring switches.

Listing 2.1: Switch deactivation procedure

```
While ( no more switch is deactivated)
{
for all n    // number of switches
  if (switch[n] has at least one faulty or deactivated neighbor
       in both column and row or switch[n] has two faulty or
      deactivated neighbor in row )
     switch[n] is deactivated

for all n
  if (switch[n] is semi-faulty and has more than one semi-
      faulty, faulty or deactivated neighbor)
     switch[n] is deactivated
}
```

Figure 2.1 shows the switch labeling after the deactivation step in a 7×7 NoC. According to the definition of faulty region, faulty, semi-faulty and deactivated switches must form a rectangular area which is surrounded by non-faulty switches. In this example, initial labeling says there are four faulty switches and five semi-faulty switches. Switch (3,1) and (2,2) are deactivated due to having two faulty neighbors. One of the semi-faulty switches, which is located at (5,5) is changed to the deactivated switch, because of having two semi-faulty neighbors. But switches (1,5) and (2,5) remain as semi-faulty switches due to having only one semi-faulty neighbor. Deactivation procedure must be repeated until the procedure does not find any switch to be deactivated.

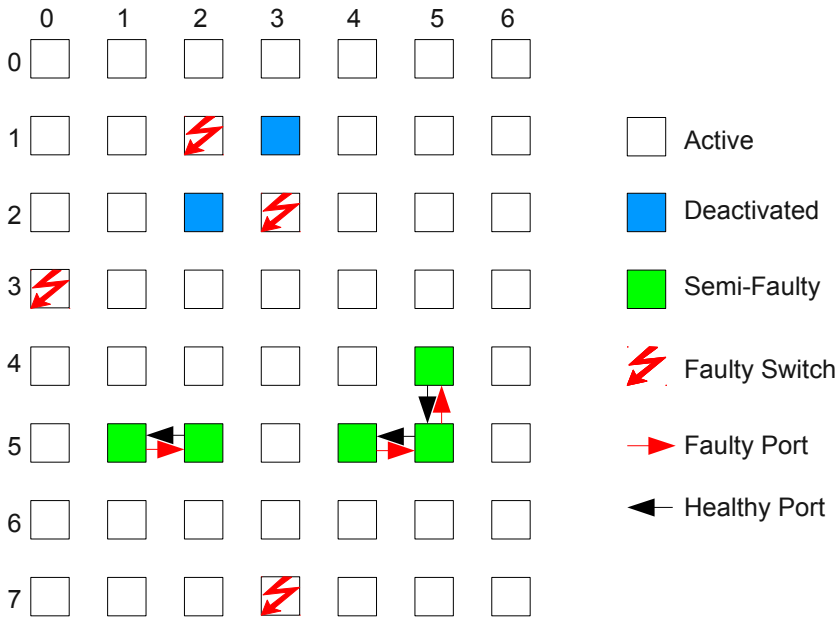

Figure 2.1: An example of a faulty network.

2- Ring Construction

The second step is to construct the rings. The switches surround the faulty region construct the ring. The switches in the ring are distinguished by two simple rules:

1. Each active switch which has a faulty or deactivate neighbor is a *ring* switch. Figure 2.2 shows that all active switches around the (2,1), (3,1), (2,2) and (3,2) are labeled as ring switches. Additionally the other four corner switches of the faulty region are considered as a ring switch. For instance in figure 2.2, the switches (1,0), (4,0), (4,3) and (1,3) are considered as ring switches.

2. Semi-faulty switches and their two out of four neighbors, which are orthogonal with the faulty port of this switch are the ring switches. As an example in figure 2.2, the semi-faulty switch (1,5) and its two neighbors (1,4) and (1,6) are labeled as ring switches and also the semi-faulty switch (2,5) and its two neighbors (2,4) and (2,6) are labeled as ring switches.

Figure 2.2 shows all of the ring switches in yellow boxes around the faulty regions. After execution of step 1, switch deactivation, and step 2, ring construction, no switch with semi-faulty label will remain. Some of them are labeled as deactivate in step 1 and the rest will be labeled as ring switch during step 2.
A ring switch may be in two different rings. in this case an overlap flag is set in

the switch. There is a different routing algorithm for this condition, which will be clarified in the packet routing section.

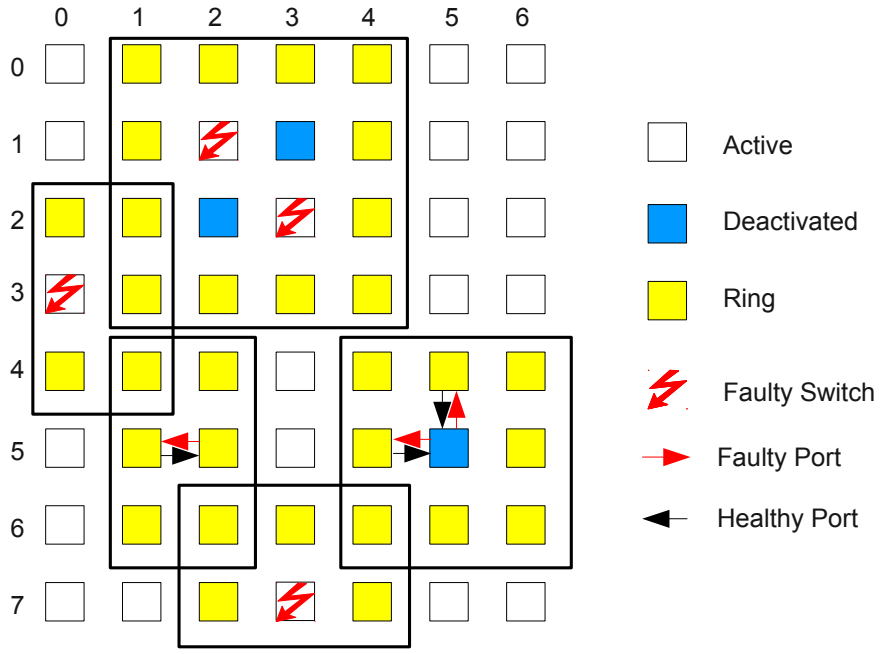

Figure 2.2: Ring construction around the faulty region.

3- Chain and S-chain Construction

Faults in boundaries are not supported in some region-based algorithms. To extend the routing algorithm for faults in the boundaries, two new switch labels called *chain* and *s-chain* must be added. The ring is called *chain* if the faulty region touches the west boundary of the mesh, and the ring is called *s-chain* if the faulty region touches the south boundary of the mesh.
Figure 2.3 demonstrates eight possible location for faulty regions which touch the NoC boundaries. According to our assumption switches around faulty regions 1 to 5 are labeled as ring switches. Faulty region 6 is surrounded by s-chain labeled switches and the label chain is used for the switches around faulty regions in location 7 and 8.

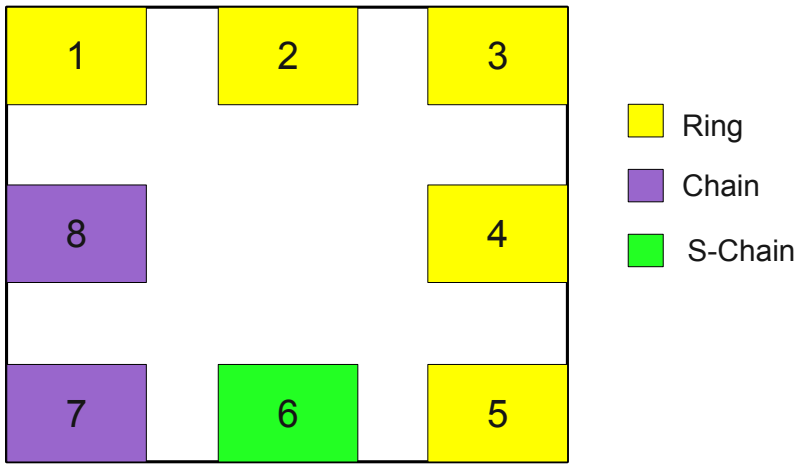

Figure 2.3: Possible position of the boundary ring, chain and s-chain.

As it is indicated in figure 2.4, there is a s-chain around the faulty switch (3,7), and there is a chain around the faulty switch (0,3).

4- Reference Switch Assignment

The switch in the northeast corner of the ring is called reference switch of the ring. The reference switch coordination must be stored in all of the ring switches. This information is needed during the packet routing. In figure 2.4, there are five faulty regions, which are distinguished with rectangular shapes, the switches (4,0), (1,2), (6,4), (4,6) and (2,3) are the five reference switches of the rings, chain, and s-chain. In an overlap ring switch like switch (1,4), two reference coordinations, (4,0) and (1,2) must be stored.

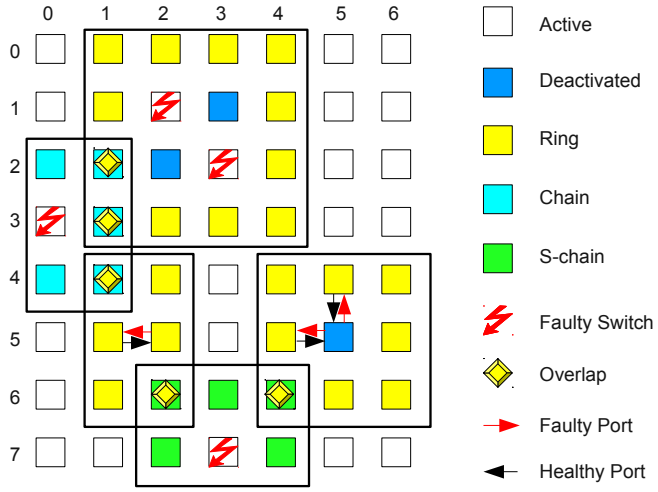

Figure 2.4: Ring creation around the faulty region.

5- Unsafe Switches

To reduce the number of deactivated switches, we define a new class of switches, called *unsafe* switches, for them the connected processing element can send and receive packets, but the switch itself can not be used as an intermediate switch. Deactivated switch is labeled as *unsafe* if it has at least one healthy neighboring switch in the west, south or east with healthy bidirectional communication link. There is only one exception that may be considered as an intermediate switch. The unsafe switch that is located in west-east highway can be considered as an intermediate switch.
For instance in figure 2.4, all the deactivated switches are changed to the unsafe switches. Switch (2,2) has a healthy neighboring switch in the west side, switch (3,1) has a healthy neighboring switch in the east side and switch (5,5) has a healthy neighboring switch in the east side.

6- Highway Concept

Decreasing the latency and having more balanced traffic is the motivation behind the highway idea. This idea takes the advantages of the ability to find the faulty port in a defective switch. The main idea of the highway is to route the packets through the faulty region. In a highway the packets are allowed only to move in one direction. Therefore destination address of the packet must be checked before entering to the highway. Figure 2.5 shows an example of routing a packet through the faulty region using a highway from west to east.

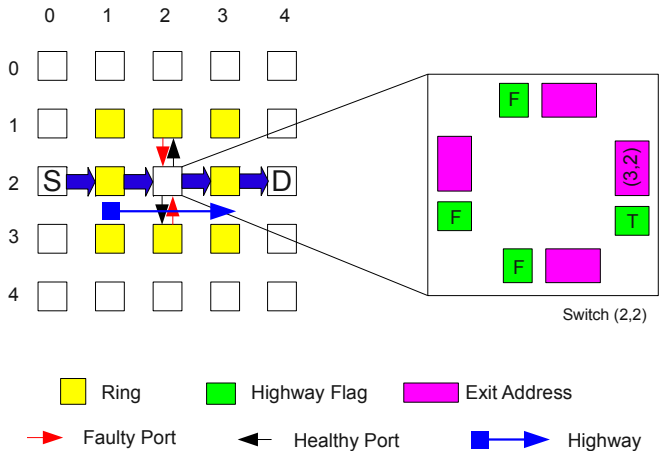

Figure 2.5: An example of the west-east highway.

In each ring switch, there are four highway flags and their corresponding exit addresses. The east highway flag is true if there is a healthy and direct path from the west side of the ring to the east side. The exit address is the address of the switch in the east boundary of the faulty region, where the packets exit from the highway. The same definition is used for the other three highway flags and addresses correspondingly.
For instance in figure 2.5, the south, west and north flags of the switch (2, 2) are set false, because there is no highway for these three directions. Since there is a healthy path from the west side to the east side of the faulty region, the highway flag in the east is set to true and its corresponding exit address is set to (3,2).
Three rules must be applied during highway construction to ensure the deadlock avoidance:

1. The condition for having the east-west and south-north highways is that: there must be no unsafe switch in the highway path. Figure 2.6 shows an example of east-west and south-north highways. The reason for this limitation will be discussed in the deadlock-free proof section.

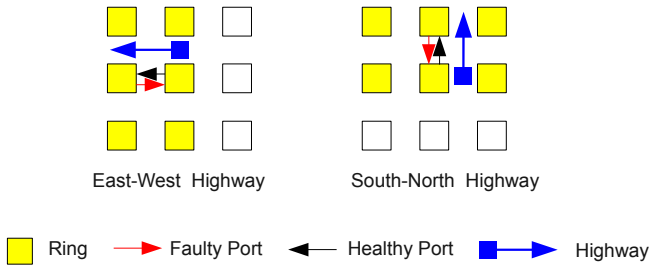

Figure 2.6: The two possibility of south-north and east-west highways.

2. The entrance switch of the highway can not be an overlapped ring switch (A ring switch which has more than one reference).
3. Only five turns (WS, SE, WN, SW and NE) are allowed for the packet enterance to the highways.

The reason behind these rules will be discussed in the deadlock-free proof section.

2.3 Packet Routing

After configuration phase is completed, the network can be used for the communication. In this section, we explain how the routing algorithm works and we start with packet structure.

2.3.1 Packet Header Structure

Figure 2.7 shows, the packet header which consists of source and destination addresses, packet type and reference fields.

Figure 2.7: Packet header fields.

Source address indicates the switch address where the packet is generated in processing element and is injected to the network. Destination address indicates the switch address where the packet is going to reach and be delivered to the processing

element.
The reference field saves the reference switch of the latest ring the packet has traversed. The reference field of the packet is used for the routing of the packets of type RO and where the switch has more than one reference switches (overlapped condition).
There are three types of the packets, row first RF, column first CF and row only RO. The usage of the packet type on routing algorithm will be described in the next section.

2.3.2 Routing Algorithm

The routing algorithm is distributed, and each switch after receiving the packets makes the routing decision according to the header information, the switch position and the switch label. Each switch changes the packet type when the packet is forwarded to the next switch. The packet type is set initially to the RF, and it may be changed dynamically in its path to the destination base on the current and destination coordinations. As it is shown in figure 2.8 the packet type is changed from RF to CF either when current switch is located in the same column as the destination or the destination switch is located in the east side of the current switch but in a different row. The packet is called RO if the current and the destination switches are located in the same row and the destination switch is located in the east side of the current switch.

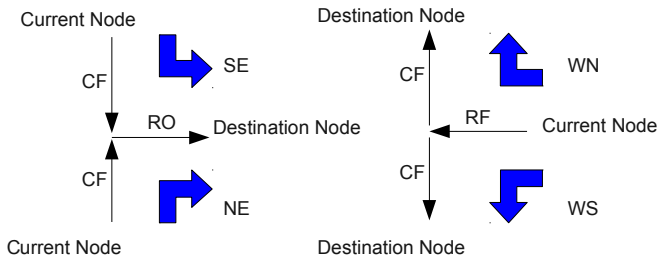

Figure 2.8: Packet type changing priority [6].

Once a packet type is changed to CF it can not be changed back to RF, and if a packet type is changed to RO, it will not be changed to any other types.
Figure 2.9 shows the packet type modification while the packets are moving from different sources to different destinations. Suppose that a packet is sent from S1 to destination D1. The packet starts as an RF packet since D1 is in the west of

S1, the packet type changes to CF packet when it reaches switch (2,2), which is on the same column. Now consider sending packet from source S2 to destination D2, packet type is CF since D2 is in the east of S2, when the packet reaches to switch (0,5) the packet type changes to the RO. The reason is that current switch and the destination switch are in the same row and the destination is located in the east side of the current switch. For the same reason the last packet which is traveling from S3 to the D3 has RO packet type.

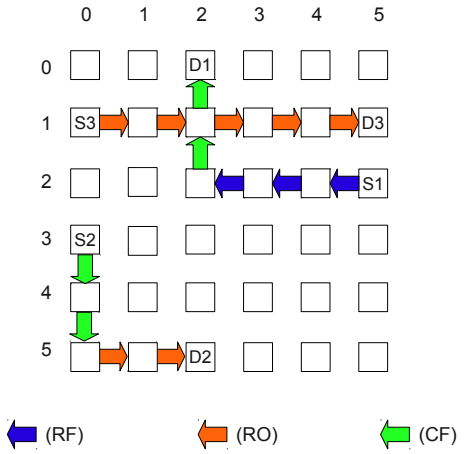

Figure 2.9: Example of packet type changing.

The routing algorithm for non-faulty regions has the same characteristic as XY routing algorithm, in both of them four turns out of eight turns are not made. As it is shown in figure 2.8 there exists only SE, NE, WN and WS turns in algorithm and in XY routing algorithm exists ES, EN, WN and WS.
In the faulty situations, the ring switches are able to navigate the packets around the faulty regions toward their destinations.
After packet type modification, the switch uses a decision tree to route the packet based on the routing algorithm rules. Figure 2.10 shows the decision tree for the routing algorithm according to the switch type and packet type.

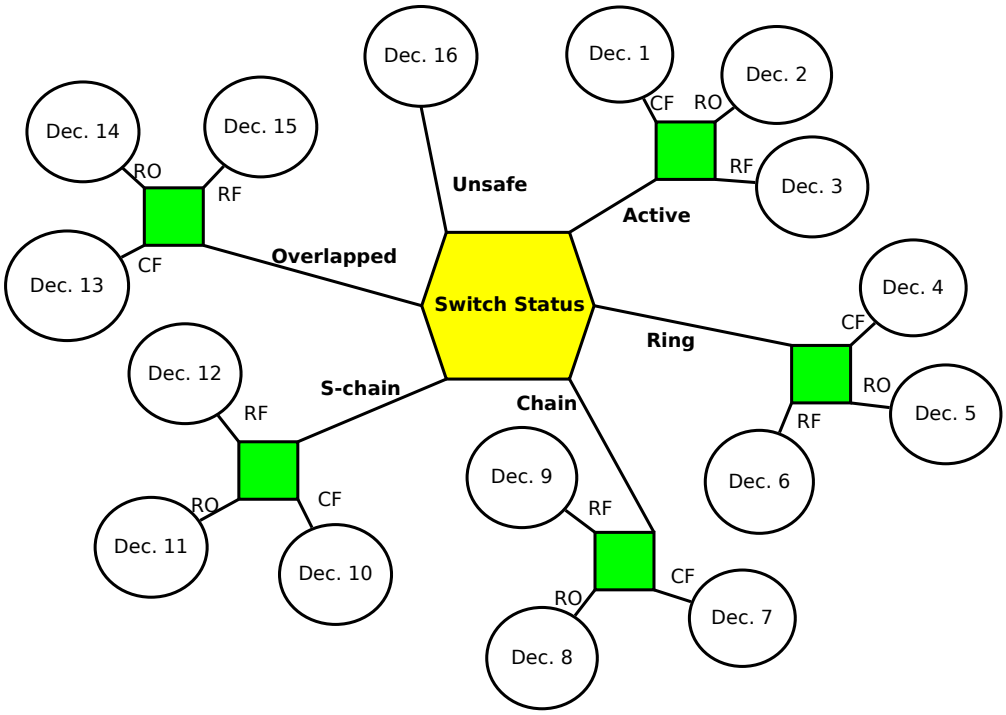

Figure 2.10: Routing algorithm decision tree.

As it is shown in figure 2.10, there are 16 different decisions that may occur during routing of a packet in each switch. If the destination address is equal to the current switch, the packet is delivered to the local processing element of that switch. In addition if the destination is in the neighbor of the current switch and there is a healthy link between them, the packet is forwarded directly to the destination regardless of the switch label. In the following, the packets which intend to move from north to south are called NS packets, and the packets that intend to move from south to north are called SN packets, C stands for the current switch, D stands for the destination switch and S stands for the source switch. In order to have a better understanding, the decisions are described by pseudo-codes.

2.3.3 Decisions of Active Switch

All healthy switches, except ring switches are labeled as an active switch. As it is shown in listing 2.2, decisions of active switch only depends on the packet type and the destination address.

Listing 2.2: Decisions of active switch

```
*********Decision 1, CF packet:
if (packet is NS)
  forward the packet to the south port;

esle if (it is SN packet)
  forward the packet to the north port;

*********Decision 2, RO packet:
The packet must be forwarded to the east port;

*********Decision 3, RF packet:
The packet must be forwarded to the west port;
```

2.3.4 Decisions of Ring Switch

Decision 4, CF packet:
The decision making in ring switches is based on the packet type, destination address and the position of the ring switch in the ring. Listing 2.3 shows the pseudo code for the CF packet type. CF packets can be divided also in two forms. The packets which intend to move from south to north (SN packets), and the packets that intend to move from north to south (NS packets). In the ring switches there are three different ways of routing the SN packets, forwarding to north port, routing clockwise and routing counterclockwise. It is the same for the SN packets but instead of forwarding the packets to north port the packets are forwarded to the south port. For instance, when the SN packet is in the north boundary of the ring, it will be routed to the north port, or when the NS packet is in the north boundary of the ring, it will be routed counterclockwise.

Listing 2.3: Decisions of ring switch for CF packets

```
Decision 4, CF packet:
if (packet is SN)
{
if (highway north direction flag is true and the north exit
    address is greater than destination.y)
  forward the packet to the north port;

if (C is on the north boundary of the ring)
  forward the packet to the north port;

else if (C is on the west boundary of the ring and D is in the
    same column as C)
  forward the packet to the north port;

else if (D is lower than reference switch of the ring and C is
    in the same column as D and C is in the west side of the
    reference switch and  D is not one column before the
    reference switch)
  route counterclockwise;

else if (D is lower than reference switch of the ring and D is
    in the next column of the C and D is in the west side of the
     reference switch and  D is not one column before the
    reference switch)
  route counterclockwise;

else if (the destination is lower than reference switch of the
    ring)
  route counterclockwise;

else
  route clockwise;
}
else if (packet is NS)
{
if (highway south direction flag is true and the south exit
    address is less than destination.y)
  forward the packet to the south port;

else if (C is on the east or south boundary of the ring)
  forward the packet to the south port;

else if (C is on the west boundary of the ring and the west
    port is available)
  forward the packet to the west port;

else
```

```
    forward it counterclockwise;
  }
```

Figure 2.11 shows an example of an NS packet which is traveling from north to the south part of the network. Since the source and destination switch are located in the same column address, the packet type is CF. Later on, when the packet reaches to the destination row, the packet type is changed to the RO.

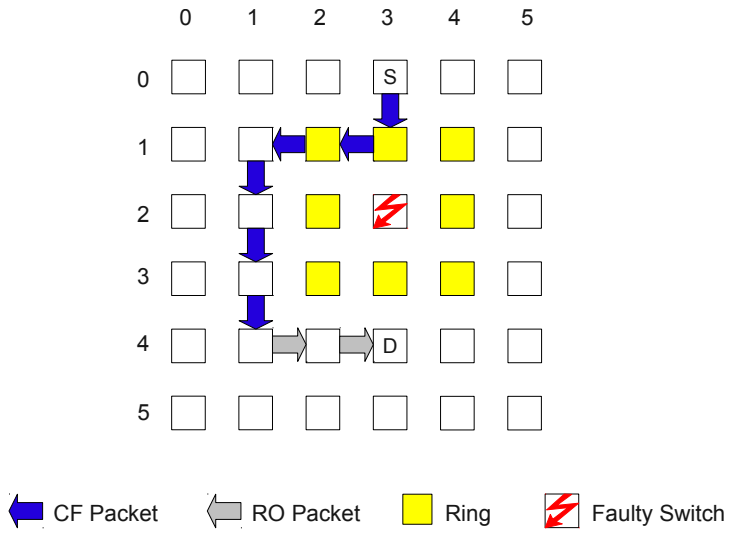

Figure 2.11: Example of an NS packet which is routed counterclockwise.

Figure 2.12 shows another example for CF packets which are traveling from south to north part of the network. As it is shown, when the destination address (D1) is higher than reference switch the packet is routed clockwise, but when the destination address (D2) is lower than reference switch the packet is routed counterclockwise.

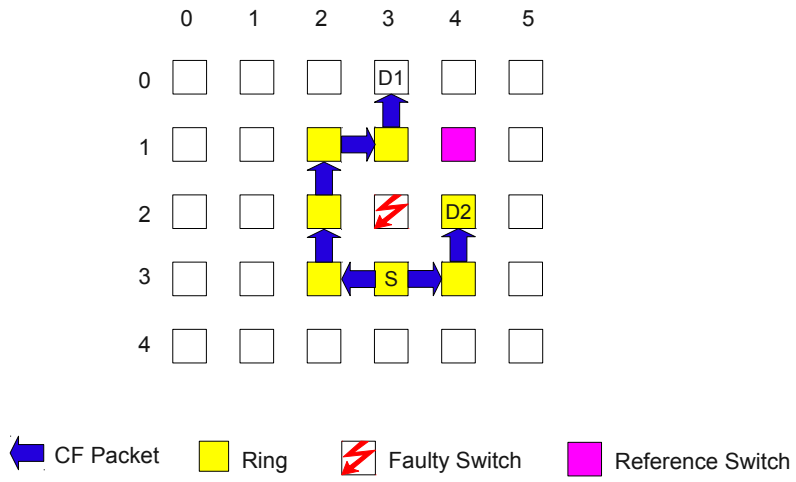

Figure 2.12: Example of SN packets which are routed counterclockwise and clockwise.

Decision 5, RO packet:
Listing 2.4 shows the pseudo code for the RO packet type. The decision making for RO packets in the ring switch is almost the same as active switches. The only difference is that when there is no possibility to forward the RO packets to the east port, the packets are routed counterclockwise.

Listing 2.4: Decisions of ring switch for RO packets

```
if (highway east direction flag is true and the east exit
    address is less than destination.x)
  forward the packet to the east port;

else if (C is in the same row as D and the east port is
    available)
       forward the packet to the east port;

else
       route counterclockwise;
```

Figure 2.13 shows an example of an RO packet. As already mentioned, the packet type is RO when the destination switch and current switch are in the same row, and the destination is located on the east side of current switch. Switch (1,2) is an active switch, so the packet is forwarded to the east port. Switch (2,2) is a ring switch and the east port is not available so the packet is routed counterclockwise. For the other switches in figure 2.13 the east port is available so the packet is routed to the east port.

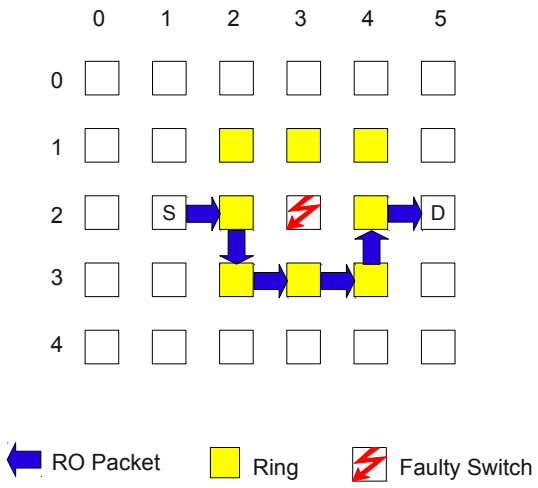

Figure 2.13: Example of RO packet.

Decision 6, RF packet:
Listing 2.5 shows the pseudo code for the RF packet type. The decision making for RF packets in the ring switch is almost the same as active switches. The only difference is that when there is no possibility to forward the RF packets to the west port, the packets are routed clockwise.

Listing 2.5: Decisions of ring switch for RF packets

```
if (highway west direction flag is true and the west exit
    address is greater than destination.x)
  forward the packet to the west port;

else if (the west port is valid)
  forward the packet to the west port;

else
  route clockwise;
```

Figure 2.14 shows an example of an RF packet. As already mentioned, the packet type is RF when the destination is located on the west side of current switch. Switch (5,2) is an active switch, so the packet is forwarded to the east port. Switch (4,2) is a ring switch and the west port is not available so the packet is routed clockwise. For the other switches in figure 2.14 the west port is available so the packet is routed to the west port of each switch.

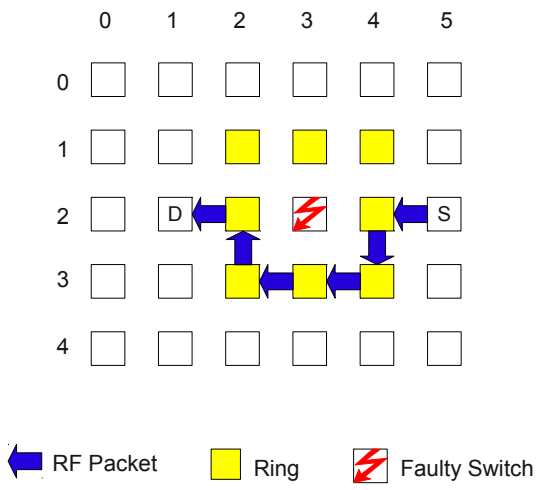

Figure 2.14: Example of RF packet.

2.3.5 Decisions of Chain Switch

As already mentioned in section 2.2, the ring is called chain if the faulty region touches the west boundary of the mesh. Since there is no possibility of routing packet in the west side of the chain, the routing algorithm is much simpler than a normal ring. The listing 2.6 shows how the routing is performed in chain switches for three different packet types.

Listing 2.6: Decisions of chain switch

```
*********Decision 7, CF packet:
if (packet is NS )
{
if (south port is available and D is not in the west of C)
  forward the packet to the south port;

else
  route clockwise;
}
else if (packet is SN)
{
if (north port is available and D is not in the west of C)
  forward the packet to the north port;

else
  route counterclockwise;
}
```

```
*********Decision 8, RO packet:
if (C is in the same row as D and the east port is available)
  forward the packet the east port;

else
  route counterclockwise;

*********Decision 9, RF packet:
if (C is in the same row as D and the east port is available)
  forward the packet to the west port;

else if (D is higher than C)
  route counterclockwise;

else
  route clockwise;
```

2.3.6 Decisions of S-Chain Switch

As already mentioned in section 2.2, the ring is called s-chain if the faulty region touches the south boundary of the mesh. The listing 2.7 shows how the routing is performed in s-chain switches for three different packet types. Basically, an RF packet is always routed along west port as long as the west port is available, otherwise it is forwarded counterclockwise. For an RO packet if current switch is in the same row as destination switch and the east port is available the packed is forwarded to the east port, otherwise it is routed clockwise. When the packet type is CF, for an NS packet, it is routed clockwise and for SN packet, it is routed counterclockwise.

Listing 2.7: Decisions of s-chain switch

```
*********Decision 10, CF packet:
if (packet is NS)
{
if (C and D is in the west boundary of s-chain)
  forward the packet to the south port;

else
  route clockwise;
}
else if (packet is SN)
{
if (C is in the north or east boundary of s-chain and the north
    port is available)
  forward the packet to the north port;
```

```
else if (C is in the west boundary of s-chain and the west port
     is available)
  forward the packet to the west port;

else
  route counterclockwise;
}
*********Decision 11, RO packet:
if (C is in the same row as destination switch and the east
    port is available)
  forward the packet to the east port;

else
  route clockwise;

*********Decision 12, RF packet:
if (the west port is available)
  forward the packet to the west port;

else
  route counterclockwise;
```

2.3.7 Decisions of Overlap Situtation

Decision 13, CF packet:
The overlap may occur between two or three rings. Each ring has its own reference switch. The reference switches coordinates is compared, and if the packet is a SN packet then the ring that its reference switch is more northwards is selected, and if it is NS packet then the ring that its reference switch is more southwards is selected, and after ring selection it is the same as decision 4 in the selected ring. There is only one exception for the NS packet, if the packet is NS packets and C is on the west boundary of the one of the rings (because of the overlap) and the west port is available then the packet is forwarded to the west.

Decision 14, RO packet:
Overlap situation in the RO packet type is more complicated than the other packet types, and only in this case the reference field of the packet header is required. If the current switch is in the same row as the destination switch and current switch is in the north east or south east corner of the previous ring then the ring that its reference switch is more eastward is selected otherwise the ring is not changed, and the ring that its reference switch is equal to the reference (already saved in to the packet header) is selected, and after ring selection, decision 5 in the selected ring will be taken.
Figure 2.15 shows two examples for two different cases. The switch (3,3) is the

first overlapped switch, and it is located in the south east of the left ring. For the blue packet, the current switch and the destination (D1) are in the same row so the ring with the leftmost reference switch is selected, but the green packet selects the ring which its reference switch is the same as reference field, which in this case the reference field is (3,1).

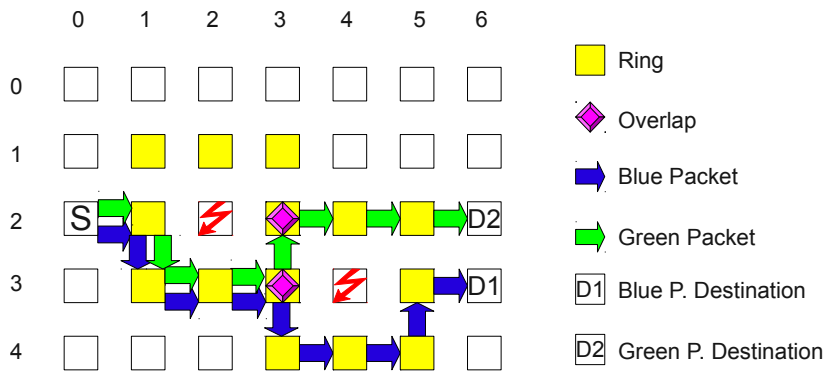

Figure 2.15: An example for decision 14 with RO packet.

Decision 15, RF packet:
The reference switches coordinate are compared, and the ring that its reference switch is more westwards is selected, and then the rules of decision 6 in the selected ring will be applied.

2.3.8 Decision of Unsafe Switch

Decision 16:
In the unsafe switch, no matter what the packet type is, the packet is forwarded to the healthy neighboring switch, via the healthy port.

Chapter 3

Proof of Deadlock-Free

Deadlock may occur if the packets have the permission to hold some resources while requesting for the other resources. In other words, if two or more than two packets are waiting for each other resources in a cycle then deadlock occurs. The only way to be relieved of the deadlock situation is that at least one packet releases the resources. In the wormhole switching method the resources are the buffers of the channels in which the packets are stored. Figure 3.1 shows an example of a deadlock situation. In this example, four packets are waiting for each other's resources in a cycle. Each packet is represented with a color and the arrows show the direction of the packet to be routed. The bold arrow indicates the resources which are currently occupied by the packet while the dotted arrow shows the resources which are requested by the packet. The red packet (to be routed from C to A) holds channel 3 and is requesting channel 2 which is hold by yellow packet. In the same time the yellow packet is requesting channel 1 which is hold by blue packet, while the blue packet requests channel 4 which is hold by the green packet. Finally, the green packet requests channel 3 which is hold by the red packet. In a cycle, all the packets hold a channel and requesting another channel which is hold by a packet in the waiting cycle. This is a deadlock situation. To avoid deadlock situations, the algorithm must route the packets in a way that these kind of circular wait never happens.

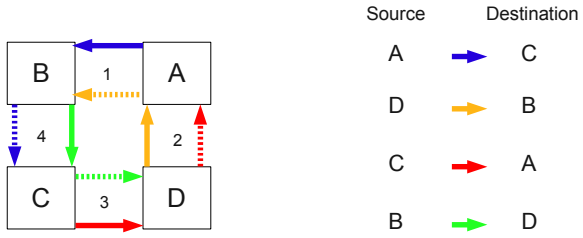

Figure 3.1: Example of a deadlock situation involving four packets.

To prove that the algorithm is deadlock-free, we use the alignment theorem [1], explained in 3.1, and the theorem of channel dependency graph [14], explained in 3.2.

3.1 Alignment Theorem

Each channel is located between two neighboring switches. A turn consists of a horizontal channel and a vertical channel. The channels have one switch in common such that the source switch of one channel is the destination switch of the other; the common switch of these two channels is called the turning switch of the turn[1].
As it is shown in figure 3.2, in the mesh architecture eight turns are possible. A turn is called ES turn if the direction changes from east to south. In other words, an ES turn consists of two channels, the first one to the east and the second one to the south. According to the direction of the channels in the turn, rest of the turns are called EN (form east to north), WN (from west to north), WS (from west to south), SE (from south to east), NE (form north to east), NW (from north to west) and SW (from south to west). The routing algorithm may prohibit some of the turns to avoid deadlock. For instance, one of the consequence of the packet type is occurrence of the four turns (WS, SE, WN and NE) which are allowed in all switches. The other four turns may happen only in the ring, s-chain and chain switches.

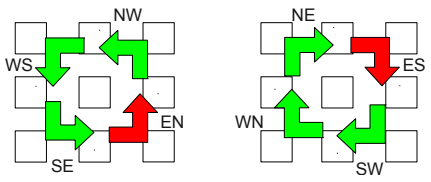

Figure 3.2: The eight available turns in mesh topology.

Allignment: Two turns are *aligned* if the turning switches of them are in the same column, and there exists a set of packets, {P1, P2, .., Pi} such that: when P1 takes the first turn, Pi takes the second turn, and there exists routing possibility through the switches between two turns so that for all $1 \leq j \leq i$, packet Pj waits for Pj+1 resources. For instance in figure 3.3, two turns EN, and NW respectively in switches (4,5) and (4,1) are aligned, because they are in the same column and the blue arrows show the possible routing path between these two turns.

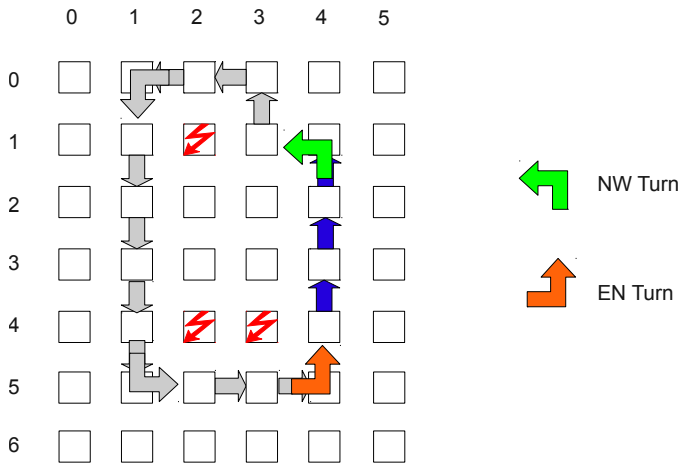

Figure 3.3: Example of two turns alignment.

Theorem 1: In the deadlock situation, at least one ES turn is aligned with one SW turn, or at least one EN turn is aligned with one NW turn [1].

Proof: The deadlock situation may happen when a set of packets are in the waiting cycle (clockwise or counterclockwise like figure 3.3). The waiting cycle includes both horizontal and vertical channels. Considering the right most column of a counterclockwise waiting cycle (figure 3.3), it is observed that an EN turn is aligned with a NW turn. For instance in figure 3.3, column 4 is the right most column of the waiting cycle, and the NW turn at switch (4,1) is aligned with the EN turn at switch (4,5). In the right most column of a clockwise waiting cycle, an ES turn is aligned with a SW turn.
To avoid deadlock, such a cycle must not exist. Therefore, if the algorithm guarantees that the ES turn never is aligned with the SW turn (in a clockwise cycle) the cycle is never made. In addition, to avoid the counterclockwise cycles it is sufficient to show that an EN turn is never aligned with an NW turn. Consequently,the deadlock is avoided.

3.2 Channel Dependency Graph

Definition 3: A channel dependency graph [14], D, for a given interconnection network, I, and routing function, R, is a directed graph, $D=G(C,E)$. The vertices of D are the channels of I. The edges of D, are the pairs of channels connected by R:

$$E = \{(c_i, c_j) \| R(c_i, n) = c_j \quad for \quad some \quad n \quad \exists \quad N\}$$

I	interconnection network, a directed graph I=$G(N,C)$
N	the set of switches
C	the set of channels
R	a routing function R: $C \times N \rightarrow C$
D	the channel dependency graph
E	the edges of D
n_i	a switch
c_i	a channel

Table 3.1: Summary of notation

Figure 3.4 shows an interconnected network and its corresponding dependency graphs. The channels of the network are the nodes of the channel dependency graph. The interconnected network in figure 3.4 has 8 channels, as a result the corresponding channel dependency graph has also 8 nodes.

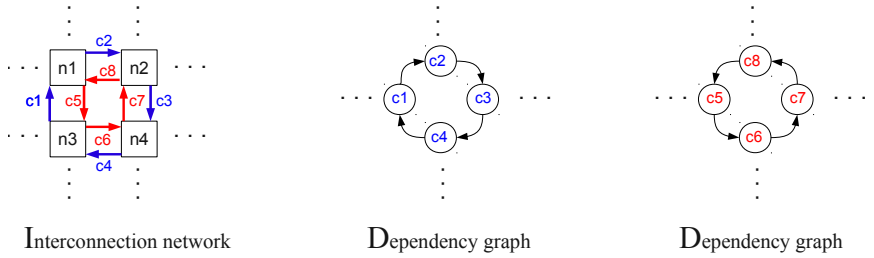

Figure 3.4: Channel dependencies in a bidirectional interconnected network.

Theorem 2: A routing function, R, for an interconnection network, I, is deadlock-free iff there are no cycles in the channel dependency graph, D [14].

3.3 Deadlock-free Proof for Non-Faulty Regions

Based on the packet type rules the ES and EN turns are never happened in a non-faulty network. As a result, according to the theorem 1, the algorithm is deadlock-

free for non-faulty region.

3.4 Deadlock-free Proof for Non-Boundary Faulty Regions

As mentioned earlier, when a fault occurs in the network, a ring around the faulty switches or links is constructed. The basic idea of the routing algorithm in faulty cases is to turn the packets around the faulty region using the ring. The faulty region may contain either one or more faulty switches, or just a faulty link. Around a faulty region, we can imagine four areas in the north, west, south and east. In each of the areas, current switch or destination switch may exist. Figure 3.5 shows four areas next to each other. There are twelve combinations of placing current and destination switches around the faulty region. As an example, the current switch may be on the west side of the faulty region and the destination on the east side of the faulty region. This combination is presented by an arrow between the current and the destination location (west $\rightarrow$ east). Based on the packet type rule, four of these combinations which are north $\rightarrow$ west, south $\rightarrow$ west, west $\rightarrow$ north and west $\rightarrow$ south will never meet the faulty region. But there are eight other cases that may meet the faulty region. Figure 3.5 shows the eight possible routing paths which meet the faulty region. Packets in the path 1, path 2, path 3 and path 4 are routed counterclockwise around the faulty region. The other four cases are routed clockwise around the faulty region. The corresponding decision for the packets in path 1, path 2 and path 4 are decision 4, and the corresponding decision for packets in path 3 is decision 5 which forwards the packet counterclockwise. According to decision 6, all the packets which their current address is on the east side of the ring and during traveling meet that area(i.e., path 5, path 6 and path 7), are forwarded clockwise. Finally, (south$\rightarrow$ north) packets (i.e, path 8) are routed clockwise since their destination address are upper than the reference switch (decision 4).

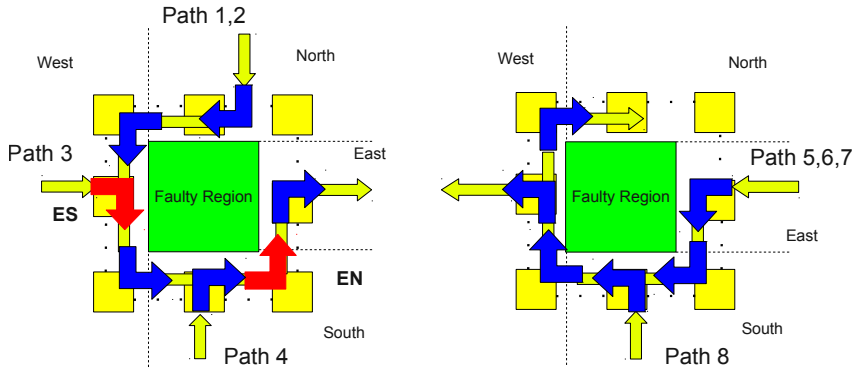

Figure 3.5: path 1(north→ east), path 2(north→ south), path 3(west→ east), path 4(south→ east), path 5(east→ west), path 6(east→ north), path 7(east→ south), path 8(south→ north)

Because of routing rule around the faulty region, all eight turns occur and the deadlock may happen. In order to prove the algorithm is deadlock-free, according to the theorem 1, we need to prove that ES turn is not aligned with SW turn and EN turn is not aligned with NW turn. As it is shown in figure 3.5, the ES turn can only be made in the west side of the ring, and EN turn can only be made in the southeast corner of the ring. In the following these two cases are analyzed and we will show that the critical turns are never aligned with each other and the algorithm is deadlock free.

3.4.1 Case 1: EN Turn

As presented in the previous section, an EN turn can be made only in the southeast corner switch of the ring. In figure 3.6, the ring switch in which the EN turn may happen is shown by an east-north green arrow.

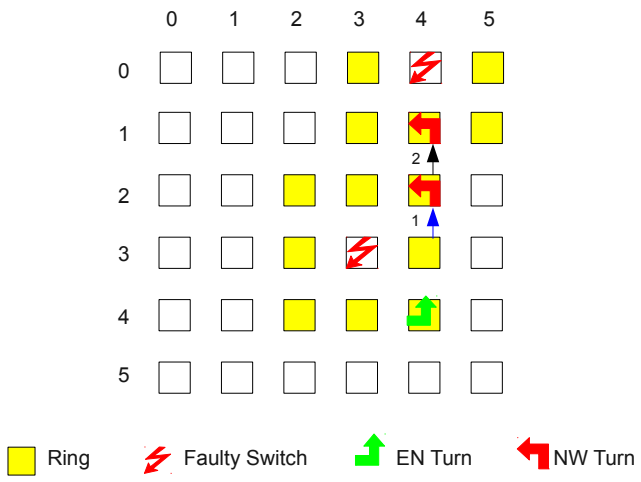

Figure 3.6: Example of an EN turn on the southeast corner switch of the ring.

In order to have a deadlock-free algorithm, we need to show that EN turn is never aligned with a possible NW turn in the same column. First we must show that the NW turn never occurs in the right most column of the same ring. For example, our algorithm must not allow any NW turn in switches (4,3) and (4,2) in the ring of figure 3.6. The NW turn in switch (4,3) will route the packet into the faulty region, which is not a correct routing decision. According to figure 3.5 there is no possibility to have a turn in the switch (4,2), the main reason is that path 5, path 6 and path 7 are always routed clockwise (decision 6) and the path 8 is also routed clockwise because the destination is higher than the reference switch (4,2) (decision 4).
Another position that NW turn may occur is higher than the reference switch of the current ring, for example switch (4,1) in figure 3.6. Although this turn and EN turn are in the same column (4), we will prove that there is no possible routing path between these two turns; Therefore, they are not aligned. In order to have a routing path between EN turn in switch (4,4) and NW turn in switch (4,1), there must be a possible routing path between channel 1 (blue arrow) and channel 2 (black arrow) presented in figure 3.6. In other words, if a packet uses channel 1 to be routed in the north direction, it will then choose channel 2 to continue its path toward the north. Now, we will show that the case will never happen. The reason that there is no connection between channel 1 and channel 2 is that any packet traveling from south to north will never use the east boundary of a ring except if its destination is located lower than the reference switch of the ring (decision 4). Therefore, in spite of having EN and NW turns on the same column, there exist no possible routing path which connects these two turns. Therefore, they are not aligned. Consequently, there will not cause a deadlock.
Since the small ring is a new idea in this work, for the sake of completeness we have

investigated the situation for the smaller rings too. In the example of figure 3.7 the same situation has been shown with a smaller ring. In this case, we can use the same explanation and prove that this case is also deadlock-free.

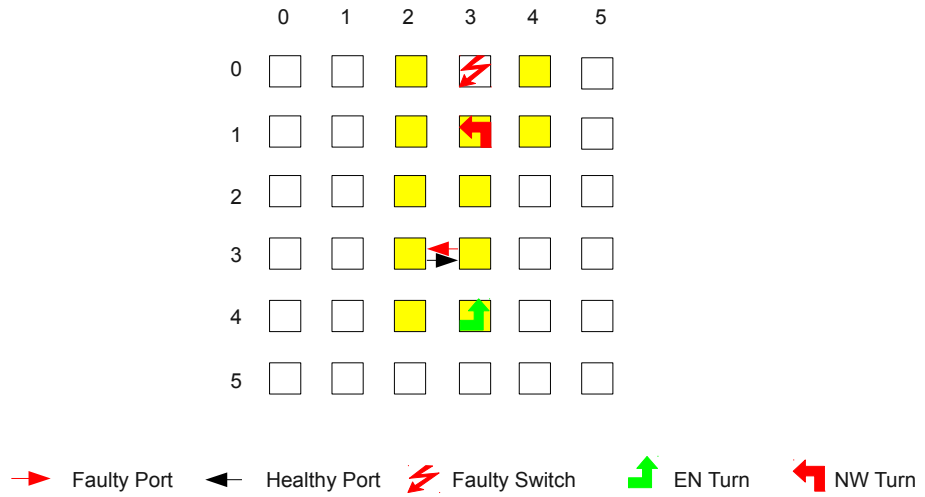

Figure 3.7: Faulty network for a vertical small ring.

Figure 3.8 shows another possible example with the smaller rings. The point is that there is no possibility of EN turn in the south east corner of the small horizontal ring. The reason is that, EN turn will be only made if the east side of the ring consists of at least three ring switches. But in a small horizontal ring, the east boundary of the ring consists of two switches.

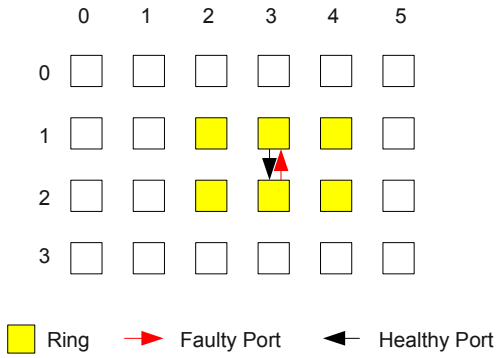

Figure 3.8: Faulty network for a horizontal small ring.

3.4.2 Case 2: ES Turn

As already mentioned the only place which ES turn may occur is on the west boundary of the ring. In figure 3.9, the ring switch in which the ES turn may happen is shown by an east-south green arrow.

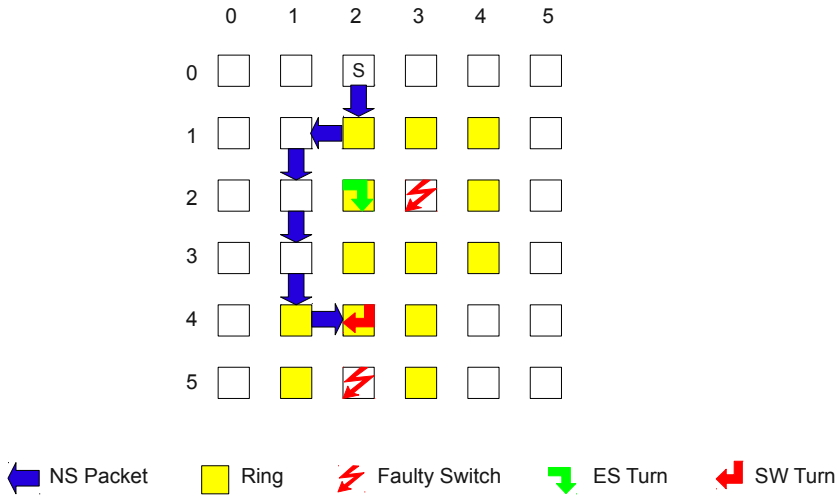

Figure 3.9: Example of ES turn in the west side of the ring.

In order to have a deadlock-free algorithm, we need to show that ES turn is never aligned with SW turn.
The first possible case that SW turn may happen is in the west side of the same ring, the same as switch (2,3) in figure 3.9. But as it is shown in figure 3.5 there is no SW turn in the west side of the ring, and the SW turn only occurs in the north or southeast of the ring.
The second possible case that the SW turn occurs outside the ring is in the north side of another ring. Figure 3.9 shows an example of this situation. Although the SW turn (2,4) and ES turn (2,2) are in the same column, we will show that they are not aligned, and these two turns cannot be the cause of deadlock. The reason is that, the NS packets (figure 3.9) which travel from north to south and their destination is under the south boundary of the ring are not allowed to use the west boundary of the ring, and they are routed one step toward the west as long as a channel is available (decision 4), so there is no possible path between these two turns. The packets which their destination is on the east side of the ring are routed counterclockwise, and only these packets are allowed to use west boundary ring switches.
Accordingly, all of the SW turns under the south boundary of the ring cannot be the cause of deadlock.
The same proof can be used for the smaller rings too. Figure 3.10 shows a vertical

small ring in which ES turn at the switch (2,2) is possible. Since the same rules for packet routing is used for the smaller rings, the same proof can be used to explain the situation of figure 3.10 is deadlock-free

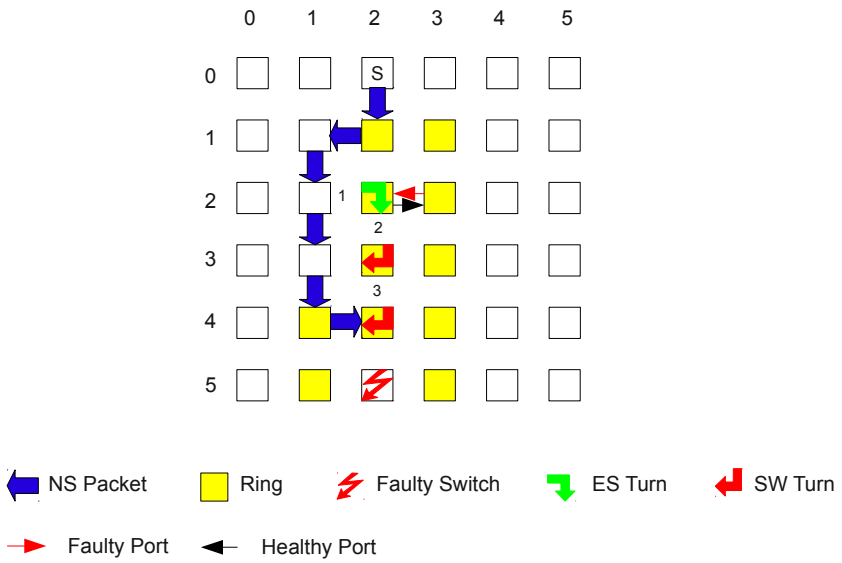

Figure 3.10: An example of vertical small ring.

Since the small ring is a new idea in this work, for the sake of completeness another situation of the small ring is shown in figure 3.11.
If the RO packet faces to the faulty or deactivated switches in the west boundary of ring ES turn will be made. Since there is no faulty or deactivated switch in the west boundary of the small vertical ring, there is no possibility of ES turn. In order to have consistent algorithm for the rings the NS packet are not allowed to use west boundary of the ring.

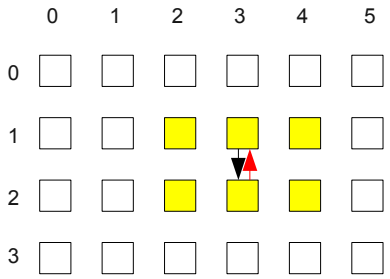

Figure 3.11: An example of horizontal small ring.

3.5 Deadlock-free Proof for Boundary Faulty Region

As it was already discussed in the configuration phase, if there is a faulty switch in the west boundary of the network, special rules must be applied for packet routing. The ring around the faulty switch in the west boundary is called chain. Also, having a faulty switch in the south boundary of the network enforces defining a set of rules for the ring switches around it. The ring is called s-chain. Figure 3.12 and figure 3.13 show these two faulty regions. As it is shown in figure 3.12 the ES turn can be made in northeast of chain and s-chain, and EN turn can be made in the southeast corner of the chain. Since the chain and the s-chain are in the boundary of network the and the boundary switches are faulty, no cycle can be formed in their channel dependency graph. For instance in figure 3.12 the channel 4, 8 and 11 are not connected to any other nodes of the channel dependency graph. As a result, according to the theorem 2, these two turns can not be the cause of deadlock.

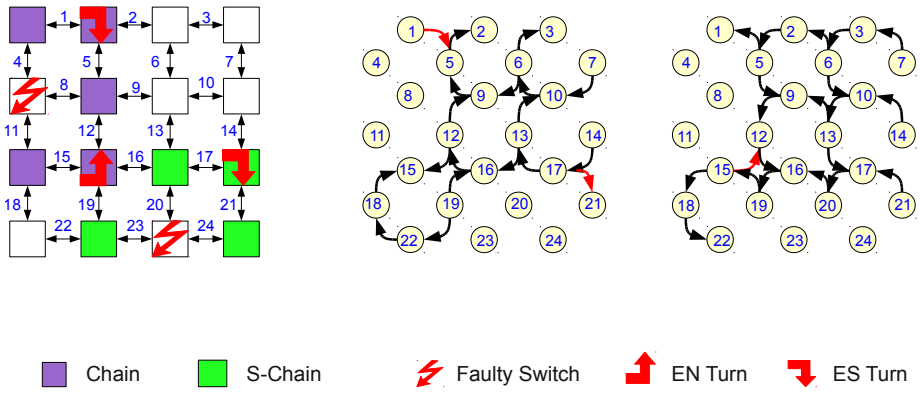

Figure 3.12: Deadlock free proof for chain and s-chain, there is no cycle in the channel dependency graph.

Figure 3.13 shows all possible paths around the s-chain switches. Path 1 shows the routing path of the packet which is traveling from east side of the network and meets the faulty region. Since the packet type is RF the packet is routed counterclockwise, and the corresponding rule is decision 12.
Path 2 shows the routing path of the packet which is traveling from west side of the network and its destination is in the east side of faulty region. Since the packet type is RO the packet is routed clockwise, and the corresponding rule is decision 11.
Path 3 shows the routing path of the packet which is traveling from north side of

the network and meets the faulty region, and its destination is in the east side of faulty region. Since the packet type is CF the packet is routed clockwise, and the corresponding rule is decision 10.

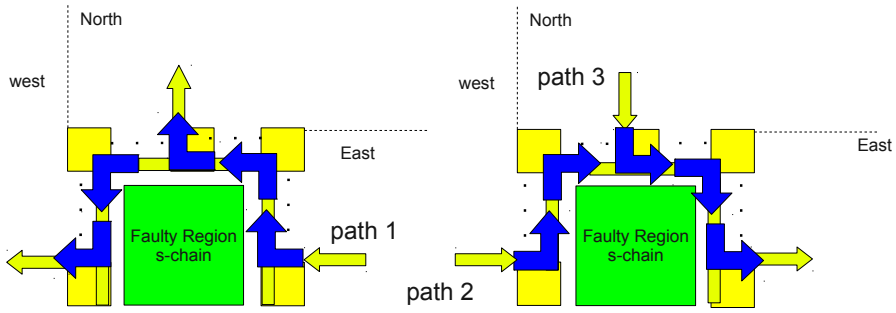

Figure 3.13: All possible paths around the s-chain switches.

As it is shown in figure 3.13 an EN turn can be made in the west side of the s-chain. As an example, figure 3.14 shows the s-chain switch (2,3) which the EN turn may happen. If we can prove that EN turns in the west side of the s-chain are not aligned with NW then according to theorem one, this case is deadlock-free. As it is shown in figure 3.13 there is no NW in the west side of the s-chain. So the first possible position for the NW turn is upper than the north side of s-chain switch (2,1). Although these two turns are in the same column, there is no possible routing path between these two turns. The reason is that, according to the decision 10, the CF packets which are in the west side of the s-chain are always forwarded one step to the west.

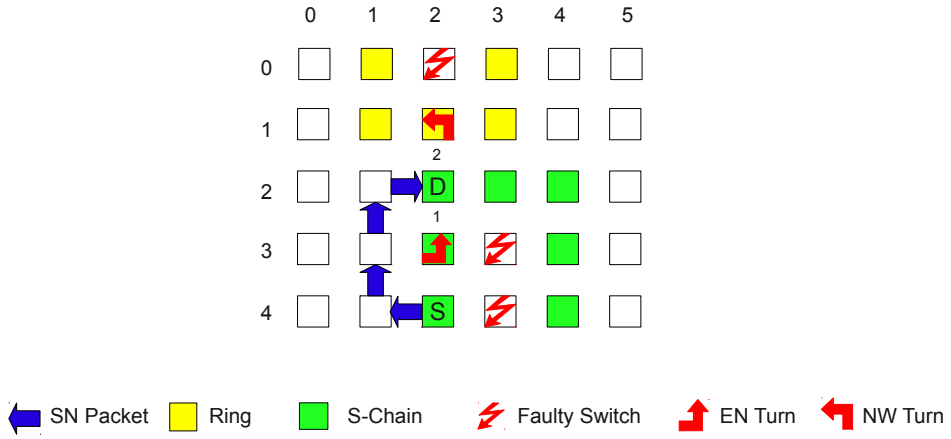

Figure 3.14: EN turn in the west side of the s-chain.

3.6 Deadlock-free Proof for Highway

3.6.1 Highway Without Unsafe Switches

Figure 3.15 shows possible path around the ring the same as figure 3.5 and the highways which may exist. As shown earlier in section 2.2, only five turns (WS, SE, WN, SW and NE) are allowed for the packet entrance to the highways. As mentioned in configuration phase, the highway may contain unsafe switches. In this section, we will discuss both cases and prove in any case the algorithm is deadlock-free. Firstly, let's assume that there is no unsafe switch in the highway.

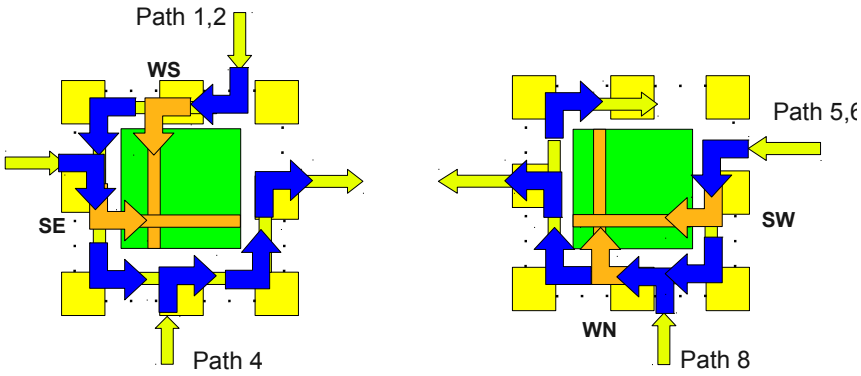

Figure 3.15: The four possible entrance to the highway and corresponding turns, WS, SE, WN and SW.

According to the alignment theorem, two turns SW and ES turns, and also EN and NW turns must not be aligned, in order to prove that the algorithm is deadlock-free. Among the four turns, highways create only SW turns, which must be proven that will never be aligned with ES turns.
The ES turn can be made in the west side of the ring but the SW can only be made in the east side of the ring in case of a highway. Therefore, the only possible case that these two turns are in the same column, is having two rings. This situation is shown in figure 3.16.

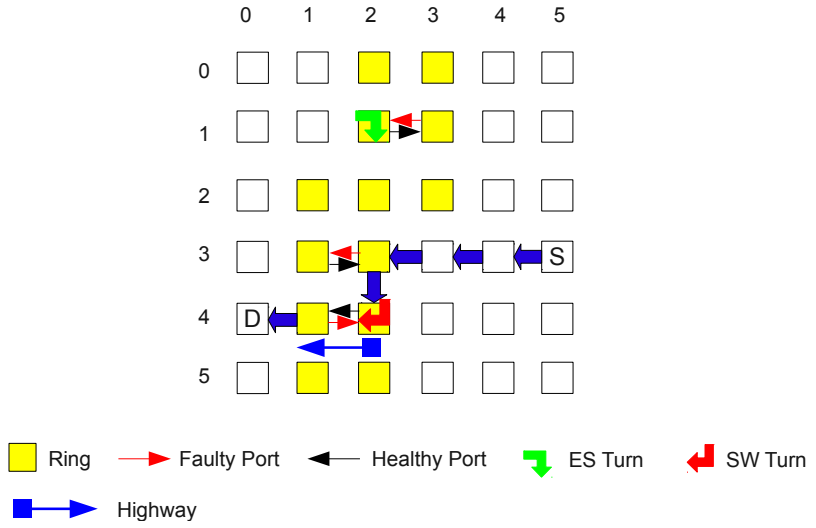

Figure 3.16: SW turn in the east side of the ring in case of highway.

In this example, the packet is forwarded through the highway, and the SW turn is made in the east side of the lower ring. Although the ES and SW turns are in the same column, it has been already proven in section 3.4.2 that there is no possible routing path between an ES turn in the upper ring and an SW turn which is made lower than the south side of the upper ring. As a result these two turns are not aligned and this situation is deadlock-free.

3.6.2 Highway With Unsafe Switches

As mentioned in section 2.2, the west-east highway is the only highway that supports unsafe switches. Looking into figure 3.17, we observe that using west-east highway may create SE and NE turns which are not one of the turns considered in alignment theory. Therefore, using highway with unsafe switches will never cause deadlock. The reason is that, SE and NE turns are the only possible turns for accessing this highway (figure 3.17), and based on the alignment theorem we need to only consider SW, ES, EN and NW turns, so these two turns are allowed to be used in each switch. Hence, the deadlock can never happen.

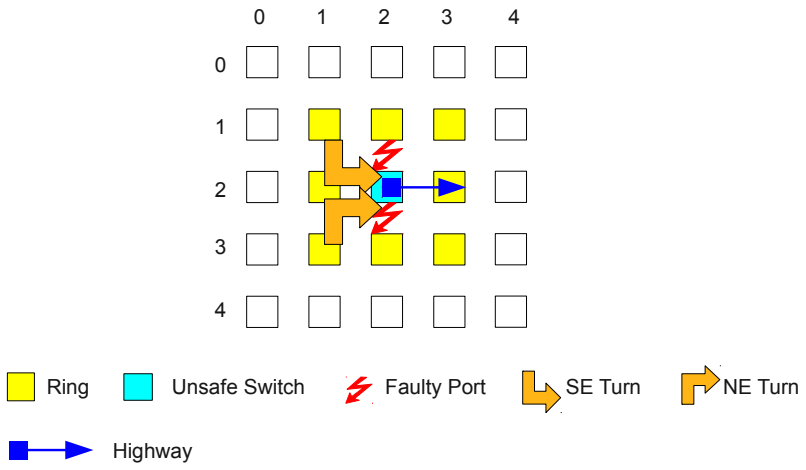

Figure 3.17: Highway with unsafe switch.

Chapter 4

Simulation and Results

In this chapter, we present the experimental studies to evaluate the performance of our algorithm in a defective NoCs. The quality of algorithm in terms of throughput, latency and drop ratio in faulty cases is investigated, and compared with the state-of-the-art region-based fault tolerant routing algorithm [7]. The simulation platform for experimental results in this work is an open source systemC based NoC simulator, which has been modified to implement our routing algorithm.

4.1 NoC Simulator

NoC architecture consists of many switches which are running in parallel, and interacting concurrently. Since evaluating the performance of our algorithm in real chip is complicated and time consuming, systemC based molding is preferred. The systemC library accelerates simulation by using interface method calls (IMC) to implement communication between hardware components [25].
Among available open source simulators the noxim network on chip simulator is selected in this study. Noxim is an academic open source NoC simulator that takes advantages of systemC library and developed at university of Catania (Italy) [26].
Noxim structure consists of switch module and a top level module that connects the switches in a mesh form. Each switch module has two routers and processing element sub-modules. The routing algorithm is performed in router module and processing element handles the network traffic. Noxim has a command line user interface, in which the network size, input buffer size, packet size, packet injection rate and traffic pattern can be customize by the user.
Total number of received packets/flits, average throughput and max/min average delay are the main monitoring parameters of the Noxim. The intermediate transactions can be logged to monitor the packets transmission switch by switch.
Since Noxim was not designed for faulty network, fault injection feature, and related evaluation metric such as drop ratio was not provided. We have modified the simulator in order to support faulty network features.

4.2 Simulation Setup

This section describes how simulation is performed and what are the conditions that is considered in our evaluation.

We evaluated our algorithm on 2D mesh network. The network size during simulation is fixed to 12×12 tiles. The input port of each switch has a FIFO size of 4 flits and we used uniform traffic model to simulate the performance of the network. Under the uniform traffic model a PE sends a 4-flit packet to other PE with equal probability. Simulation time is 6000 cycles and 1000 cycles as warm-up time. The warm-up time means that the statistic results are considered after 1000 simulation cycles in order to have more realistic evaluation. In non-faulty network each node can generates 85 packets in 6000 simulation cycles without any packet loss.

We have simulated seven faulty conditions for the 12×12 network by injecting 1, 3, 5, 7, 10, 15 ,20 faults in the entire network. Simulation of each faulty condition has been repeated 100 times, in each of them the faults are located randomly in some of the network switches. To have a realistic distribution among the faults which corrupt only one switch port (single port deactivation) , and the faults which corrupt the entire switch operation (switch deactivation) the statistics provided by [11] is used. Accordingly, 60% of the faults lead to port deactivation and the rest 40% have switch deactivation effect. The result are averaged over 100 simulations for each faulty condition.

4.2.1 Evaluation Metrics

The performance metric used in evaluation are drop ratio, average latency and throughput. which are defined as follows:

Drop ratio: Drop ratio shows the rate of unsuccessful packet deliveries, which is defined by the following formula [11]:

$$drop\ \ ratio = \frac{\#injected\ \ packets - \#received\ \ packets}{\#injected\ \ packets} \times 100\% \qquad (1)$$

In a network with live-lock and deadlock free routing algorithm, injected packets are delivered to the destination when there exist enough resources in the network. Upon packet injection by the sender processing elements, available network resources (i.e., buffer space for packet storage in the connected switch) are considered. When the resources are not enough for the packet injection, PE cannot inject the packet. According to the application the packet may be dropped or wait until some resources get released. In our simulations, we evaluate the network metrics under certain

loads. When due to the non-enough resource packet injection must be postponed, it is considered as a drop. Therefore, drop ratio parameter implicitly indicates to the cases where resources are not enough to achieve certain load in the network.

Packet Latency: Packet latency is defined for each packet as the time between packet injection by the sender and packet delivery at the destination, and it is commonly measured in clock cycle (simulation cycle). The latency has direct relation with the network load ($load_{net}$). Latency of a packet is less when the network load is lower. The minimum time between packet injection and delivery can be counted by the switch delays and the number of switches in the routing path. However, due to network traffic, the packet has to wait in some switches to get required resources. In this thesis, packet latency is calculated by the following formula:

$$latency = T_{receive} - T_{send} \qquad (2)$$

In which T_{send} is the time that the header flit of the packet leaves the source PE, and $T_{receive}$ is the time that the tail flit of the packet reaches the destination PE.

Throughput: Throughput refers to how many packets can be transferred and received by destination switch in network in a given amount of time. It is used to measure the performance of NoCs. Normalized throughput is obtained by the following formula:

$$throughput = \frac{\#received\ \ packets}{simulation\ \ cycle\ \ \times\ \ network\ \ size} \qquad (3)$$

Saturation Throughput: Saturation throughput is the maximum throughput that the network can produce, and the throughput cannot be more than saturation throughput [12].

4.3 Experiments

4.3.1 Experiment 1: Connectivity

The purpose of the connectivity test is to be assure that all processing elements, which are connected to non-faulty and non-deactivated switches, can communicate with each other.
For this purpose, we have simulated 7 faulty conditions and each faulty condition has been repeated 100 times in different location. To check pair to pair connection, each processing element sends a packet to all other PEs, (e.g., in a 12×12 non-faulty network, each PE sends packets to the other 143 PEs).

We have monitored drop ratio in each condition. The results show 0% drop ratio. It means that, all PEs, connected to the non-faulty and non-deactivated switches, can communicate with each other. In other words, there exists a routing path among each pair of PEs in the network using the routing algorithm presented in this thesis.

4.3.2 Experiment 2: Δt_{min}

In our evaluation each processing element (PE) is sending packet to any other PE with the same time interval. The minimum time interval required between two consecutive packet injections is called Δt_{min} [11]. If PE injects the first packet at time $t = t_1$, and the second packet at time $t = t_2$, then $t_2 - t_1 \geq \Delta t_{min}$ otherwise the packet is dropped in PE. Minimum time interval varies based on the network size, buffer size, traffic model, packet size and other network switch parameters. In order to find Δt_{min}, we have reduced the time interval and monitor the drop ratio in a non-faulty network.

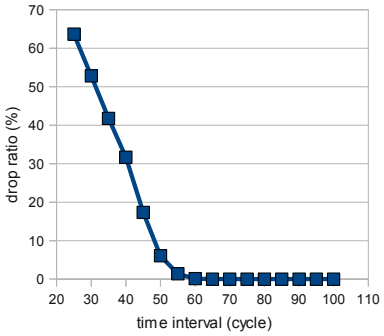

Figure 4.1: Δt_{min} for uniform traffic model.

As it is shown in figure 4.1, for uniform traffic, when the time interval is less than 60 cycles, the packets are dropped in the non-faulty network. So the Δt_{min} for uniform traffic model is 60 cycles.
By having the minimum time interval, the load per PE is defined as:

$$load_{PE} = \frac{\Delta t_{min}}{\Delta t_{avg}} \qquad (4)$$

Where Δt_{avg} is the average time between two consecutive packet injection and is greater than Δt_{min} [11].

$load_{net}$ has a direct relation with $load_{PE}$ and the number of the PEs that are able to inject packet. The network load is defined as:

$$load_{net} = load_{PE} \times \frac{active \;\; PEs}{network \;\; size} \qquad (5)$$

In a non-faulty network $load_{net}$ is equal with $load_{PE}$ since all PEs contribute in the network traffic. But in the faulty network $load_{net}$ is less than $load_{PE}$.

4.3.3 Experiment 3: Saturation Throughput

In order to achieve the saturation throughput we increase the $load_{PE}$ by decreasing the Δt_{avg}. As it is shown in figure 4.2, increasing the packet injection rate doesn't lead to further increase of network throughput. The red and yellow part of the figure show the situations where the Δt_{avg} is less than Δt_{min} and we have packet loss in these situations.

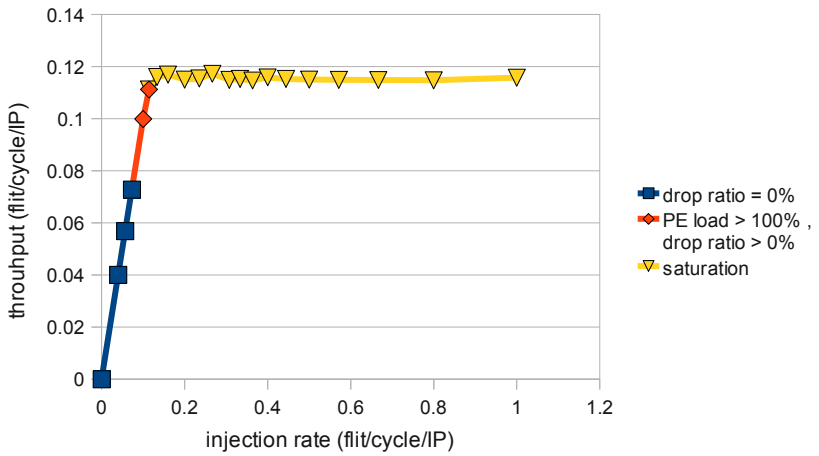

Figure 4.2: Saturation throughput in non-faulty network.

4.3.4 Experiment 4: Performance Comparison

To demonstrate the effectiveness of our fault tolerant routing algorithm, we compare it to a state-of-the-art fault tolerant routing algorithm [7] which use the same network architecture and switching technique.

Variable Load

In the first experiment, both routing algorithms are computed in presence of 5 injected faults in the network. In our simulation the traffic model is uniform. $Load_{PE}$ is the same in two networks but $load_{net}$ depends on the available PEs. Figure 4.3 (a) shows that in reference algorithm ten switches are deactivated, but in proposed algorithm only two switches are deactivated.
Figure 4.3 (b) shows that when the network load is under 40 percents the throughput of the two algorithms are the same. Drop ratio is zero percent and all generated packets reach their destinations. But when the network load is increased the proposed algorithm has higher throughput due to the less drop ratio. The reason is that in the reference algorithm more switches are deactivated so the available routing paths (network resources) are less than the proposed algorithm. With the same reason, as it can be seen from the figure 4.3 (c) the proposed algorithm has less average latency than the reference algorithm. Figure 4.3 (d) shows that both algorithms have no drop ratio in low loads. And the reason is that the algorithms are fault tolerant and all generated packets reach their destinations, but in higher loads, since reference algorithm has less network resources, more packets are dropped.

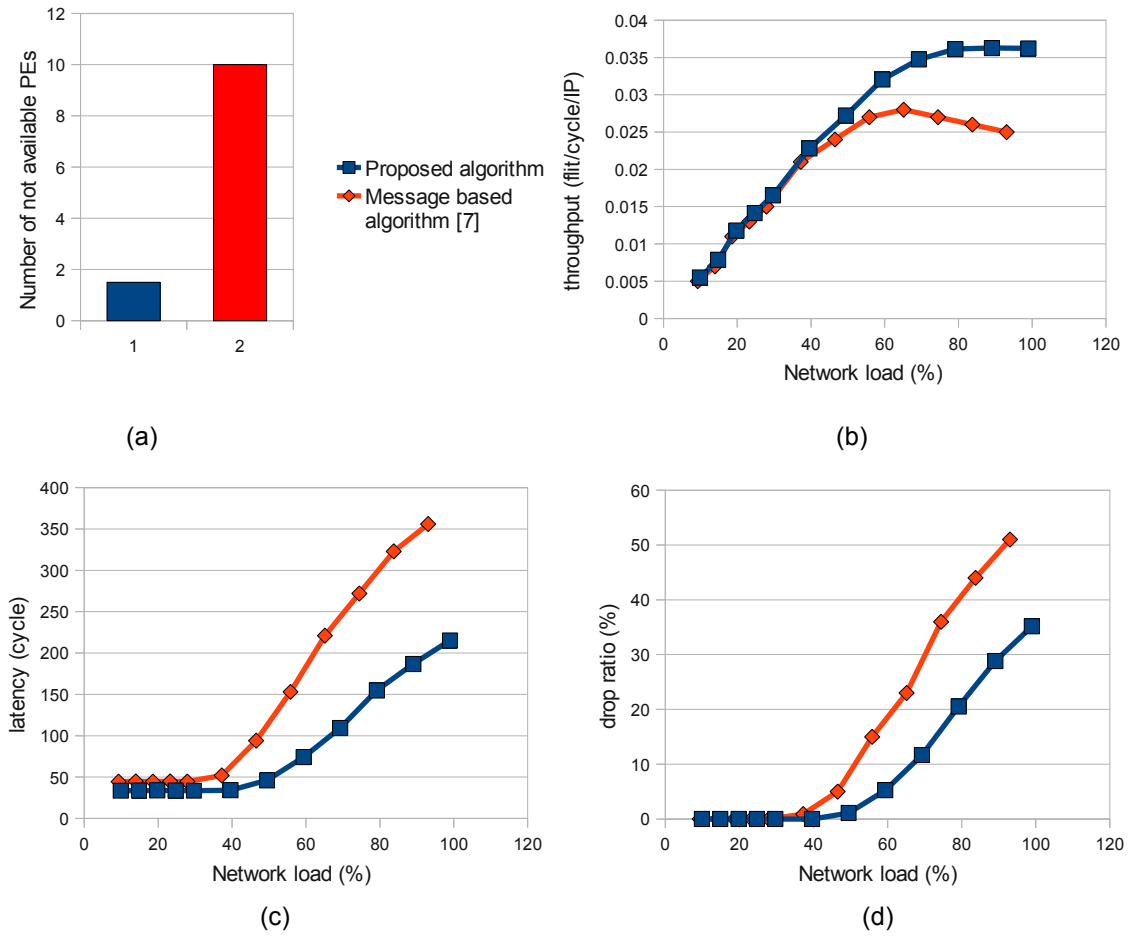

Figure 4.3: Performance comparison of the algorithms under the uniform traffic when the fault counts are 5, and the network size is 12×12 (a) Number of not available processing elements (b) Average throughput (c) Average packet latency (d) Average drop ratio.

To further compare the proposed algorithm with the reference algorithm, the number of faults are increased to 10 faults.

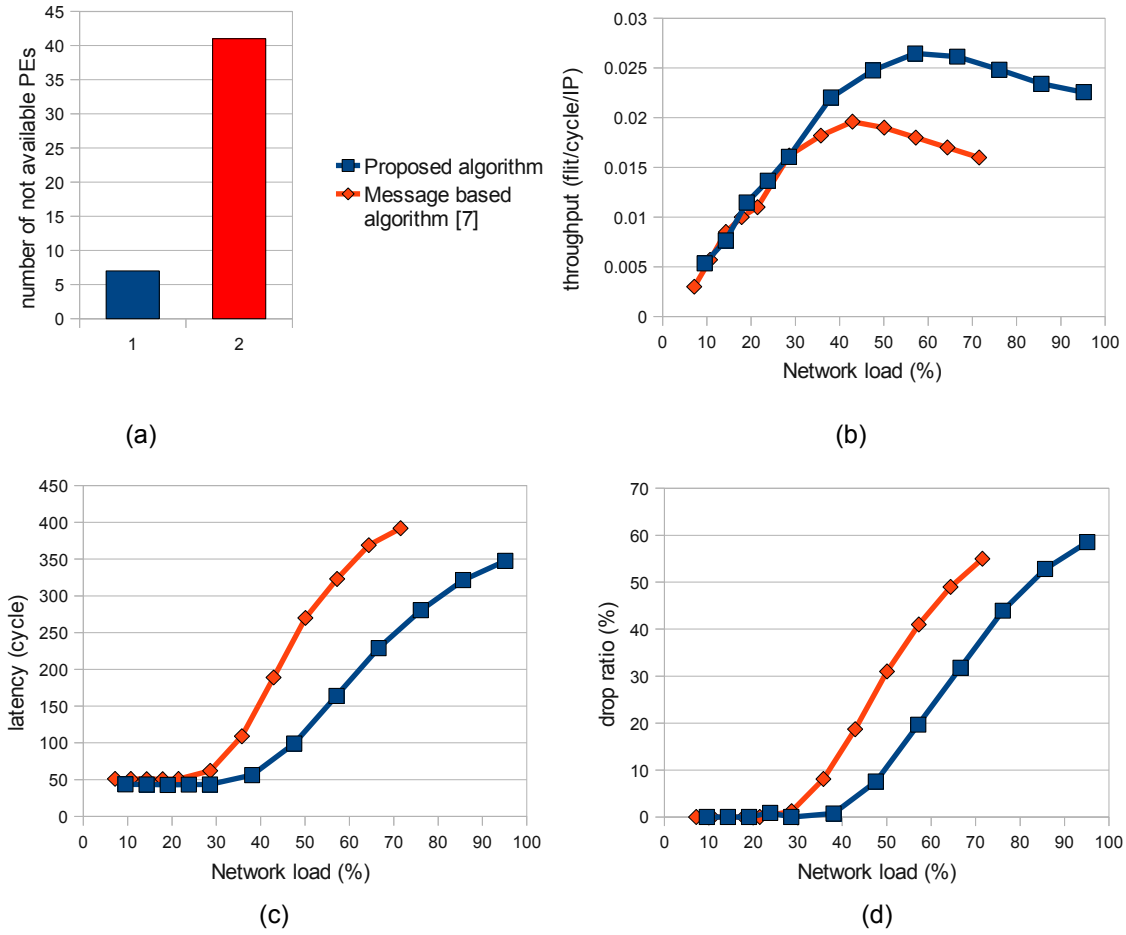

Figure 4.4: Performance comparison of the algorithms under the uniform when the fault counts are 10, and the network size is 12×12 (a) Number of not available processing elements (b) Average throughput (c) Average packet latency (d) Average drop ratio.

Figure 4.4 (a) shows that in reference algorithm forty switches are deactivated, but in proposed algorithm only ten switches are deactivated. Figure 4.4 (b) indicates that, due to the number of the available PE the maximum possible $load_{net}$ improves from 70% to 97%, which results in higher throughput in our algorithm. Figure 4.4 (c) shows that reference algorithm has higher latency. The main reason is that forty switches are deactivated in the reference algorithm, on the other hand only ten switches are deactivated in our algorithm. As a result there are less available routing paths in reference algorithm. As it is shown in figure 4.4 (d), the same as previous examine the higher drop ratio of the reference algorithm is due to less available routing path.

In the next section we compare the algorithms in terms of throughput and latency with a variable number of faults.

Variable Faults

In this experiment the uniform traffic has been applied, and the average latency, throughput and number of deactivated switches are compared in presence of various number of faults in 12×12 NoCs with a fixed $load_{PE} = 10\%$. Due to the low load there is no packet loss and the drop ratio is zero percent.
Figure 4.5 (a) shows that the network with the proposed algorithm is able to deliver more packets and the proposed algorithm has always higher throughput. The result shows that difference between the throughputs has direct relation to the number of the injected faults. As it is shown in figure 4.5 (b) when the number of faults are less than 10, the average latency of the proposed algorithm is less than the reference algorithm due to more possible routing paths. But by increasing the fault density, the result is the opposite, since the number of generated packets in the reference algorithm decreased and the network is less loaded.
Figure 4.5 (c) shows that the number of not available PEs in the proposed algorithm is higher than the reference algorithm especially with high fault densities. This is due to two factors. First, the PEs that are connected to the semi-faulty switch are often still reachable because only a port towards a neighboring switch is affected. Second, because of having smaller faulty region less switches are deactivated in proposed algorithm.

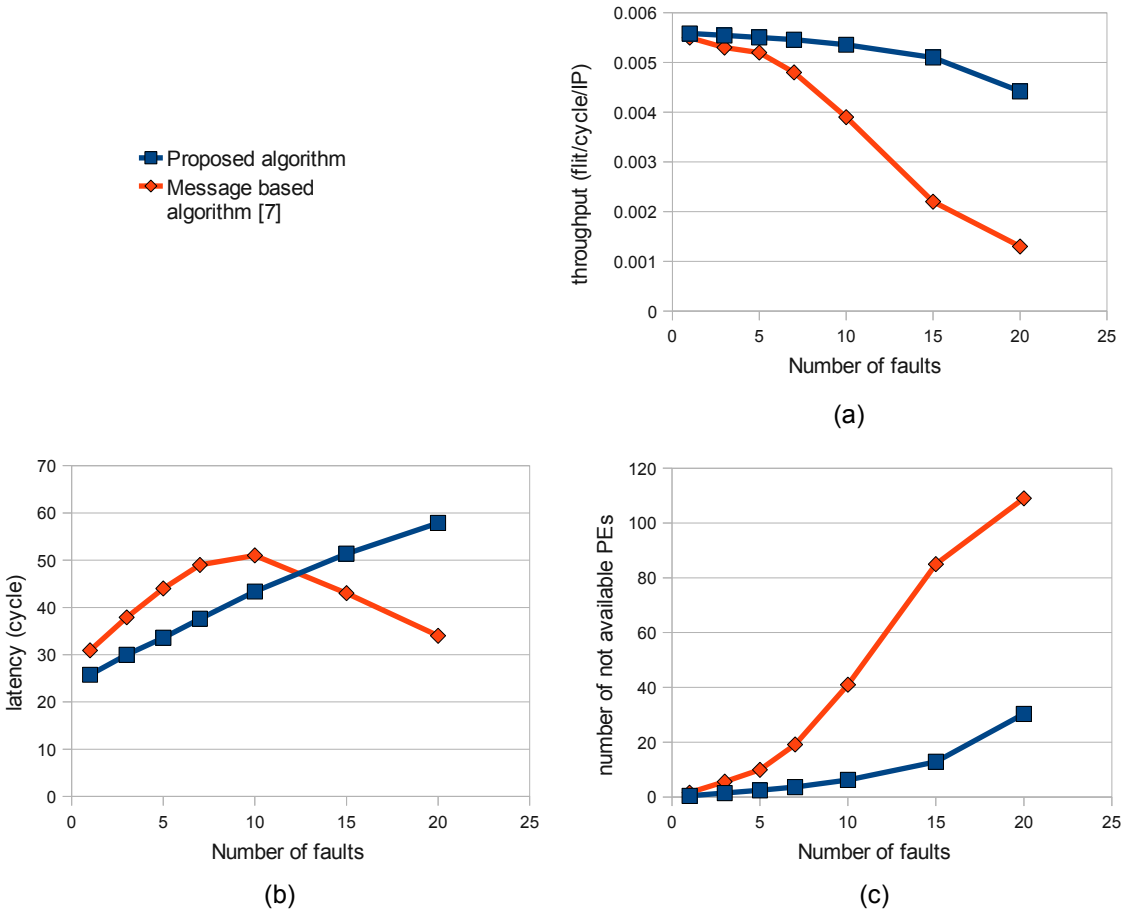

Figure 4.5: Performance comparison of the algorithms under the uniform traffic when the processing element load is 10%, and the network size is 12×12 (a) Average throughput (b) Average packet latency (c) Number of not available processing elements.

Chapter 5

Conclusion

5.1 Summary and Main Contributions

Finding out a routing algorithm which is able to deal with fully and partially defective switches and defective interconnects is the main objective of this thesis. The objective was achieved, and we found out the non-deterministic algorithm that fulfills our requirements.
The first step was reviewing the literature. In this step we found out that there are some algorithms that support the faulty links by taking advantage of table based idea[4], virtual channel [17] and probability based algorithms[18]. The problem of virtual channel and table is the area overhead, and the probability based algorithms are not deterministic. Another common solutions for fault tolerant routing algorithms is region based algorithms, which have been widely proposed since the mid of 90s. The main disadvantages are that a lot of healthy switches have to be deactivated to guarantee deadlock free, and also a link failure is modeled as two faulty switches.
The proposed algorithm is an enhancement of the available region-based approach for NoCs [1]. The novelty of our approach is that link failures are modeled as semi-faulty switches and as a result the faulty region is smaller and less healthy switches are deactivated.
Due to distinguishing faulty-links [11] we proposed a new idea (highway) that let the packets forwarded through the faulty-region which results in less traffic congestion. In chapter 3 by using alignment theorem [6] and channel dependency graph [14] we proved that our algorithm is deadlock-free.
The performance comparison in chapter 4 shows the advantages of the proposed algorithm with state-of-the-art fault tolerant routing algorithms [7]. Since our algorithm has less deactivated switches it has always higher throughput and less latency. And by increasing the number of faults the throughput gain is increased, which is the main achievement.

5.2 Future Work

Overall, this thesis was a success. This dissertation extended the previous fault tolerant routing algorithms and supports partial defective switches. In order to complete the thesis several possible directions for future research are expected.
The majority of future developments are focused on hardware implementation of the proposed algorithm, within this step the area overhead of the proposed algorithm will be measurable. Moreover by implementing on a real chip the flexibility and feasibility of the idea can be determined.
Since the routing algorithm for non-faulty region is the same as XY algorithm, another research direction would be investigating in a congestion aware algorithm for non-faulty region.

Chapter 6

References

[1] Boppana R.V., Chalasani S., "Fault-tolerant wormhole routing algorithms for mesh networks", IEEE *Transactions on Computers*, vol. 44, no. 7, pp. 848-864, 1995.

[2] Zhen Z., Greiner A., Taktak S., "A reconfigurable routing algorithm for a fault-tolerant 2D-mesh network-on-chip", *Design Automation Conference*, pp. 441-446, 2008.

[3] Jovanovic S., Tanougast C., Weber S., Bobda C., "A new deadlock-free fault-tolerant routing algorithm for NoC interconnections", *International Conference on Field Programmable Logic and Applications*, pp. 326-331, 2009.

[4] Fick D., DeOrio A., Chen G., Bertacco V., Sylvester D., Blaauw D., "A highly resilient routing algorithm for fault-tolerant NoCs", *Europe Conference in Design, Automation and Test*, pp. 21-26, 2009.

[5] Jie W., "A fault-tolerant and deadlock-free routing protocol in 2D meshes based on odd-even turn model", IEEE *Transactions on Computers*, vol. 52, no. 9, pp. 1154-1169, 2003.

[6] Chen K., Chiu, G-M, "Fault-tolerant routing algorithm for meshes without using virtual channels", *Journal of Information Science and Engineering*, vol. 14, no. 4, pp. 765-783, 1998.

[7] Holsmark S., Kumar S., "Corrections to Chen and Chiuâs fault tolerant routing algorithm for mesh networks", *Journal of Information Science and Engineering*, vol. 23, pp. 1649-1662, 2007.

[8] Fukushima Y., Fukushi M., Horiguchi S., "Fault-tolerant routing algorithm for network on chip without virtual channels", *Defect and Fault Tolerance in VLSI Systems*, pp. 313-321, 2009.

[9] Holsmark R., Palesi M., Kumar S., "Deadlock free routing algorithms for irregular mesh topology NoC systems with rectangular regions", *Journal of system architecture*, vol. 54, Issue 3-4, pp. 427-440, 2008.

[10] Fukushima Y., Fukushi M., Yairi I. E., Hattori T., "A hardware-oriented fault-tolerant routing algorithm for irregular 2D-mesh network-on-chip without virtual channels", *Defect and Fault Tolerance in VLSI Systems*, pp. 52-59, 2010.

[11] Dalirsani A., Holst S., Elm M., Wunderlich H., "Structural Test for Graceful Degradation of NoC Switches", *European Test Symposium*, pp. 183-188, 2011.

[12] Neeb C., Wehn N., *Issues on Efficient Network-on-chip*, Kaiserslautern Techn. Univ., 2008.

[13] Xu Y., Zhou J., Liu S., "Research and analysis of routing algorithm for NoC", IEEE *International Conference on Computer Research and Development (ICCRD)*, pp. 98-102 , 2011.

[14] Dally W.J, Charles L.S, "Deadlock free message routing in multiprocessor interconnection networks", IEEE *Transactions on Computers*, pp. 547-553 , 1987.

[15] Micheli G. De, Benini L., *Networks on Chips*, Morgan Kaufmann, 2006.

[16] Dally W.J., Towles B., *Principles and practices of interconnection Networks*, Morgan Kaufmann, 2004.

[17] Rantala V., Lehtonen T., Plosila J., "Network on chip routing algorithms", TUCS Technical Report, 2006.

[18] Schonwald T., Zimmermann J., Bringmann O., Rosenstiel W., "Fully adaptive fault-tolerant routing algorithm for network-on-chip architectures", IEEE *Conference on Digital System Design Architectures Methods and Tools*, pp. 527-534, 2007.

[19] Kohler A., Radetzki M., "Fault-tolerant architecture and deflection routing for degradable NoC switches", IEEE *International Symposium on Networks-on-Chip*, pp. 22-31, 2009.

[20] Momeni L., Rezazadeh A., Fathy M., "A low latency routing algorithm for irregular mesh network-on-chip", IEEE *Modeling symposium on computer modelling and simulation*, pp. 328-333, 2010.

[21] Kuo-Shun D., Ching-Tien H., Jyh-Jong T., "Matrix transpose on meshes with wormhole and XY routing", IEEE *Symposium on Parallel and Distributed Processing*, pp. 656-663, 1994.

[22] Mehranzadeh A., Khademzadeh A., Mehran A., "FADyAD- Fault and congestion aware routing algorithm based on DyAD algorithm", *International Symposium on Telecommunications (IST)*, pp. 274-279, 2010.

[23] Ge-Ming C., "The odd-even turn model for adaptive routing", IEEE *Transactions on Parallel and Distributed Systems*, vol. 11, no. 7, pp. 729-738, 2000.

[24] Pasricha S., Yong Z., Connors D., Siegel H., "OE+IOE: A novel turn model based fault tolerant routing scheme for networks-on-chip", *International Conference on Hardware/Software Codesign and System Synthesis*, pp. 85-93, 2010.

[25] Viaud E., Pcheux F., Greiner A., "An efficient TLM modeling and simulation environment based on conservative parallel discrete event principles", IEEE *Design, Automation and Test in Europe, DATE*, pp. 94-99, 2006.

[26] http://noxim.sourceforge.net

[27] Mirza-Aghatabar M., Koohi S., Hessabi S., Pedram M., "An Empirical Investigation of Mesh and Torus NoC Topologies Under Different Routing Algorithms and Traffic Models", *Conference on Digital System Design Architectures Methods and Tools*, pp. 19-26, 2007.

[28] Agarwal A., Iskander C., Shankar R., "Survey of Network on Chip (NoC) Architectures Contribution", *Journal of engineering, computing and architecture*, vol. 3, issue 1, 2009.