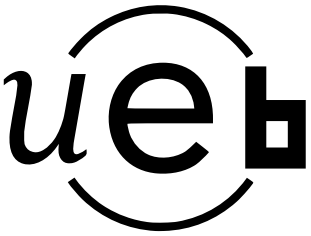
N° d'ordre : 00000 ANNÉE 2013



THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Traitement du signal et télécommunications

École doctorale Matisse

présentée par

Istas PRATOMO

préparée à l'unité de recherche IETR – UMR6164
École National Supérieure des Sciences Appliquées et de Technologie

---

Adaptive NoC for Reconfigurable SoC / NoC Adaptatif pour SoC Reconfigurable

Thèse soutenue à Rennes
le 08 Novembre 2013

devant le jury composé de :

Bertrand GRANADO
Professeur à l'UPMC / *Rapporteur*
Dominique HOUZET
Professeur à Grenoble-INP / *Rapporteur*
Achmad AFFANDI
Professeur à ITS de Surabaya / *Examinateur*
Christophe MOY
Professeur à SUPELEC de Rennes / *Examinateur*
Sébastien PILLEMENT
Professeur à l'Universite de Nantes / *Directeur de thèse*

To my wife, Eka Prasetiya Noviyanti
our daughters, Nadifa Istasabira and Belmine Lumier Pratomo
and my parents, Sri Atmini and Slamet Widodo almarhum

Acknowledge

First and foremost I offer my sincerest gratitude to my supervisor Prof. Sebastien Pillement. This work would not have been possible without his guidance, suggestions, remarks, constructive feedback, comments at various stages and support. He has taught me with patience, both consciously and unconsciously. I appreciate all his contributions of time, ideas and It has been an honor to be his Ph.D. student.

Equally, I wish to thank the head of the Lab Prof. Oliver Sentieys for his kind attitude, generous help and precious suggestions during my PhD thesis. I also want to thank all the members of the jury: Bertrand GRANADO, Dominique HOUZET, Christophe MOY and Achmad AFFANDI who evaluated this work. I wish to thank all the members of the IRISA/CAIRN Lab who make my stay a memorial period. It is a great pride for me completing the PhD thesis in this dynamic Lab.

I am extremely grateful to the Directorate General for Higher Education (DGHE), Ministry of National Education, The Republic of Indonesia for their financial support of this thesis. Finally, I would like to thank my wife, my daughters and my parents for their never ending support and encouragement, that has help me to stay concentrated on my thesis.

Lannion, France
May 2013

Abstract : Chips will be designed with billion of transistors and heterogeneous components integrated to provide full functionality of an current application for embedded system. These applications also require highly parallel and flexible communicating architecture through a regular interconnection network. The emerging solution that can fulfill this requirement is Network-on-Chips (NoCs).

Designing an ideal NoC with high throughput, low latency, minimum using resources, minimum power consumption and small area size are very time consuming. Each application required different levels of QoS such as minimum level throughput, delay and jitter. In this thesis, firstly, we proposed an evaluation of the impact of design parameters on performance of NoC. We evaluate the impact of NoC design parameters on the performances of an adaptive NoCs. The objective is to evaluate how big the impact of upgrading the value on performances. The result shows the accuracy of choosing and adjusting the network parameters can avoid performance degradation. It can be considered as the control mechanism in an adaptive NoC to avoid the degradation of QoS NoC.

The use of deep sub-micron technology in embedded system and its variability process cause Single Event Upsets (SEU) and "aging" the circuit. SEU and aging of circuit is the major problem that cause the failure on transmitting the packet in a NoC. Implementing fault-tolerant routing techniques in NoC switching instead of adding virtual channel is the best solution to avoid the fault in NoC. Communication performance of a NoC is depends heavily on the routing algorithm. Adaptive routing algorithms such as fault-tolerant has been proposed for deadlock avoidance and load balancing.

This thesis proposed a novel adaptive fault-tolerant routing algorithm for 2D mesh called *Gradient* and for 3D mesh called *Diagonal*. Both algorithm considers sequences of alternative paths for packets when the main path fails. The proposed algorithm tolerates faults in worst condition traffic in NoCs. The number of hops, the number of alternative paths, latency and throughput in faulty network are determined and compared with other 2D mesh routing algorithms. Finally, we implemented Gradient routing algorithm into FPGA.

All these work were validated and characterized through simulation and implemented into FPGA. The results provide the comparison performance between proposed method with existing related method using some scenarios.

Keywords : adaptive NoC - impact parameter - fault tolerant routing

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Microprocessor is the most important component in all electronic equipment. It allow to runs specific application software at the user interface level, to manage appropriately internal hardware and software interface based on its functionality. Current applications for embedded systems are modeled as a large number of communicating tasks with different characteristics [39]. Single core processor systems cannot handle anymore the requirement of this highly complex applications and the related real time constrains. Thus, communication architecture is required to support the full functionality of these applications. An embedded system will be designed with billions of transistors and heterogeneous components integrated together in a chip called Multiprocessor System on Chip (MPSoC).

The MPSoC paradigm or the embedded system adopts the architecture of the desktop computers and bus architectures used in PCB by assembling dedicated hardware on a single chip. The current Personal Computer (PC) contains multiple processors, multicore Central Processing Unit (CPU), Digital Signal Processor (DSP), and other application specific-processors (like GPU) to support high computing power, with less power consumption for advanced applications such as multimedia or 3D games. The SoC platform has recently evolved into MPSoC. The interconnection network appeared as the critical bottleneck in MPSoC and the principal important component for high-performance MPSoC. MPSoC has replaced VLSI (very-large-scale integration) or ULSI

(ultra-large-scale integration). MPSoCs are widely used in the equipment systems such as cell phones and portable game devices.

Classically, shared buses and direct point-to-point connections are used for the communications between processing elements on a chip. Both technologies are simple and easy to manage. But, as silicon technology advances further, several problems related to buses and point-to-point links has appeared. With a high numbers of interconnected nodes, bus arbitration can become a bottleneck that can increase the delay. While direct point-to-point, each core needs a lot of pins thus becomes messier in terms of wiring. Buses and point-to-point interconnections are no more possible to support communications between resources on embedded system or System-on-Chip (SoC). To support efficiently these communications it is then required using highly parallel and flexible regular interconnection networks.

Network-on-Chips (NoCs) has been proposed as a solution for communications infrastructure in SoC. NoC architecture provides the communication infrastructure for the resources on a chip. The adaptivity of this communication paradigm provides different Quality of Services (QOS) based on applications need. Moreover, the NoCs can also provide flexible infrastructure to overcome performances degradation due to change in environment or application requirements [36].

A NoC architecture consists of resources or processing elements such as processor or storage units in the network, and switches or router that are connected using channels so that they are able to communicate with each other by sending messages. A generic NoC infrastructure, as presented in Figure 1.1, is the combination of various hardware elements (e.g., processing element, switches, and links) and protocols communication (e.g., routing, switching policies) that determine the communication architecture. The processing elements can be a processor, DSP, Field Programmable Gate Array (FPGA) block or RAM while the switches are in charge to routes and buffers messages between resources.

A physical layout that describes how the switches or routers in NoC are connected to each other is defined as the topology. In SoC, the key requirement for the network

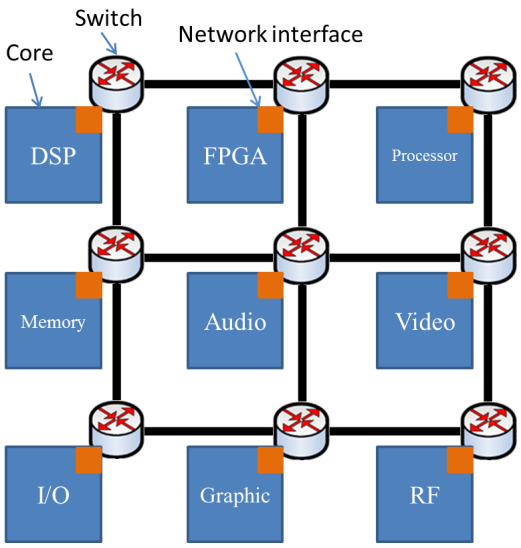


Figure 1.1: General NoC architecture which contains resources (processor, memory, DSP, etc), switch/router and network interface.

topology is scalability with low power consumption [79]. Mesh topology is the most used topology for NoC due to its relative simplicity and its high scalability [13]. In generic 2D mesh NoC topology, several tiles of routers and resources are connected in a grid-like fashion (Figure 1.1). It is known as a regular structure and short inter-switch wires. From this structure, a variety of 3D topologies can be derived. The 3D topology consists of stacking 2D mesh layers connected by I/O or Through-Silicon-Vias (TSVs). It is the solution of the increasing uses of semiconductor and the scalability in embedded system based on the Moore's Law prediction. 3D architecture are designed to avoid suffer from high power dissipation and a large network diameter distance between nodes. It offer better network-on-chips performance compared to the 2D architecture [55] due to the increasing complexity of chips and limited scope of 2D topologies. In this work, we mainly consider the 2D and 3D mesh topology. We designed a fault tolerant routing algorithm for 2D and 3D mesh topologies. Some applications such as video and audio decoding have specific constraints on communication requirements. For these applications, the traffic flow between the pair producer-consumer should need some guarantees on the network performances such as latency or throughput. Latency is the

time needed to send packet from a source to a destination while throughput also known as bit-rate, defines how many bit arrive at destination per second. Traditional packet switching networks which uses packet to communicate with other nodes do not offer guarantees as all packets share the same resources. The quality of service (QoS) refers to a resource reservation mechanism where special packets do not share the resources with other packets. These special packets are called Guaranteed Service (GS) packets while the others are called Best Effort (BE) packets. In worst case scenarios, GS reserve resources to guarantee it services, while BE do not reserve any resources.

Designing a NoC is very time consuming. Different levels of QoS are required to the users based on application constraint. Minimum threshold level of throughput, delay and jitter are needed to fulfill the application requirements. To support a certain QoS for various applications in dynamic condition, an adaptive NoC is mandatory [44]. An adaptive NoC should provide a minimum level of performances to support QoS needed by different application requirements and support flexible communication or dynamic reconfiguration to react and adapt to the changes of working conditions [54]. A designer should consider the QoS requirements of different applications to design an adaptive NoC [27]. A challenge facing designers of SoCs containing NoC is to find NoC instances that balance the cost (e.g. area) and performance (e.g. latency and throughput) [57].

In a NoC based architecture, all nodes are connected and transfers packet through the network via routers. The performances and properties of the adaptive NoCs are dependent on the design of the infrastructure layer, application layer and communication architecture layer as shown on Figure 1.2. In this thesis, routing techniques are designed as a solution towards the dynamic adaptation of NoC. The goal of the routing algorithm is to distribute traffic evenly among the paths supplied by the network topology, so as to avoid hotspots and minimize contention, thus improving network latency and throughput.

An adaptive routing algorithm is proposed for deadlock avoidance and load balancing. The main objective is to reduce the overall latency of communications inside the network. Furthermore they can be used to avoid failures that cause deadlocked packet.

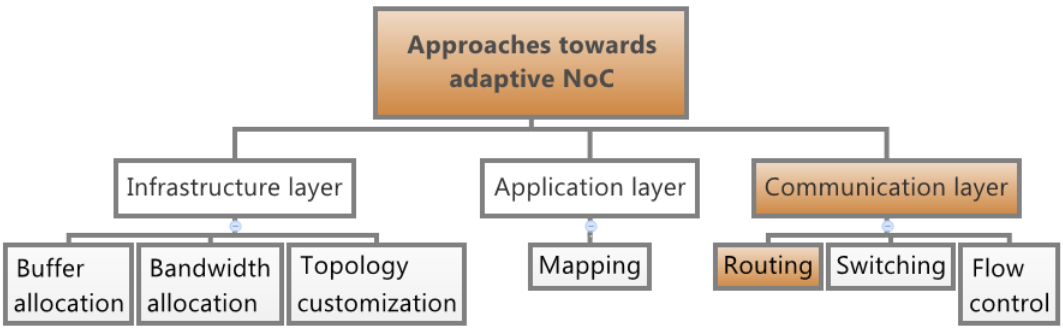


Figure 1.2: Proposed novel routing technique as an approaches towards adaptive NoC

Adaptive fault-tolerant routing [44] is the only solution to reduce the chance of packet entering hot-spots or faulty nodes so that probability of blocking for packets is reduced.

A NoC can be designed in three levels: high level abstraction, medium level and hardware level. A NoC designer can use high-level design to design larger, more complex system and higher performance embedded system using system-level design tools with less effort. The advantages of this method are that it requires less-time to obtain the output performances of the design based on application requirement. Further, NoC designer can also prototype or manufacture their design using high-level synthesis or medium level synthesis. But accuracy of result cannot be guarantee.

Most of existing fault-tolerant routing methods employ virtual channel (VC) in routers to guarantee deadlock-freeness [9]. VCs are known as logical channel which differs from physical wires. It associates multiple queues at each input port in the router so that when a packet is blocked at one port, the other packet can use VC to choose another port. Thus VCs can increase the utilization of physical channel and also the throughput on the network. However, implementation of VCs requires large capacity memory or buffers, which in turn increases area overhead and power consumption, thus making it impractical. Hence, adaptive fault-tolerant routing without VC is desirable for NoCs [74].

A SoC that is composed by several core processors can be built in a single FPGA. FPGA has been chosen by NoC designer to prototypes their design due to the fast and

low cost implementation. In our research, we considered the Xilinx Virtex-5 FPGA to implement our proposed fault-tolerant routing algorithm.

1.1 Problem statement (open challenges)

1.1.1 Impact of design parameters on performances

The introduction of the Quality of Service into a Network-on-Chip requires some kind of end-to-end path reservation in order to guarantee the latency and the throughput of offered packets. The questions addressed for this topic are packet latency, throughput, shared resources and path allocation. The interconnections in NoC have plenty of parameters that affect the performances and the capacity of the interconnection. These parameters can be, for example, the number of data lines, clock frequency, arbitration scheme, the priorities of the blocks, or the maximum time a packet can reserve a shared resource [60]. Almost all NoC parameters such as topology, processing element (PE) number, application type, traffic type, routing algorithm, switching algorithm, packet size, buffer size, flit size and number of virtual channel have influence on throughput, latency and power consumption. For this purpose, a NoC designer should fine several parameters (and their values) impacting the network performances. This way, the adaptation of the network has to fulfill performance requirements.

1.1.2 Fault-tolerant routing algorithm

The use of deep sub-micron technology in an embedded system increase susceptibility to Single Event Upsets (SEU) that can decrease the reliability of NoC. A SEU occurs when a radiation causes a bit-flip in some latches (1 to 0 or vice versa). This undesired modification may cause the dysfunction of the architecture. Another problem that leads to permanent faults in the circuit is the "aging" of circuits. Aging of circuits is caused by physical effect in deep-submicron process and cause permanent errors.

The most important adapted strategies criteria requirement for future circuits is the dependability. The dependability has an impact on computation parts of the embedded

system architecture. If a processor fails, it cannot be used anymore and a different task placement has to be computed. In the same way if the error occurs in the NoC, a re-routing of message in the NoC can be enhanced by adding fault tolerance capabilities, thus they can adapt communication flows to follow fault-free paths.

The important requirements of routing are to avoid deadlock and live-lock. A deadlock occurs when a flit (an elementary portion of a packet) or a packet waits for a resource that will never be released. The routing can create deadlock if bad decisions are taken. The routing-time of packets is one of the key factors critical to the performance of NoCs. Thus, NoC routing schemes should be enhanced by adding fault tolerance capabilities so that they can adapt the dynamically and flexible communication flows to avoid the performance degradation. This leads us to define a communication infrastructure able to handle faults and manage errors in its resources.

In 2D mesh topology, the classical routing algorithms divide the destination node into several zones, and hence route the packets using one routing criterion. While in 3D mesh NoC, classical routing algorithms divide 3D mesh topology into horizontal and vertical layer which some of them reuse existing 2D mesh routing algorithm for horizontal destination. The algorithms are then not able to handle failures in network if a fault occurs in a router or on a link. It also cannot adapt the route to avoid the use of this resource. Other weaknesses are on how it chooses the sequence of alternative routes when the main routing path fails. Inappropriate selection of alternative path may increase the number of hops of packet to reach its destination, thus degrading the performance of NoC.

1.2 Objectives and contributions

1.2.1 Objectives

The main goals of designing a NoC are to get high throughput, low latency, minimum number of resources, minimum power consumption and small area size. In this thesis we first propose an evaluation of the impact of NoC design parameters on its performance.

It shows that the accuracy of choosing and adjusting the network parameters can avoid performance degradation. The results can be considered as a basis for the control mechanism in an adaptive NoC to avoid the degradation of QoS. For these purposes, first we evaluated the impact of NoC design parameters on the performances of an adaptive NoCs. The objective of this step is to evaluate how big is the impact of upgrading a value of a given parameter on performances. In a second phase we evaluated the impact of different type of parameters on the performances. For this purpose, we identified the most important parameters influencing the performances of the network. We then adjust network parameters under different conditions and hence evaluate their impact on the performance variations. One of the challenges of the study lies in the accuracy in choosing and adjusting the NoC design parameters that can upgrade the performances in minimum QoS condition. The results on latency, throughput and reliability were evaluated using the Noxim simulator and show the impact of the parameters on studied performances.

We designed a novel adaptive fault-tolerant routing algorithm for 2D mesh called *Gradient* and for 3D mesh called *Diagonal*. Both algorithms consider sequences of alternative paths for packets when the main routing path fails. The proposed algorithm can avoid more faults and tolerates multiple failures in worst condition traffic in NoCs. It has minimum hops, lower latency and higher throughput in worst network conditions when compared to conventional routing algorithms. To evaluate the performance of these networks, scenarios with various link-faults and node failures schemes were created and simulated using Noxim simulator. Hence the number of hops from source to destination nodes, the number of alternative paths, latency and throughput in faulty network are determined and compared with other adaptive routing algorithms. Further, we implemented *Gradient* into RTL level then simulate the performances using Modelsim and evaluated the hardware cost thanks to Xilinx ISE tool.

1.3 Organization of the manuscript

This thesis is organized in six chapters. Chapter 2 provides an overview of the evolution of the NoC, the NoC architecture, the impact of parameters to performances, fault tolerant routing algorithm and NoC simulator. State of the art on communication such as bus and direct point-to-point are explained in the NoC evolution. Main topologies, switching technique, flow control technique, scheduling technique and routing technique of NoC are also presented. The problem of non-fault tolerant routing algorithm fault-tolerant routing concept and NoC simulators that are used to evaluate NoC design are also introduced.

In chapter 3, the impact of NoC design parameters on the performances of NoC are defined and evaluated. The detail of NoC performances, parameters, and their dependencies are explained. The methodology on how to evaluate the impact of each parameter on each performance is presented. The last section provides the experimental results including evaluation of network saturation condition and the impact of parameters on the NoC performances.

Chapter 4 presents the proposed Gradient and Diagonal fault-tolerant routing algorithms for respectively 2D mesh and 3D mesh topologies. Scenarios with various link-faults and node failures schemes are defined to evaluate the number of hops required to reach a destination nodes and the number of alternative paths. In the presence of faults, the comparison with state of the art algorithm results of minimum hops and alternative path are presented.

Chapter 5 presents the experimental results and implementation of proposed fault-tolerant routing algorithm. The comparison of network performance between proposed algorithms with other routing algorithm is also presented. In the last section, the synthesis result of Gradient implementation on FPGA and the evaluation of frequency on the performance are presented. Finally, the last chapter concludes this work and proposes the perspectives.

Chapter 2

State of the art - Definitions

This chapter gives an overview of adaptive NoC architectures and their evolutions. The comparisons between bus interconnection, general network and NoC interconnection are presented. The architectures of NoC such as resources, interconnection, topology and protocol communication are explained. The 3-D NoC technology is also presented. Then, the characteristics of NoC parameters and the performances of NoC are defined. Finally, an existing related work of adaptive fault and NoC simulators are presented.

2.1 From Bus to NoC

Nowadays, most electronic equipment use system or application that is embedded in a chip. Multi-core in embedded systems has become the basis blocks of computer systems [56]. Direct point-to-point interconnections shown in Figure 2.1 were the first communication infrastructure designed for on-chip systems to communicate between cores. The connection is direct and use dedicated wires without any needs of centralized arbitration. The arbitration decides when incoming data can be served by the router.

In terms of bandwidth availability, latency, and power usage, dedicated point-to-point links are optimal as they are designed especially for this purpose. Also, they are simple to design, verify and easy to model. In small systems of low number of cores, this communication structure is viable. But, as the systems grow and the design

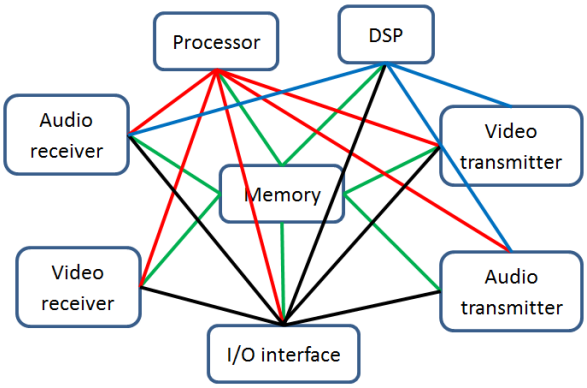


Figure 2.1: Direct point to point connection infrastructure for SoC [7]

cycle time requirements decrease, the number of cores and the number of links will increases exponentially. In large number of cores, it requires a lot of pins for each core, large routing time and area, and becomes very messy in terms of wiring. In terms of performances, the delays and quality of signals become unpredictable, low utilization of routing resources and very low possibility of reuse are experienced in this communication paradigm.

Classically, buses are used for the communication between processing elements on a chip. In a SoC, buses are advantageous because they provide high performance interconnections while they can still be shared by several communication blocks as shown in Figure 2.2. In most SoC applications, a shared bus interconnection is adopted to communicate between each integrated processing unit due to the low-cost and simple control characteristics. But, as the number of units into the system increase, the communication overhead between cores grows and hence quickly become a communication bottleneck. The increasing number of cores in Multi-Processor Systems-On-Chip (MPSoCs) causes the unfeasible intercommunication between cores using single shared bus or a hierarchy of buses. This is because their poor scalability with system size and their shared bandwidth between all attached cores [5].

Data communication in a general networks have replaced buses in small systems: as the PCI-Express, a network-on-a board, replacing the PCI board-level bus [56]. In data communications, networks can be classified into packet switching and circuit switching

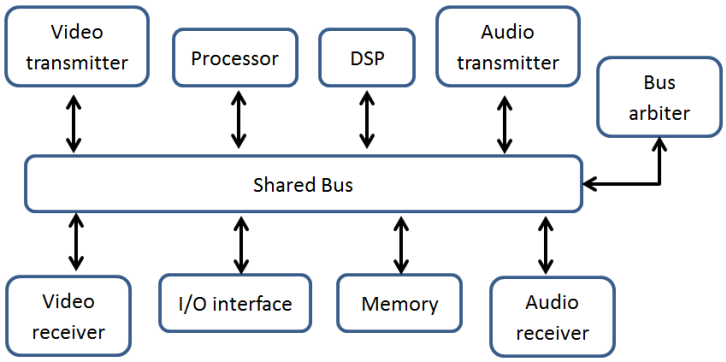


Figure 2.2: Shared bus interconnection infrastructure for SoC

networks. The packet switching uses packets to be transferred with the destination while the circuit switching defines circuits. Circuit switching use dedicated end-to-end circuit to communicate the data.

In the computer network world, different strategy is used. In packet switching network, the packets contain a header packet and payload of packets. Header packet contains routing information needed to route the packet over the network while on the circuit-switching network, an end-to-end circuit (i.e. a physical path) has to be established before any communication can happen. Thus, the routing method is not needed and this is suitable for streaming and guaranteed service traffic. While packet switching send the packets through undedicated path. Thus, it needs routing method to route the packet to destination.

Packet switching is suitable for Best Effort (BE), a traffic on which there is no guarantee on the performances. The disadvantage of using dedicated end-to-end circuit is that it need an allocator to establish the circuit while in undedicated circuit, the resources can be released automatically. In terms of performance, header packet in packet switching is an overhead while circuit switching have guaranteed throughput and predicted latency. The performances of packet switching are depended on the network condition.

Packet-switched on-chip networks (NoC) as shown on Figure 2.3 were replacing buses and crossbars [37] as communication in many-core chips. This architecture is adopting

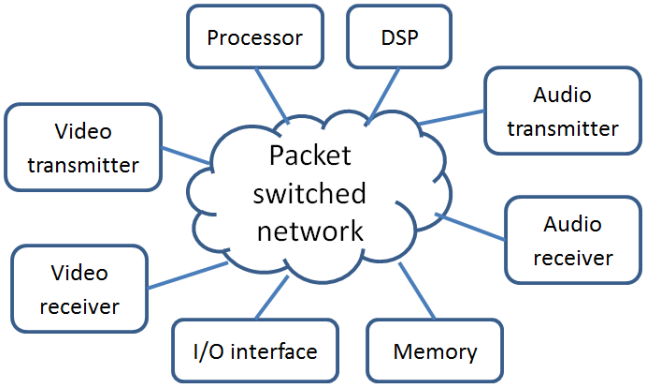


Figure 2.3: Packet switched network communication infrastructure

the computer network architecture for communicating by sending and receiving packets between nodes or processing element. NoC also uses switches and routers to forward packet from sender to destination. The routing decisions are distributed if the network protocol is made non-central and the same router may be re-used for all network size. All network wires can be pipelined that's why the local performance will not degrade when scaling the network size. The similarities and the differences between NoCs and general computer network are shown in Table 2.1.

2.2 Network-on-Chips architecture

The idea of NoC infrastructure is to separate the communication concerns and the application with the physical layout. Thus the architecture can be scalable and configurable as a network. With this communication infrastructure, a hardware resource can be connected to any other resources as an element in the network. General NOC architecture consists of network elements and resources as presented in Figure 2.4. The network elements consist of switches, channels and Resource-Network-Interfaces (RNI) while the resources are processor, core or embedded systems that are integrated into the network. The network elements provide communication services to the whole set of embedded systems. The resources are connected to switch via RNI, while the switches connect to other switches in network using channels. A channel is a two one-directional

Table 2.1: Similarities and differences between NoCs and Computer Networks (CN)

Similarities	differences
Consist of network element (router/switch, link, PE)	NoC designed toward application domain while CN for general purpose
Use packet switching	NoC topology is fixed by design while CN support plug and play router
Flit/packet use header flit that content protocol information such as routing, etc.	Energy is important constraint in NoC thus low power techniques is needed.
Implement communication protocol such as routing, arbitration and flow control.	NoC can't support heavy communication protocol

point-to-point interconnects.

As mentioned, a NoC support hundred or even thousands of resources. The main problem of designing NoC is how to connect them so they can communicate with maximum performances. The physical layout and connections between nodes and channels in the network can be referred as a topology. While communication architecture determine how they communicate each other. The most important element of network is the switch (or router). It routes the data from a sender node to a destination node. A switch contains three major parts: arbitration or scheduling, routing and control flow techniques.

2.2.1 Topologies

The most important parameter of a NoC is the network topology. It gives significant effects on NoC performances due to the fact that it determines the distances between connected node. The two most common topologies for NoCs are mesh and torus as presented in Figure 2.5-a and 2.5-b. These both topologies can be described as k-ary n-cubes, where k is the number of nodes along each dimension while n is the number of dimensions [8]. For 2-D mesh, the value of n is 2 while for 3-D the value of n is 3. For

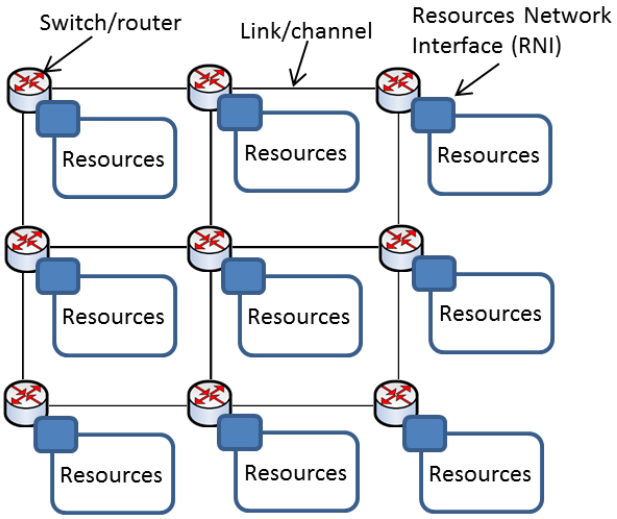


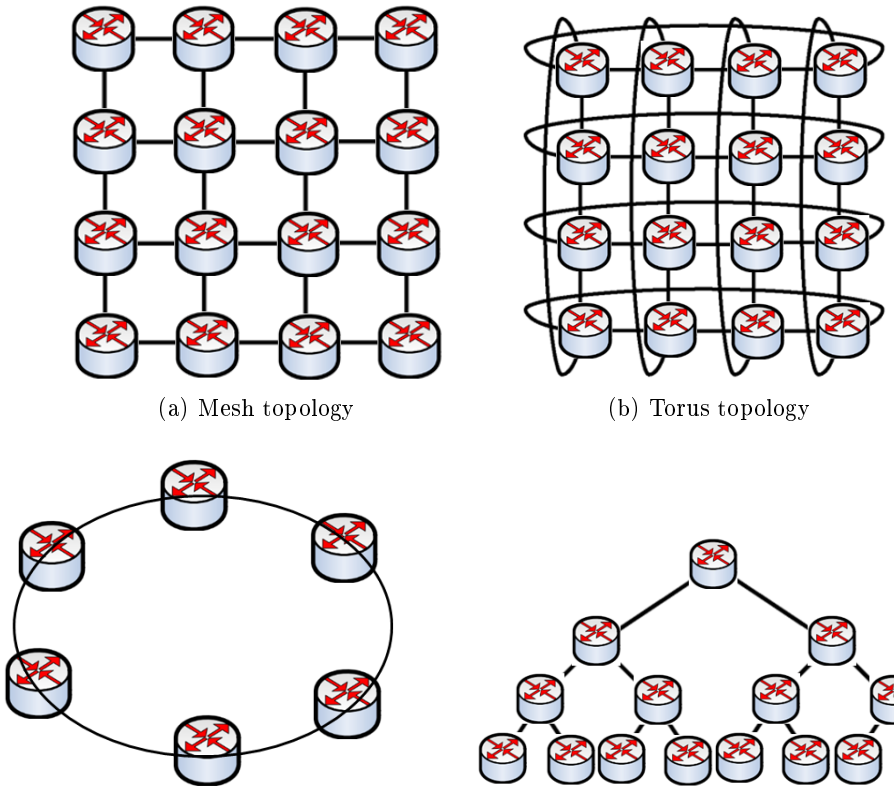
Figure 2.4: General NoC communication architecture

example, a 4-ary 2-cube is equal to a 4×4 mesh or torus with 16 nodes while a 4-ary 3-cubes is equal to 4x4x4 mesh or torus with 64 nodes. In torus topology, nodes along the edge of the network are connected thus these nodes have the same port number than the nodes in the center of the network. A torus is also edge-symmetric [68], this property helps the torus network to balance traffic across channels.

Figure 2.6 present general 2-D meshes router architecture. It has four bidirectional ports (i.e. east, west, north and south) to connect with neighbour routers and a local one to connect to its PE. Each input port has a buffer as temporary storage of data before it is served by the router.

In this thesis we consider mainly 2-D mesh topology due to the wide usage of this topology that has been implemented in the SoC domain. Further, this topology has been used by most researchers to design novel routing algorithm in NoC.

In Ring topology (Figure 2.5-c), all the nodes are connected to each-other in a closed loop. Each node is connected to two other nodes on either side, and can communicate with these two adjacent neighbors. The traffic flows in one direction use a token. This topology does not need central point or server to control the connectivity between nodes. Each node has equal access to resources. The main drawback of this topology is that when one node is faulty, the entire network is affected.



(a) Mesh topology

(b) Torus topology

(c) General ring topology used in NoC

(d) General binary tree topology used in NoC

Figure 2.5: Regular topologies used in NoC: (a) 2-D mesh, (b) torus, (c) ring and (d) binary tree

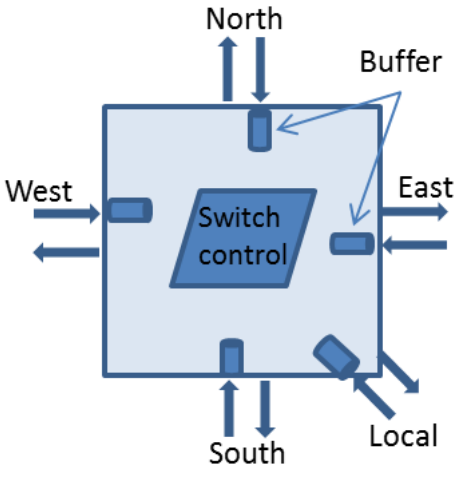
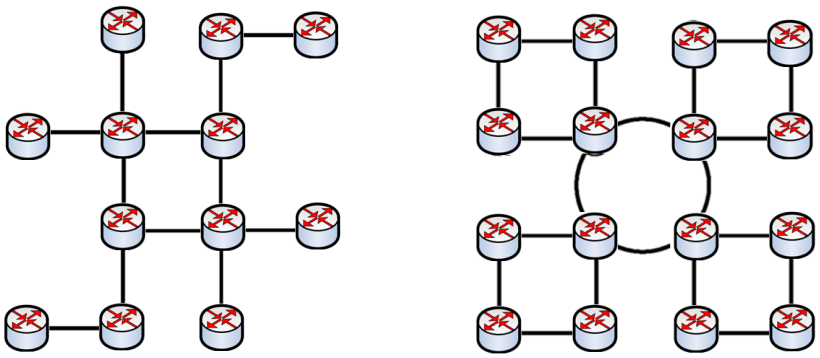


Figure 2.6: General 2-D mesh router architecture



(a) Irregular topology (b) Mixed topology

Figure 2.7: The example of (a) irregular and (b) mixed topology used in NoC

The tree topology shown in Figure 2.5-d are composed of routers (or switches) forming several hierarchical levels. The source PEs is located at the left side of the NoC while destinations (PEs' inputs) are at the right side. Thus, this topology is classified as unidirectional topology.

Irregular topologies presented in Figure 2.7 are usually designed based on clustering techniques. It's derived from altering the connectivity of a regular topology structure such as removing certain links from a mesh or mixing different topologies [8]. In irregular topology, user may specify different router architectures (i.e. number of input and output port, buffer size, routing, etc.) with other router in network. The goals are to reduce the number of ports, switches or channels that can reduce significantly power and area. With irregular topology, some blocks in network can direct be connected without need of switch, or some router need only two input/output ports instead of complete input/output ports.

2.2.2 Switching

The main component of the interconnection architecture in NoC is the switch. Inside a switch, a data is transferred from an input port to any of its output ports. The data may need to be buffered before going out through the output port. The switch control may consist of arbitration, flow control and routing algorithm that govern where and when the data are forwarded. The arbitration decides when incoming data are served by the

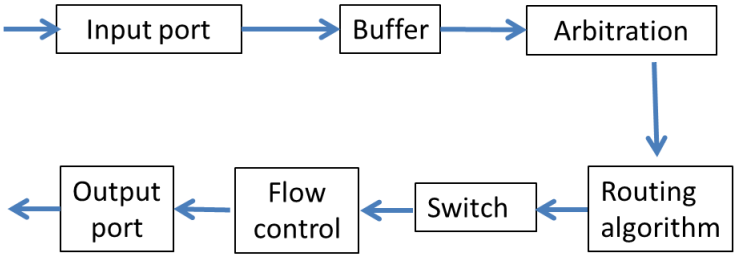


Figure 2.8: General switching process

router, while the flow control decides when the data are sent to the next hop (router). The routing algorithm decides the route on which the data has to be forwarded via a crossbar. The diagram block of general router architecture is presented in Figure 2.8. The time gap between data enters the input port and the time when the data leaves the switch through an output port is called switch-delay [23].

Store-and-Forward (SAF) technique is the first switching technique proposed in NoC. SAF switching techniques convert every packet into flits. The first flit of a packet is the header flit and the last flit is the tail. SAF technique forward the header flits to the next switch if only all body flits of a packet are received. Compared to other switching technique, the SAF switching is not suitable with the requirement of NoCs because it have large latency because all body flits of packet have to be received before forwarding it to the next switch. Moreover, it requires large buffer sizes to accommodate all the flits resulting in a large area needed [23]. The advantage of this technique is no deadlock can occurs.

Wormhole switching technique is similar to SAF switching technique. Both implement packet-based operation with simple control mechanism between routers. In wormhole switching technique, the header flits of packet is forwarded to the next switch before the next flit of packet arrives. Thus, the channel buffer at every router can be as small as a single flit. Moreover, wormhole switching have better latency than SAF because the header flit is processed without waiting the arrival of the next flits or entire packet [23]. The movement of wormhole switching technique looks like a worm. The tail flit follow the same routing path as the header flit. The main drawback of wormhole

switching is the performance degradation due to a chain of packet blocks [23] and risks of deadlocks.

2.2.3 Flow control

Flow control is a technique used in network to control the transmission of data to avoid congestion in a busy network. It is designed to avoid the over usage of the queue that causes congestion and collisions in the network. In NoC, flow control governs the allocation of buffers and links in every router. It determines when buffers and links are assigned to messages, the granularity at which they are allocated, and how these resources are shared among the messages using the network [56]. The implementation of complex flow control protocol in NoC router requires complex wiring that increases router micro-architecture area and power consumption.

The main flow control techniques used are handshaking and credit based. Handshaking control flow is used to avoid the overload of the receiver buffer. It governs the process before the communication between two nodes. The sender start sending the packet only if it receive the acknowledge messages from the destination. The acknowledge messages from destination node can be ready or not-ready to receive the packet.

Credit based communication is the extension of the handshake. It transmits data after receiving return signal from neighbor node that has sufficient free space to store the data. Credit based control how use the concept of buffer management [38]. The flits are transmitted when the buffer space in destination router sufficient. The flits in current input-buffer of router become arbitration for the output-port of neighbour and routers. The current buffers decrement the credit count when flit departs from the current router.

2.2.4 Routing

In term of networks, routing has been classified in several ways: as source routing or distributed routing, and as deterministic, oblivious or adaptive routing. In source rout-

ing, the source node decides the path on which the sending packet will be propagated, while in distributed routing the current node in the networks decides the next hop of the packet. With these schemes, a distributed routing can adapt the route based on network condition while source routing cannot. Routing algorithms are also classified as deterministic, oblivious or adaptive. In a deterministic routing algorithm, all packets take the same path from a source node to the destination [23]. In oblivious routing, the decision of the routing path are fixed or not considering the state of the network condition while in adaptive routing the packets are routed depending on local decisions that considering the state of the network. The main goal of adaptive routing is to avoid congested areas.

Most of routing algorithms in NoC are based on the wormhole switching technique due to its simplicity and its deadlock-free [70]. In this technique, the tail flits follow the same path as header flit. XY routing algorithm [25] is one of the example of a deterministic routing algorithm. It is the most widely used routing strategy for 2-D mesh due to its deterministic, simple, easy and deadlock-free algorithm. In this algorithm a packet first traverses along the X dimension and then along the Y dimension to the destination. XY algorithm is deadlock-free in normal condition but does not support adaptivity. This algorithm is then not able to handle failures in the network. If a fault occurs in a router or on a link, the XY algorithm cannot adapt the route to avoid the use of this resource.

The deterministic routing may cause potential hot-spots if particular router receives more requests than it can serve at a time [56]. Thus it resulting large delay in communication. The solution for this deterministic routing problem is by using Virtual Channel, but this solution comes with a huge hardware overhead.

The main influences factors of routing performances are number of path hop and path distribution. Based on the hops views, routing algorithms can be classified into minimal and non-minimal routing [56]. A minimal routing algorithm use minimal paths pair from source to destination while non-minimal can take both minimal and non-minimal paths. The adaptive routing allows alternative paths between the same pair of

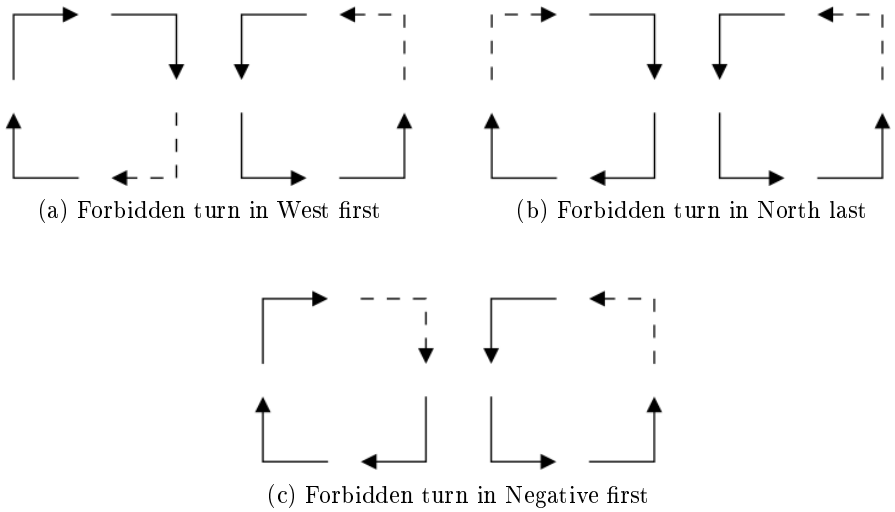


Figure 2.9: The forbidden turns (dashed line) in turn model routing algorithm: (a) West first, (b) North last and (c) Negative first [52]

source and destination nodes. This property provides fault tolerance, because it usually enables the routing algorithm to select a path that avoids faulty network components.

The turn model [25] routing algorithm is an example of adaptive routing. It travel data using a deterministic algorithm, but when the router or the channel fail or is already used for a communication, then the data turn on another direction. Thus, the data reach the destination through an alternative path. The weakness of this algorithm happens when the only path to reach the destination is on the forbidden turn as shown in Figure 2.9. The Odd-Even turn model [12] is designed as a solution of the weakness on previous turn model. This algorithm improves the network performance due to the forbidden turn are more evenly distributed in the network. However, this algorithm doesn't have alternative selection path which is not suitable in faulty network.

2.3 3-D NoC Technologies

Performances, power consumption and the size of chip area are the main consideration in designing a NoC. Existing NoC topologies employ routers with a small number of ports (low-radix) to avoid the increasing power and latency [79]. But, with the need of high NoC performances on current application, it pushes the use of more processing

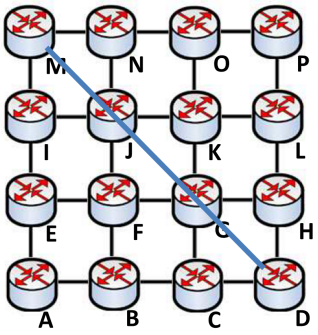
element inside same system that can increase the interconnection requirements between PE. As the network scale, topologies with high-radix (scalable) routers become more feasible than low-radix topologies in terms of both power dissipation and latency caused by large average minimum distance between PE [79].

The conventional 2-D integrated circuit (IC) has limited floor-planning choices, and consequently, it limits the performance enhancements arising out of interconnection architectures [55]. Fat-tree topology has been used to support increasing number of PE and avoid large distances between nodes. But, with a the large number of PE, fat-tree need routers with high number of ports and consequently more wires increasing latency of forwarding packet [49]. A 3-D topology with wafer-to-wafer bonding that consist of multiple stacked connected by Through Silicon Via (TSV) is proposed in [67].

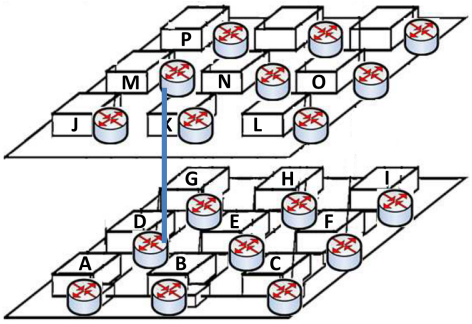
Most of new design topologies interconnection uses vertical integration to face the 3D challenge. This concept is similar with vertical-building like hotel or apartment which is a number of rooms closely connected by using elevator or stairs. This is the idea of 3-D topology to make the distance between nodes closer compared to 2-D topology as illustrated in Figure 2.10. It shows that to the connection between node-D and node-M in 2-D topology is farther (Figure 2.10-a) than 3-D topology (Figure 2.10-b). Using TSV for vertical chip interconnection, leads to shortest distance between two layers, and the bulk capacitance of a wire is also smaller [67].

The main objectives of 3-D NoC technology are scalability and power dissipation. Three-dimensional ICs is capable of achieving better performances, functionalities, and packaging density. The 3-D architecture also offer better interconnection performances compared to the 2-D architecture [55].

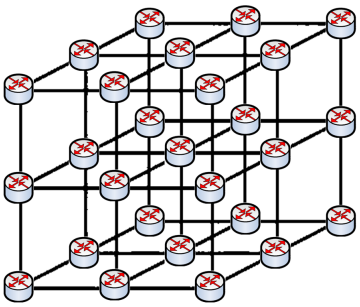
The 3-D mesh NoC can be also fully connected (i.e. all routers have access to the upper and lower layers) as presented in Figure 2.10-c. Similar to 2-D router architecture, 3-D mesh router architecture is presented in Figure 2.11, it employs seven port: six bidirectional port connect to neighbor router (above, below, west, east, north, south) and one port to the local PE. The objective of the design of multilayer or 3-D architecture is to enhance the performance, energy efficiency, and thermal behavior of the



(a) Long connection in 2-D topology



(b) Multi layer topology



(c) 3-D mesh topology

Figure 2.10: The concept of 3-D topology to make the distance between node closer than in 2-D topology

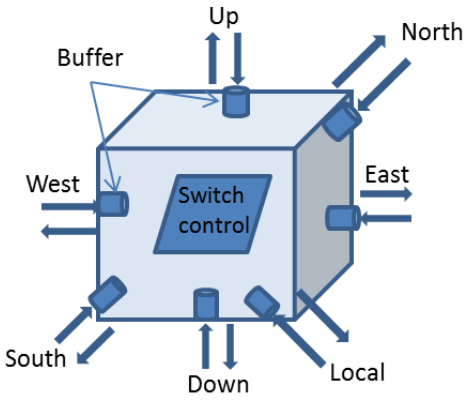


Figure 2.11: 3-D mesh router architecture

interconnection.

The author in [79] proposed architecture of 3-D NOC then compare the power consumption with other topologies such as fat-tree, flattened butterfly, mesh and Clos. Some 3-D architectures such as Stacked mesh, Ciliated 3-D mesh and 3-D BFT which use a bus spanning as vertical connection of 2-D mesh structure has been proposed by Feero and Pande in [20]. Other 3-D architecture which consists of multiple network layers connected via crossbar switches called XNoTs is proposed in [46].

The most widely used static routing algorithm for 3-D mesh is XYZ. It is a deterministic, simple, easy and deadlock-free algorithm. As its counterpart in 2D, packet first traverses along the X dimension, then Y dimension, and finally along the Z dimension. XYZ algorithm is deadlock-free in normal condition but does not provide adaptivity in faults condition thus it cannot handle failures in the network. If a fault occurs in a router or on a link, the XYZ algorithm cannot adapt the route to avoid the use of this resource.

2.4 Performances, parameters and dependencies

The main goals of NoC design are to get high throughput, low latency, minimum resources requirement, minimum power consumption and small area size. But, the most important consideration on designing a SoC is the trade-off between network perfor-

mances, energy consumption, and silicon area requirements after mapping. Generally, the system performance consists of two parts: computation and communication performance. In NoC, the quality can be measured from its energy consumption, area size and network performance.

Several fixed design NoC architectures such as topology, routing and switching schemes has been proposed to get high performances in certain application, however it may reduce performances in other applications. Thus, NoC designer must consider the impact of each parameter on the result performances.

NoC parameters consist of hardware architecture (i.e router micro-architecture, link architecture and topology), and communication architecture (i.e application, security, traffic, transport protocol, packet size, header size, routing, flow control, switching, flit size, and buffer size). While NoC performances can be classified into latency, throughput and reliability.

Latency is defined as the time spent to transfer one packet from a source node to a destination node [31]. In term of NoC, latency is defined as the time elapsed between the moment the PE source sends the first bit of a data and the moment the PE destination receives the last bit of data. Throughput sometimes known as data rate, represents how many bits arrives at destination node per second. In network, it corresponds to the rate at which packets are delivered by the network and presented in percentage from total network capacity.

Reliability is the reliable communication which provide notifications to the sender of the the delivery of transmitted data [31]. In terms of OSI model, reliability depends on the transport protocol layer. Transport protocol ascertains whether the packet arrived at the destination correctly. Complex transport protocol increases the latency of packet. Error control and flow control is a part of control mechanism in transport protocol. Error control and error detection combined with retransmission further can increases the NoC congestion.

In chapter 3, we propose a method to evaluate the impact of design parameters on the performances of NoC. The result of this work will help NoC system designers in esti-

mating the system performance, associated overheads and defining the best parameters to adapt its system condition.

2.5 Fault-tolerant and deadlock-free techniques

An error in the networks can be caused by the faulty in the circuit. There are two type of faulty circuit: permanent faults and temporary fault. Permanent faults can be caused by dielectric breakdown, poor fabrications and irreversible wear-out damage [72] while temporary fault are caused by the operating conditions process such as voltage and temperature fluctuations.

If an error caused by temporary fault, the operation needs to be retried or corrected [63]. But if an error is caused by permanent fault, it needs some form of redundancy in time, space, or information due to retrying an operation will not solve the problem [53]. In that case, sufficient redundancy or spare units are required to continue error-free operation.

The failed element must be remove from the communication system or the faulty chip region must be shuttled down if the permanent fault are caused by poor fabrication yield or lifetime failure [72]. Other solution is by re-routing the packet avoiding faulty area, thus fault-tolerant design is needed. Fault tolerance is the ability of a system to continue operating in the presence of unexpected faults. This property of NoC not only affect fault-tolerance routing strategy, but as simple flow control strategy called Dimensional Bubble Flow Control (DBFC) in [73] has been proposed to avoid fault in network by routing the packet based on its buffer state.

2.5.1 Problems of non-fault tolerant routing

A major problem on oblivious routing typically arises when the network starts to block traffic. The only solution is then to wait for a reduction of the traffic amount and to try again. Deadlock, livelock and starvation are potential problems on both oblivious and adaptive routing.

Deadlock occurs in a network when a group of packets are unable to progress because they are waiting on another one to release resources (buffer, link, etc.). If a sequence of packets forms a cycle in the network [14], then the network is deadlocked. In network communication, a deadlock is dangerous because when few resources are occupied by deadlocked packets, other packets will block on these resources thus completely paralyzing the network. Moreover the network would remain in this state until an external intervention occurs. For example, in minimal adaptive routing algorithm that always routes packets along the shortest path. The algorithm is effective when more than one minimal or as short as possible route between sender and receiver exist. The main drawback is when the minimal path is faulty, the packet is blocked due to in fact that it route only on minimal path (i.e. entering a faulty region).

Closely related network pathology is livelock. In livelocks, packets continue on moving inside the network but without making progress toward their destinations. This becomes a concern for example when packets are allowed to take non minimal paths through the network. Livelock occurs when a packet keeps spinning around its destination without ever reaching it. This problem exists in non-minimal routing algorithms. Livelock should be cut out to guarantee packet's throughput. There are a couple of resorts to avoid the livelock.

Other problems of non-fault tolerant routing are contention and starvation. Contentions are defined as delays imposed to a packet in order to wait for a resource to be available. Contentions are not problematic like deadlocks and livelocks because the network recovers from contention without any external intervention. It recovers as soon as previous communication finishes or as soon as another path is found. Starvation can be avoided using a fair routing algorithm or reserving some bandwidth for low-priority packets.

2.5.2 Fault tolerant and deadlock freeness

Designing a fault-tolerant routing algorithm is mandatory to achieve design of reliable NoCs. Fault-tolerant routing algorithms can use alternative routes when the main

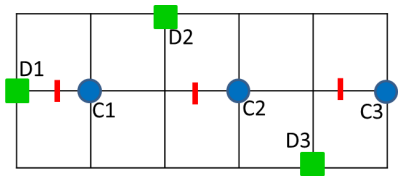
routing path fails. The main difficulty of these methods is on how to choose the sequence of alternative routes since an inappropriate selection may increase the number of hops of packet to reach its destination, thus degrading the performance of NoCs or causing possible deadlocks or live-locks.

Some switching techniques were designed to avoid the deadlock packet in the network. Wormhole and store-and-forward switching technique were proposed for deadlock freeness, however there are no general result that show the adaptivity and deadlock freeness of these techniques.

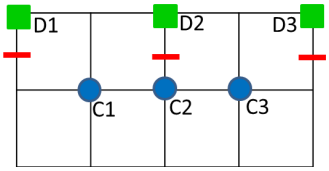
Many fault-tolerant routing algorithms have been designed to avoid the faulty resources in network. The first's proposal was based on the adaptation of turn model such as west-first, north-last, negative-first [25], and odd-even [75]. These techniques were proposed to avoid packet deadlock, lower hardware costs compared to more sophisticated algorithm. But they are not deadlock free routing algorithm and have lows performance due to the fact that they have less adaptivity than fully adaptive routing algorithms in avoiding the fault [69].

Turn model routing algorithms divide 2-D coordinates into four destination zones, based on vertical and horizontal lines. Some of them use only one route and are qualified as deterministic. We found three deadlock in the West-first routing algorithm as presented in Figure 2.12-a. Almost all deadlock in West-first routing algorithm occurs when there is a destination node in westbound part of the current node and there is a fault in the west port of current node. For North-last routing algorithm, the deadlock are presented in Figure 2.12-b. It occurs if there is a fault on the south port of the destination node in the northbound of current node. In Negative-first routing algorithm, the deadlock condition are presented in Figure 2.12-c. The deadlock occurs when the fault in the west port of current node for destination node in northwest and southeast from current node. While in Fully-adaptive routing [51] algorithm, the deadlock will occurs only if there are at least two faults between the current node and the destination node as presented in Figure 2.12-d.

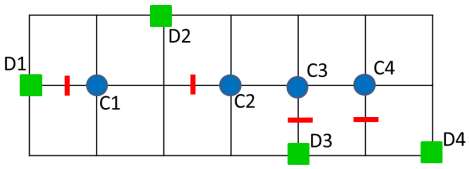
The Dyad routing scheme algorithm [29] combines deterministic and adaptive tech-



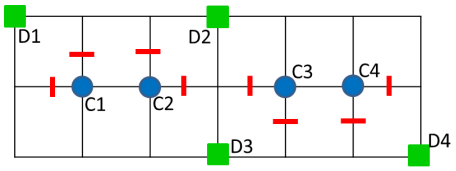
(a) Conditions where deadlocks occurs in West-first routing algorithm



(b) Conditions where deadlocks occurs in North-last routing algorithm



(c) Conditions where deadlocks occurs in Negative-first routing algorithm



(d) Conditions where Deadlocks in Full-adaptive routing algorithm [51]

Figure 2.12: Conditions where deadlocks occur in (a) West-first, (b) North-last, (c) Negative-first and (d) Full-adaptive routing algorithm

niques. Fully adaptive routing algorithm [51] uses always a route which is not congested. The algorithm does not care although the route is not the shortest path between sender and receiver. It may select a non-minimal path allowing uniform distribution of traffic but may result in deadlock. C-routing algorithm in [58] is partially adaptive and prevents live-lock and deadlock without use of virtual channel. It combines XY routing and partially adaptive routing depending on the location of source and destination nodes. All these algorithms were designed for latency optimization and do not take into account faults arising in the network. The RAFT [69] tolerant routing algorithm handles a basic one-faulty-link and can be considered as a fault-tolerant version of DyXY [40]. In [15], the authors proposed a load balancing method to reduce the network congestion using an adaptive scheduler in network interfaces based on the Global Load Balancing (GLB) information metric for arbitration in routers.

A fault-tolerant routing algorithm in [78] propose a method that if there are faulty-router, the algorithm route the packet through a cycle free contour surrounding the faulty router. However this algorithm has more number of hops when it avoids a faulty router to reach the destination. A table-based routing algorithm [21] that can support any NoC topology was also proposed to tolerate faulty links. The main drawback comes

with the update of all routing tables and can stall if the routing table fixes (i.e. the table cannot be updated). Our fault tolerant routing algorithms are proposed to overcome the drawback on minimal adaptive and table-based routing algorithm.

In [10, 11], the authors proposed a fault-tolerant deadlock-free routing algorithm that guarantee message delivery from any source to any destination node, as long as a path exists, for 2-D mesh interconnects of any size. This algorithm will serve as comparison basis for our work.

2.6 NoCs simulator

NoC simulator is a software simulation tool used to simulate the NoC design before its implementation. It used to know the characteristics, the process and the performances results of the NoC design. By the simulation we know which design meets constraint to be implemented. Simulation can give the detail level result that is not experimentally measurable with the current level of technology. Simulation is the cheapest way to design, build, test, redesign, rebuild and retest the design. Despite the advantages of simulation, the fact that a simulation is not real, the results can be far from reality due to the use of models.

Generally, NoC simulator can be classify into high level abstraction and low level environment. High-level simulators work well at the behavioral and architectural levels, but they are useful only in determining the functional correctness of a system. But when the aim is to evaluate the performance or power consumption, low level simulators are needed. OMNet++ [4], NS2 [2] and GpNoCsim [28] can be defined as high-level simulators due to the use of java language. While Noxim [19], Nirgam [3] and Nostrum [42] can be classify in low level environment due to the systemC language usage that can models hardware module. The NoC designer can also directly design a NoC in hardware level or RTL (Register Transfer Level) by writing the logical gates code in VHDL such as ATLAS [1].

In terms of output performance evaluation (i.e. throughput, delay, reliability, power and area), no one of NoC simulator can provide all NoC performance in the same time.

For example Orion [33], is designed only to provide power performance at the micro architectural-level but do not provide latency or throughput. While most high-level simulators such as NS-2 and OmNet++ can only supports latency, throughput, and reliability estimation but not power or area size figures.

Due to the similarities between NoCs and networks, Network Simulator-2 (NS-2) [2] and OmNet++ [4] is very common used tool to simulate and observe the behavior of a NoC at a high level of abstraction. Both provide many varieties of protocols of general computer network such as transport layer, network layer, MAC layer and also physical layer of OSI model. The main drawback is that it is not possible to obtain a structural and physical design view of real NoC such as logic gates canal and RTL logic design. These simulators also cannot model and measure the energy consumption.

There are some open source simulator such as NS-2 or Noxim [19] that enable the users to customize or modify the communication protocol and router architecture. Customizing the communication protocol means that user can change or modify the routing algorithm, flow control, error control, traffic pattern, packet size and topology. While customizing the router architecture enable the modification of the arbitration, routing algorithm, number of port, crossbar, buffer and switch control part of routers. Thus, user can create their router with their own topology or communication architecture. Due to its relative simplicity and its high scalability, majority NoC simulator provide mesh topology in their router library architecture such as Noxim [19], GpNoCsim [28], Nirgam [3], and BookSim [32].

Noxim [19] is the Network-on-Chip Simulator developed using SystemC under GPL license terms. The user can customize the Noxim router architecture such as number of port, buffer size, routing algorithm and selection path strategy. In term of network parameters, Noxim provide some value of packet size and packet injection rate. Noxim also provide some type of traffic time distribution or traffic pattern such as random, transpose1, transpose2, bit reversal, butterfly and shuffle. Random traffic distribution sends the packet to random destination, while transpose1 and transpose2 only send the packet to destination with address on the upper and lower halves of its own address

transposed. Bit reversal traffic distribution only sends the packet to destination address whose is bit reversal of the sender's address.

In evaluation of NoC performances, this simulator delivers the throughput, delay and power consumption in average and per-communication results. For example: the total number of received packets/flits, global average throughput, max/min global delay, total energy consumption and per-communications delay/throughput/energy. We have chosen Noxim simulator to evaluate the experimental results in our work due to its possibility to model a structural and physical design view of real NoC such as logic gates and RTL logic design.

2.7 NoCs prototyping

2.7.1 NoC design process

The processes of NoC design are classified into three level of abstraction: high level design, RTL level design and logic gate level as described in Figure 2.13. The NoC designer can preliminary design by model or specify the function of application requirement and the performance such as throughput or power consumption in high level design. In this level, they can specify the interfaces and behavior using high level tools such as C/C++, UML, Java or Matlab. A logic synthesis tool may convert high level design system model into a detailed behavioral and structural RTL.

A designer can also design his system in middle level design known as behavioral modeling. In this level design, a tool such as systemC provides the libraries or packages to model the behavior of each component or subsystem of the NoC. The result performances in middle level design are closer to the real implementation compared to high level.

The lowest level of designing NoC is in RTL level or in logic gate level. In this level, the representation of hardware circuit and the flow of digital signal between hardware register are described. The most common tools used in this level are Verilog and VHDL. Finally, a place and route tools (i.e. cadance) are needed to synthesis logic gate level

Table 2.2: Resources in Xilinx Virtex-5 (xc5vsx50t-1ff665) FPGA [77]

Resources of Virtex-5 Xilinx FPGA	Total
Number of Slice Registers	32640
Number of Slice LUTs	32640
Number of fully used LUT-FF pairs	11034
Number of bonded IOBs	360
Number of BUFG/BUFGCTRLs	32

design into physical circuit level before the fabrication process as shown in the Figure 2.13.

2.7.2 Prototyping NoC into FPGA

The logic element in modern FPGA device has increased significantly to fulfill the requirement of complex system design. A SoC composed by several core processors can be built in a single FPGA. FPGA has been chosen by NoC designer to prototype their design due to their fast and low cost of implementation. Further, it requires less-time to obtain the output performances of the design based on application requirement. A designer can also uses system-level design tools to design larger, more complex system and higher performance embedded system with less effort. The topology choice for NoC implementation on FPGA is more flexible than on ASIC due to the over-provisioned routing resource on FPGA [41].

A FPGA contains Logic Block (LB), Input/Output (I/O), programmable interconnect and other resources (i.e. memory, multiplexers, global clock buffers and boundary scan logic). Each LB consists of a number of Base Logical Element (BLE) while each BLE contain of Look Up Table (LUT). LUT is the element in FPGA that enables to realize the logic function. The input/outputs (I/Os) connect FPGA with external devices. The logic element resources of Virtex-5 FPGA uses 6-input look-up table (LUT), Flip-flop (FF) and multiplexers to controls the combination of logic input and registered output. Table 2.2 shows the number of logic resources in Virtex-5.

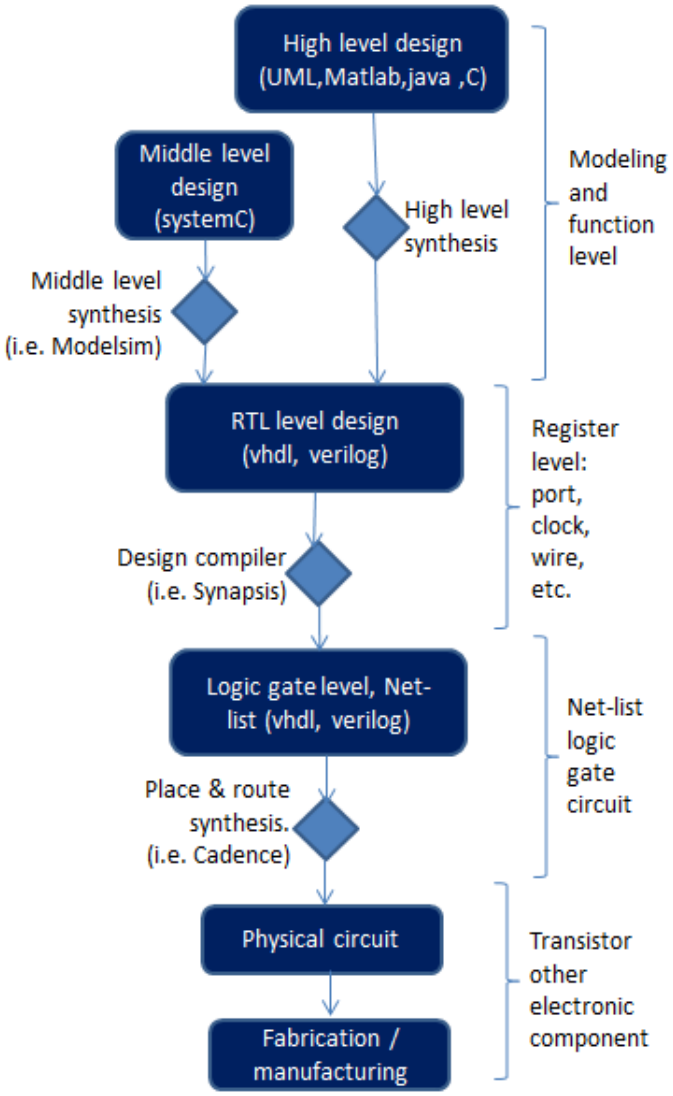


Figure 2.13: Abstraction level of NoC design process [5]

In this Ph.D, we considered the Xilinx Virtex-5 FPGA to implement our proposed fault-tolerant routing algorithm. Compared to previous generation (Virtex-4), this is the first FPGA that was fabricated at 65 nm technology node. Virtex Xilinx FPGA has been used in some similar work to implement NoC design. Moraes et. al. [47] has been successfully prototyped a 2-D mesh NoC router called HERMES onto Virtex-II Xilinx FPGA. The work in [30] has used Virtex-5 Xilinx FPGA to implement A BIST controller for fault detection. Virtex-5 has also used by the author in [35] to implemented a fault-tolerant and congestion-aware adaptive routing algorithm. In term of topology, the work in [18] has also used Virtex-5 FPGA to implement a diagonal mesh topology called FeRoNoC.

2.8 Conclusions

Embedded system in a chip are used as the basis block in most electronic equipment such as mobile phone, digital video camera etc. A single processor in an embedded system cannot handle anymore the requirement of current application on electronic equipment that require parallel and real time constrain. Thus, embedded system is designed with many processors together with other heterogeneous component integrated in a chip called MPSoC.

Point-to-point and shared bus are classically used to communicate between the processing element in MPSoC due to their simplicity, low-cost and high performance interconnections. But, when the number of processing elements increase, point-to-point and shared bus are no more possible to support the communication in MPSoC. NoCs communication architecture has been proposed as a solution to overcome their limitation.

The 3-D technology was proposed to support the need of growing network, communication problem in SoC and as solutions to the limitation performance and floor planning in conventional 2-D architecture. It realizes the stacking of die contouring either processor or memories. A number of 3-D topologies and routing techniques have been presented. 3-D mesh topology is the first most used topology, while XYZ routing algorithm is the simple, easy and deadlock-free routing mechanism.

The main goal of NoC design is to get high performances. But, the most important consideration is the trade-off between network performances and silicon area requirements after mapping. Thus, NoC designers must consider the impact of NoC parameters on the trade-off between network performances and area size. For this reason, in this thesis (chapter 3) we propose an evaluation on the impact of parameters on performances of NoC as consideration for NoC designer to define and adjust the parameters that balance the network performances and area.

A fault in network may be caused by the behavior of network or by permanent fault caused by poor fabrications yield. If it is not anticipated, it may cause error and deadlock packet then decreasing the NoC performance. Fault tolerance routing is one of the solutions to avoid this problem. Thus, in this thesis we proposed a fault tolerant routing algorithm for 2-D mesh named Gradient and for 3-D mesh called Diagonal.

Chapter 3

Impacts of NoC Design Parameters on Transmission Performance

In this chapter we evaluate the impact of NoC parameters setting on performances as a consideration in designing an adaptive NoC. We consider two families of NoC performances that are network performances and implementation costs. Network performances consist of latency, throughput, and reliability, whereas the implementation costs consists of power consumption, area size and efficient use of resources. We used eight combinations of NoC parameter values to define network condition that represents minimum QoS and hence get the resultant performance using systemC based simulator [19]. To see the impact on the performances, we adjust the value and change the type of each parameter. The results indicate the impact of parameters on the NoC performances and the tradeoff involved among them.

3.1 Main NoC Parameters

Communication architecture is used to describe a protocol layer stack. It governs how resources communicate each other and regulates how packets are sent and arrive at destination. Open System Interconnection (OSI) model is successfully employed in general network computer to overcome the problems of buses interconnection. It consists

of the definition of application, transport, network, data-link and physical layers.

Application layer provides user-interface to communicate with other applications running on other nodes through a network. The transport layer is responsible for end-to-end communication and reliable communication. The network layer defines how a packet is transmitted over the network from an arbitrary sender to an arbitrary receiver directed by the destination network address. It implements the routing algorithm of packets from source to destination and decides the form of packets. This layer is also responsible for the segmentation and reassembly of flits, point-to-point routing between switches and contention management.

Flow control is part of the data-link layer. It avoids the queue and collision traffic by controlling the allocation channel and buffer resources in the router. Physical layer is concerned with physical characteristics of the medium used for connecting switches and resources with each other. In the context of SoC, it specifies voltage levels, length and width of the wires, signal timings and the number of wires connecting two units.

3.1.1 NoC router parameters

In a NoC, adding new resources means to add new communication capacity by adding new switches and interconnects. In a bus-based system, adding a new resource has a profound impact on the performance of the rest of the system because the same communication resource is now shared among more resources. This scalability property is a necessary precondition for the arbitrary composability property but it is not sufficient to guarantee it. Further, the communication network must be able to guarantee allocated bandwidth and to enforce a decent behavior of the resources to avoid the monopolization of the entire communication bandwidth by a single resource.

A topology that defines how the resources are connected in a network is the most important choice in designing a NoC. Topology has big effect on the use of routing strategy and the mapping of core to networks nodes. It also impact on the network latency, throughput, area, fault-tolerance and power consumption. While buffer size of routers has influence to latency and throughput. When a packet arrives in a router, it

must go into the buffer of input port. When the buffer capacity of the output queue is exceeded then the last packet will be delayed, increasing the global latency.

Adding virtual channel into a network can be seen as adding input port and the corresponding input buffer to each router. Virtual channel are proposed not only as a solution for deadlock avoidance but also to skirt around blocking in flow control, thus improving throughput. In case of an heavy load on the network, virtual channel port and buffer can handle and store packets from congestion. Increasing virtual channel in the network will increase the throughput and decrease the latency. In case of congestion in a path, the virtual channel can be a solution for alternative path so that packet can be sent without passing through congested path. But increasing virtual channel numbers has a large impact on resources, area of router and power consumption.

In NoC, a packet is converted into flits. The size of the flit will influence the queue in the buffer and the processing time on the router. In heavy networks, when the flit size is bigger than buffer, the flit is queued outside the buffer or transmitted using another channel.

The routing algorithm and switching are used to decide what path a message will take through the network to reach its destination. The goal of the routing algorithm is to distribute traffic evenly among the paths supplied by the network topology, so as to avoid hotspots and minimize contention, thus improving network latency and throughput. While energy overhead of routing circuitry is typically low, the specific route chosen affects hop count directly and thus substantially affects energy consumption and latency.

3.1.2 Network parameters

A packet may have different size, header and type. The header contains routing information. The different packet format affects the performance of network especially throughput and latency. Packet size is the total size of packet load and header packet.

Table 3.1: Impact of NoC parameters to performances

Layer	NoC Parameter	Network performance			Hardware	
		Through put	Latency	Reliability	Power	Area Size
Application	traffic type					
	traffic rate					
	security method					
Transport	Transport protocol					
	Error control type					
	Flow control type					
Network	Packet format					
	Topology					
	Virtual channel					
	Packet/header size					
	Routing algorithm					
	Switching method					
Datalink	Flits size					
	Buffer size					
Physical	Channel material					
Router Micro architecture	Arbitration					
	Allocators					
	Crossbar					

Color green = give impact, color red = not give impact

Increasing injection traffic rate in the network cause heavy traffic. Network be-

comes more congested. In other hand, if the channel capacity is bigger than the network load, increasing injection rate will increase the throughput. On contrary if the channel capacity is lower than the network load, increasing injection rate will degrade the throughput. Table 3.1 shows several parameters of NoC that have influence to the performance of NoC. Almost all parameters have influence to throughput and latency. The more resources are used the more power consumption increase. The hardware of channel between resources have influenced to area size of chip.

3.2 Existing work on evaluate of the impact of parameters on performances

There are a number of methods that can be used to evaluate the impact of design parameters on the performance of NoC for providing QoS. A typical way to approach this task is by manual method, which often implies trial-and-error design phases [60]. Obviously this method tends to be cumbersome, error-prone and even tedious. However, there are analytical methods but often these mathematical methods use far too abstract models and therefore the achieved results differ from real implementations [22, 50]. These methods also make too many assumptions about the network and traffic to get accurate values for a real system.

Various techniques based on simulation are already used to identify the impact of NoC parameters on performance. Narasimhan et. al. [48] investigates the impact of different NoC topologies and traffic types on different applications using OPNET simulator. Gehlot et. al. [24] investigates the performance of NoC topologies such as CLICHE, Folded Torus, BFT, SPIN and octagon using NS-2 simulator. An evaluation in terms of cost and performance by sweeping over different parameters (e.g. network topology, network interface queue depth) using a XML is proposed in [57]. A simulation framework for CMP system based on virtutech Simics [43] and GEMS [45] tool-set is used to evaluate the choice of key network parameters (topology, flit size) on the behavior and performance of applications running on top of different network configurations

presented in [71]. Bagherzadeh et. al [6] evaluates the performance impact of the communication protocol depending on the task structure. Some works made evaluation in terms of cost and performance by sweeping over different parameters (e.g. network topology, network interface queue depth) using XML, for example in [57] and [6]. The main drawbacks of these previous works rely to the level of abstraction which leads to inaccurate evaluation, or cannot provide information on hardware implementation costs.

In the proposed work we evaluate the impact of NoC design parameters on performance using systemC based simulator. We used a systemC based simulator because designing hardware for each NoC in the design space is unrealistic, and systemC utilize low-level hardware modeling combined with C++ for implementing the network characteristic and evaluating the network performances. The purpose is to obtain sufficient accuracy in choosing and adjusting the design parameters that can upgrade the performances in minimum QoS condition. For this purpose, we define scenarios that represent network condition with minimum QoS and hence we adjust the value of each network parameter. The resulting performances such as latency, throughput, reliability and power consumption are evaluated using Noxim simulator.

In Noxim, power is estimated by total power consumption of router and each incoming and forwarding packet in router during simulation time. Noxim has defined the power value of each NoC parameter such as buffer, packet size, routing type and selection path strategy. Thus, the power consumption in simulation is dependent on the number of transferred packet, parameters used and simulation time.

3.3 Impacts of Design Parameters on Performance

To evaluate the impact of NoC parameters to its performances, first we classified the NoC performances in two part: network performances and implementation costs as shown in Figure 3.1. The network performance consist of latency, throughput and reliability while implementation cost consist of power, area size and resource use.

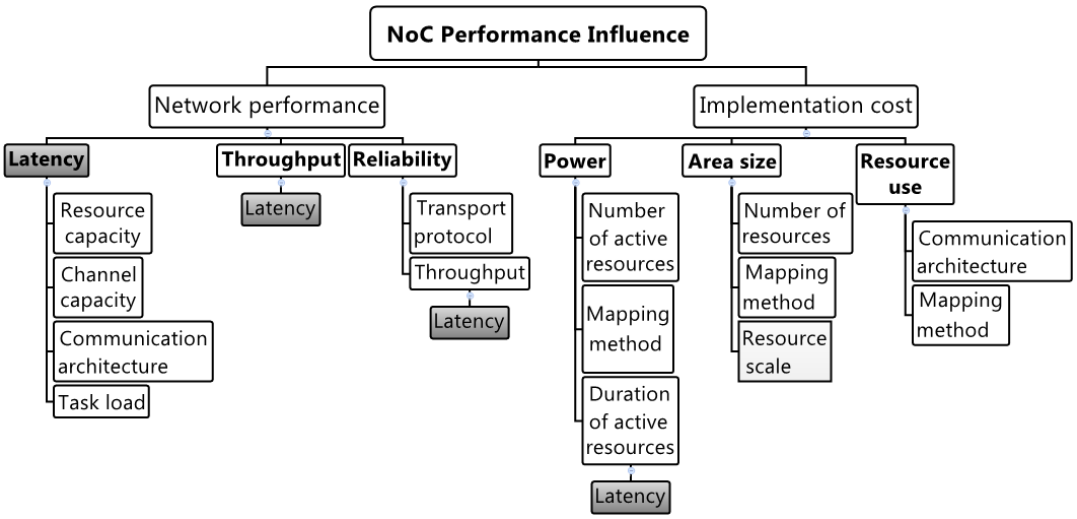


Figure 3.1: NoC performances classification, parameter impact and the major influence of latency to other NoC performances

We found that latency, throughput, reliability and power consumption are mainly influenced by resources capacity, channel capacity, communication architecture and task load as shown on Figure 3.1. Resources capacity indicates the maximum data information that can be processed in Processing Element (PE). It represents the performance of the router or processor. Channel capacity indicates the maximum amount of data information that can be reliably transmitted over a communication channel. The higher capacity of resources and number of channels can decrease latency, increase throughput and increase reliability.

Communication architecture in Figure 3.1 governs how resources communicate with each other. The choice of topology and communication protocol used in NoC is an example of communication architecture. Different implementation of NoC topology and communication protocol has differences impact on latency, throughput, reliability, power and resource usage.

Transport protocol implements the control strategy such as error control, error detection and error correction to provide Quality of Service (QoS) on the network. QoS guarantee reliable communication packet such as required bit rate, delay, jitter and bit error rate. A good transport protocol increases the reliability but is more complex and

then increases the delay and the required area. Transport protocol defines the size of packet as a solution of congestion in network. In large networks, small packet size can increase the performance because small packet size is transmitted faster than big packet sized one.

Mapping method has influences on area-size, power consumption and resource-use. The efficiency of using resources is influenced by design of communication architecture. Table 3.2 presents the main parameters that have an impact on the performances of NoC. We can note that NoC hardware micro-architecture have an impact on area and power because each resource needs power and space. Thus, increasing the resources scale or hardware micro-architecture in Figure 3.1 give impact on increasing the area size and power consumption.

Adding resources based on task placement in NoC system also cause the path of packet farther to reach destination, hence increasing latency performance. Total energy consumption is a result of multiplying the power with duration time. The increasing of latency causes the increasing in resources activity time thus increasing energy consumption. The mapping method in Figure 3.1 means topology mapping and task mapping. Topology mapping is assigning each PE resource inside a network and task mapping is assigning a task to a PE. A bad mapping method uses more resources, need more area-size and increase power consumption.

Simple communication architecture uses only few numbers of resources and area thus power, area and use of resources are minimized. Conversely, complex communication architecture uses more resources thus causing an increase on power consumption and area size. Increasing injection rate in the network cause heavy traffic. Network becomes more congested thus increasing latency. In other hands, if the channel capacity is bigger than the network load, increasing injection rate will increase the throughput. Conversely, if the channel capacity is lower than the network load, increasing injection rate will degrade the throughput.

Latency (L) is one of the performances required to provide QoS [64]. We have evaluated that latency is the most important performance of network. On limited perfor-

Table 3.2: Impact of design parameters on performance of NoC

NoC parameters	Impact on network performances			implementation cost		
	Latency	Throughput	Reliability	Power	A.size	R.use
Hardware micro						
Mapping method						
Topology						
N.resources						
Application type						
Task load						
Injection rate						
Transport protocol						
Packet size						
Routing						
Header size						
Flit size						
Buffer size						
Flow control						

A.size = Area Size, R.use = Resource use
Color green = give impact, color red = not give impact

mances of network, latency may represent other NoC performances such as throughput, reliability and power. Several NoC parameters which have influence on the latency will automatically have influence on throughput, reliability and power. Throughput depends on time because it is calculated by the number of arriving bit per time [64]. On overloaded network when latency increase, the packets are queued and increase the arriving time of the packet, thus decreasing throughput. When latency increase, the network cannot guarantee that a packet will arrive at destination in a reliable time thus impacting reliability. The increasing of energy consumption is linear to duration time of active resources. When latency increases, it will increase the duration time of active

resource thus increasing power consumption. That is why latency is the most important NoC performance indicator because it influences three NoC performances.

3.4 Methodology and Experimental Result

This section presents the methodology and the experimental results of our work. We define three scenarios in order to evaluate the impact of each parameter on the NoC performances. Firstly we evaluated the impact of parameters that causes network saturation. In this scenario we want to find the combination of parameter that cause performance saturation and also evaluate the saturation point. By knowing these information, we then avoid the use of these combinations and the value after saturation in next scenario which evaluate the impact of parameter on performance.

We implemented the scenarios into Noxim simulator then evaluate the result of each scenario. Noxim provides several type of network parameters (i.e. routing algorithm, traffic distribution, selection path strategy and arbitration) and network performance (i.e. average delay, throughput and power) needed in simulating the NoC design. Further, Noxim is customizable and modifiable due to its open source code. User can customize the router architecture such as number of port, buffer size or adding control mechanism in the router.

In terms of output result, Noxim simulator provides output performances such as latency, throughput, reliability and power. Latency is defined as the time spent to transfer one packet from a source node to a destination node while throughput is the network connection rate or channel capacity and is evaluated as a number of flit per cycles. Reliability guarantees the delivery of packet arrived at destination in a tolerable time. In the simulation, we measured the reliability by dividing the number of received packets with the number of sent packets during simulation time. To get the percentage of the reliability, we multiply the results by 100. Power dissipation denotes the energy per time required to operate an embedded system. Total power dissipation is calculated as the sum of energy for switch, buffer, wires and link. In the simulation, power calculated by the total of incoming flits, outgoing flits, active router and standby router. Each

router has different power consumption depending on the use of routing algorithms and selection path type.

3.4.1 Worst condition scenarios

In real-life application, worst condition happens when the performances degrade over the threshold limit of QoS. Thus the system must adapt this condition by adjusting the parameters that have significant impact on increasing performance. Worst condition is chosen because in normal condition the performances of network are maximum, thus the upgrading of parameters cannot give significant impact on the upgrading performance or saturated network. In normal condition, we cannot define which parameter can give more impact on performance.

We define the worst condition by combining eight parameters: number of hotspot node, buffer size, packet size, packet injection rate, routing type, selection path strategy and a number of resources. The challenges on determining worst network condition where the performance start to increase is to define the parameter values. We have evaluated some combination of parameters hence defined a minimum or maximum value of parameter that causing worst condition as presented in Table 3.3.

Table 3.3: Parameter values for worst network condition for different size of network

Parameter	3x3 mesh	4x4 mesh	5x5 mesh	6x6 mesh
Number of hotspot node	min. 1 node	min. 2 nodes	min. 3 nodes	min. 4 nodes
Failure node percentage	min. 0.5%	min. 0.5%	min. 0.5%	min. 0.5%
Buffer size	max. 1 flit	max. 1 flit	max. 1 flit	max. 1 flit
Packet size	min. 6 flits	min. 6 flits	min. 6 flits	min. 6 flits

In the scenario to evaluate the impact of parameter on performance (chapter 3.4.3), we used 2D mesh dimension from 3x3 to 6x6. Thus in this scenario we have evaluated

3x3 to 6x6 2D mesh dimension by increasing some parameters value. We found that, to degrade the performance of 3x3 mesh dimension, it must be at minimum one faulty node with a failure percentage minimum 0.5%, buffer size of maximum one flit and packet size composed by a minimum of 6 flits as presented in Table 3.3.

Logically, larger packet size can decrease the overhead of data which then increase throughput. But in the network, the limitation of buffer, retransmission and transmission delay affect degradation of throughput. Thus, to set worst condition, we degrade the performance by using big packet size. In this scenarios we used a packet size of minimum 6 flits.

Increasing packet injection rate will increase throughput but contrarily will degrade the latency when the network is overload. Thus, there is no minimum or maximal value for packet injection rate in determining worst condition.

3.4.2 Network saturation

Each designed network communication has a limit of connection performances known as bandwidth limit. In this sub-section, some scenarios are defined to evaluate the network saturation condition where throughput is maximum. We have combined some NoC parameters and adjusted the value of each parameter to upgrade the throughput until saturation.

Firstly, we define the range value for each parameter as presented in Table 3.4. Secondly, we define four scenarios (Table 3.5) to evaluate the maximum throughput when the system adapt the degradation of performance. The objectives of these scenarios are to find what combination of parameters causes performance saturation and then define the starting point of the saturation. We will then avoid the use of these parameters value in evaluating the impact of parameters on performances (chapter 3.4.3).

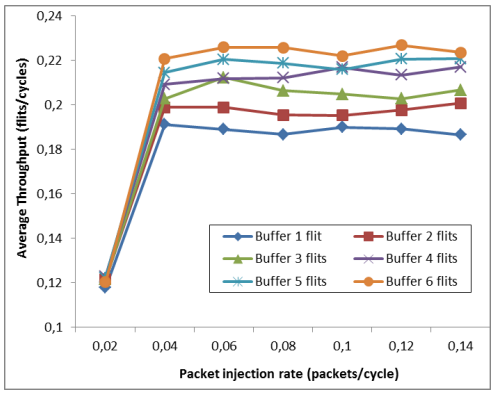
Table 3.4: Parameters range value used in scenarios to evaluate the saturation performance

Parameter	Value
Simulation time	10k - 100k cycles
Faulty percentage of node	0% - 100%
Packet injection rate (Packets/cycle) scale	0,02 - 1
Packet size	2 - 14 flits
Buffer size	1 - 10 flits

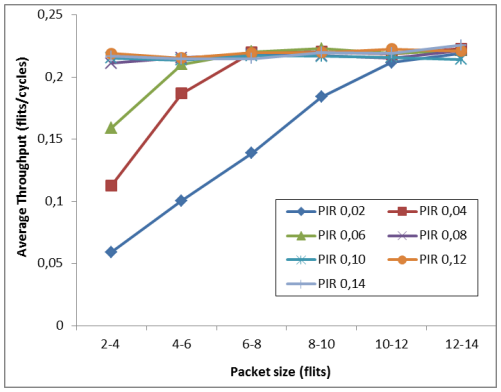
Table 3.5: Parameters range value used in scenarios to evaluate the saturation performance

No	Scenarios
1	Evaluate the network saturation on combination buffer size with packet injection rate
2	Evaluate the network saturation on combination packet injection rate with packet size
3	Evaluate the network saturation on combination buffer size with packet size
4	Evaluate the network saturation on combination faulty percentage with packet size

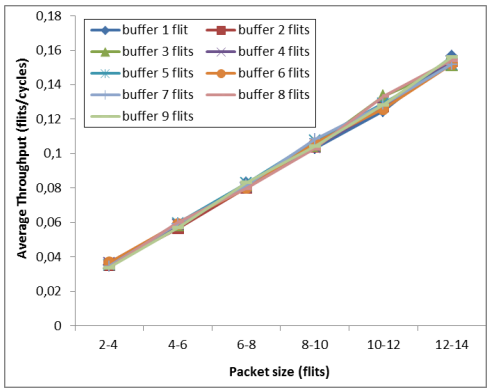
The results of the experiments are shown in Figure 3.2 while the summaries are presented in Table 3.6. The result of the first scenario (Figure 3.2a) shows that however we upgrade the parameters value, the throughput is saturated or reach maximum value (0.22 flits/cycle). The saturation point starts after packet injection rate bigger than 0.04 packets/cycle.



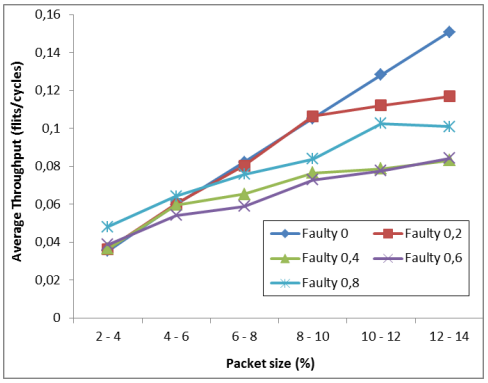
(a) Evaluation of throughput saturation using different buffer size on increasing packet injection rate



(b) Evaluation of throughput saturation using different packet injection rate on increasing packet size



(c) Evaluation of throughput saturation using different buffer size on increasing packet size



(d) Evaluation of throughput saturation using different faulty percentage on increasing packet size

Figure 3.2: Evaluation of throughput saturation

For scenario 2 (Figure 3.2b), the result shows that the maximum throughput is 0.22 flits/cycle. In this result, each packet injection rate has different saturation point. For example, the scenario with packet injection rate from 0.08 to 0.12 packets/cycle start saturated for packet size of 4 flits, while packet injection rate 0.02 packet/cycle start saturated for packet size of 12 flits.

The results of scenario 3 (Figure 3.2c) shows the throughput is not saturated, thus

the parameter value used in this scenario can be upgraded until reaching maximum performances. The result of scenario 4 (Figure 3.2d) shows that each faulty percentage have different maximum throughput. For examples, for faulty percentage 0%, the throughput is not saturated while for faulty percentage 0.2, the throughput reach maximum value on 0.12 flits/cycle. In this condition, the upgrading parameter value to adapt the degradation of the performance can upgrade the throughput maximum 0.12 flits/cycle. For faulty percentage 0.6 and 0.8, the maximum throughput are 0.08 flits/cycle while for faulty percentage 0.6 the maximum throughput 0.1 0.08 flits/cycle.

Table 3.6: Simulation result of latency and starting saturation point

No	Evaluated parameters scenario	Network saturation	
		Maximum throughput	Starting point
1	Different buffer size on increasing packet injection rate (Figure 3.2a)	0.22 flits/cycle	0.04 packets/cycle
2	Different packet injection rate on packet size (Figure 3.2b)	0.22 flits/cycle	14 flits
3	Different buffer size on increasing packet size (Figure 3.2c)	Not Saturated	-
4	Different faulty percentage on increasing packet size (Figure 3.2d)	0.15 flits/cycle	8 flits

3.4.3 Impact of adjusting parameters value on performances

In previous evaluation, we have evaluated the combination of parameters that may give impact on network saturation. The result shows that all combination of evaluated

parameters are not causing network saturation except for buffer size. We avoid the use of this combination of parameters, especially the use of parameter values after the saturation point.

Table 3.7: The defined and adjusted parameters used in the scenario

No	Parameter	Value/type
1	Routing type	XY
2	Traffic distribution type	Random
3	Selection path strategy type	Random
4	Failure node percentage	0.5% per node
5	Injection rate	Adjust based on scenario
6	Packet size	Adjust based on scenario
7	Buffer size	Adjust based on scenario
8	Number of resources	Adjust based on scenario

In these experiments we evaluated the impact of adjusting value of parameters on the performances. The objective is to find which parameter has the highest impact on performance compared to other parameters. The results of this evaluation can be used for decision taking to adapt worst network condition in designing an adaptive NoC. For this purpose, first we define the worst condition that represents QoS degradation as presented in section 3.4.1. Then, we adjust the value of evaluated parameters (No. 5, 6, 7, 8) in Table 3.7.

Adjusting the value of evaluated parameters (i.e. injection rate, packet size, buffer size and number of resources) can be decrease or increase the value. To adapt the degradation of the performance of NoC, the value of parameters must be adjusted so that

Table 3.8: The adjusting parameters to upgrade the performance

Evaluated performance	Adjusting parameter value			
	Injection rate	Packet size	Buffer size	Number of resources
Latency	Decrease	Decrease	Increase	Decrease
Throughput	Increase	Increase	Increase	Increase
Reliability	Increase	Decrease	Increase	-
Power consumption	Decrease	Decrease	Decrease	Decrease

the performance upgrade. We found that, to upgrade the performances of latency, the value of injection rate, packet size and number of resources must be decrease while the value of buffer size must be increase as stated in Table 3.8. To upgrade the throughput performances, all evaluated parameters (i.e. injection rate, packet size, buffer size and number of resources) must be increase. In vice versa, to decrease the power consumption, all evaluated parameters must be decrease (Table 3.8). For reliability performances, the injection rate and buffer size value must be increase while packet size and buffer size must be decrease.

In worst condition, the decreasing packet size can increase the performances which are decrease the latency then degrade the throughput. While increasing packet injection rate in overload network condition will degrade the latency hence degrade the performance.

The result in Figure 3.3 shows that decreasing packet injection rate and decreasing packet size give more impact on reducing the latency compared with increasing buffer size and decreasing number of resources. Thus, adjusting the packet injection rate is considered as the best decision to adapt the worst condition of network caused by latency degradation.

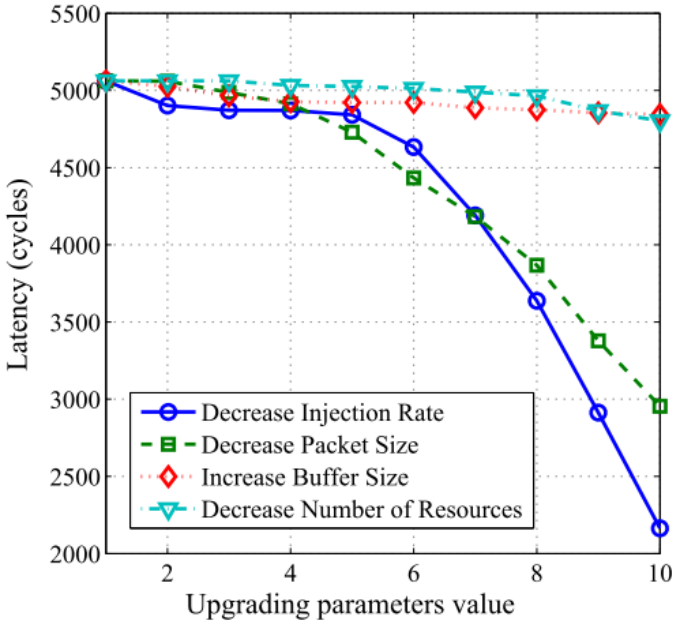


Figure 3.3: The impact of upgrading value of injection rate, packet size, buffer size and number of resources on upgrading the latency

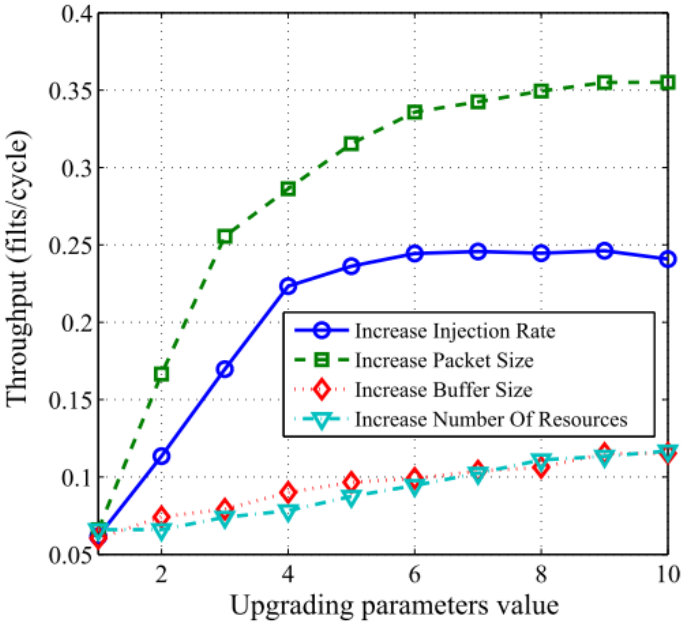


Figure 3.4: The impact of upgrading value of injection rate, packet size, buffer size and number of resources on upgrading the throughput

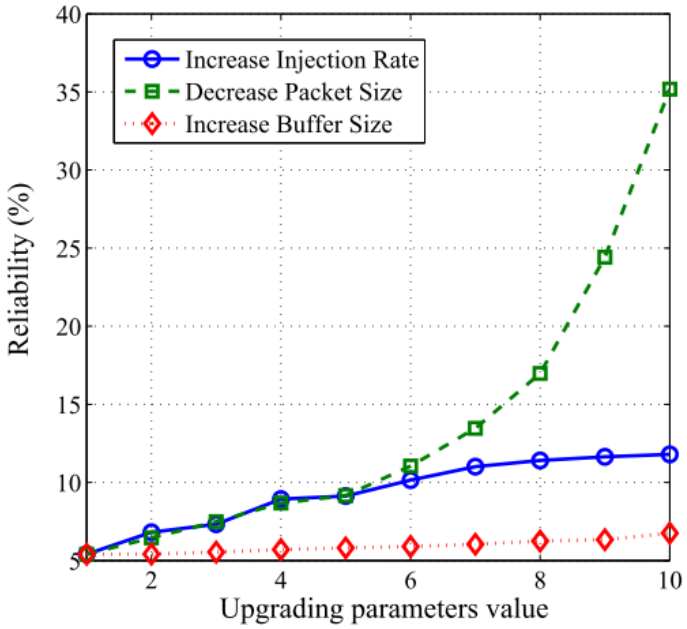


Figure 3.5: The impact of upgrading value of injection rate, packet size and buffer size on upgrading the reliability

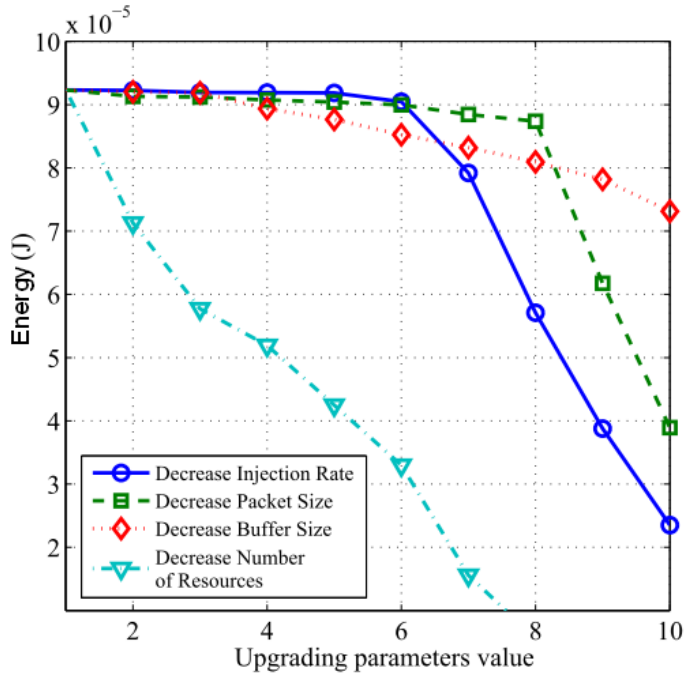


Figure 3.6: Energy consumption over different values of injection rate, packet size, buffer size and number of resources

Increasing the packet size is the best decision to adapt the worst condition caused by the degradation of throughput. It give more effect on the increasing of throughput performance than increasing injection rate, buffer size or number of resources values as shown in Figure 3.4.

In adapting the worst network condition caused by degrading the reliability performances, decreasing packet size gives more effect on increasing reliability than adjusting other parameters value as shown in Figure 3.5. While to adapt the increasing power consumption in the system, decreasing number of active resources or shut down inactive resources is the best decision due to in fact that each resources consume energy as shown in Figure 3.6.

3.4.4 Best parameters type on performances

Each NoC parameters such as routing algorithm and selection path strategy has their own technique (Table 3.9). Each technique has different output performance. In this subsection we simulate each technique then evaluate the best technique based on the output performance. We compared the performances of six routing algorithm and three selection path strategy provided in Noxim simulator.

Table 3.9: The different technique of parameters to evaluate the impact on performance

Parameter	Technique
Routing algorithm	XY, Westfirst, Negativefirst, Northlast, Oddeven, Fullyadaptive
Selection path strategy	Random, Buffer level, Neighbour on path

The evaluation of latency in worst condition shows that Fully-adaptive routing algorithm (Figure 3.7) and Neighbors-on-Path (NoP) of selection path strategy (Figure 3.8) has lower latency than other technique of parameters. The selection of this technique (fully-adaptive and NoP) are suitable to adapt the degradation QoS in worst condition.

In worst condition, Fully-adaptive routing (Figure 3.9) and Buffer-level selection

strategy (Figure 3.10) have lower degradation on throughput compared to other parameters, thus these parameters are suitable to adapt the condition of minimum QoS in presence of faults in network.

Whereas, Fully-adaptive routing (Figure 3.11) and NoP selection strategy (Figure 3.12) are the best choice to adapt the worst QoS condition caused by degradation reliability.

The results show that the best adaptation to the increase of power consumption are by choosing Fully-adaptive routing algorithm (Figure 3.13) and NoP selection strategy (Figure 3.14) while the biggest effect on decreasing the power consumption is by decreasing number of resources as shown in Figure 3.6.

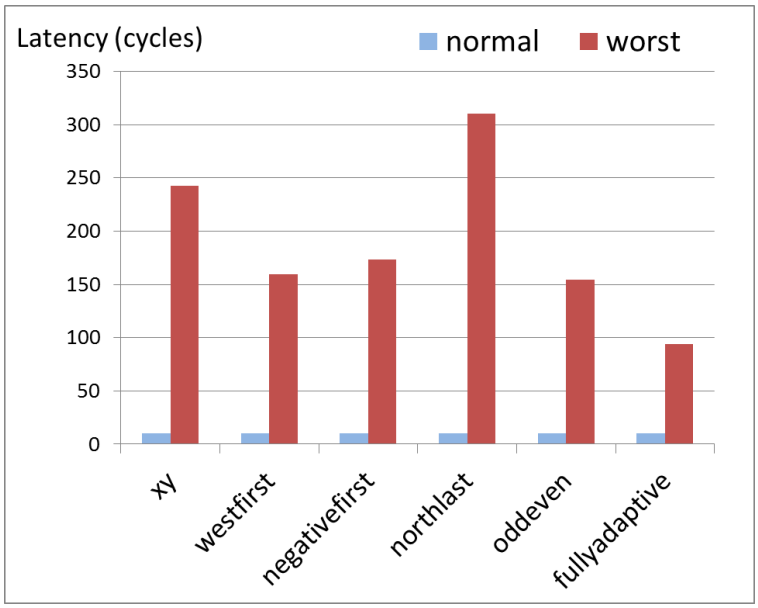


Figure 3.7: The impact of the routing algorithm on latency

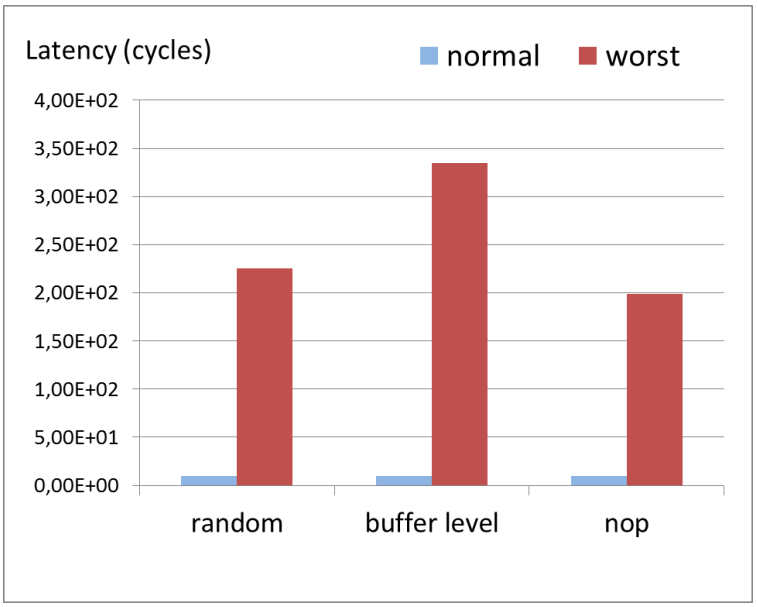


Figure 3.8: Selection-path strategy impact on latency

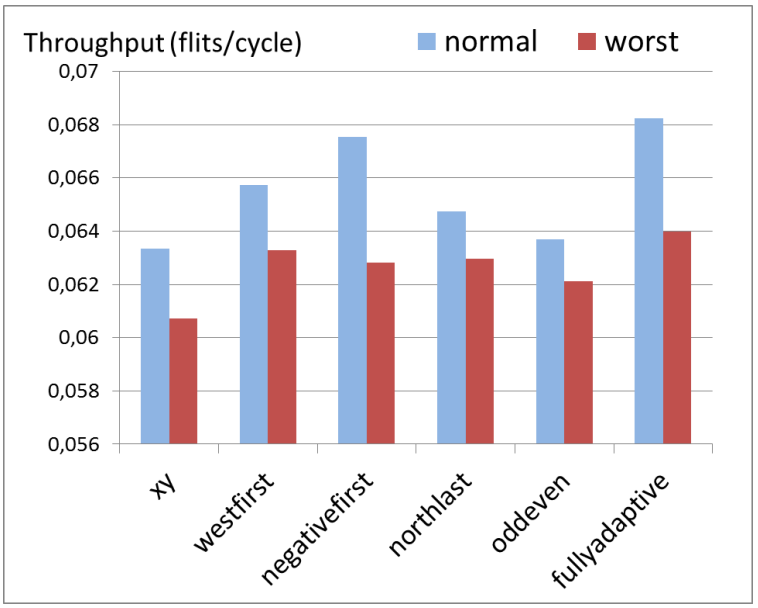


Figure 3.9: The impact of the routing algorithm on throughput

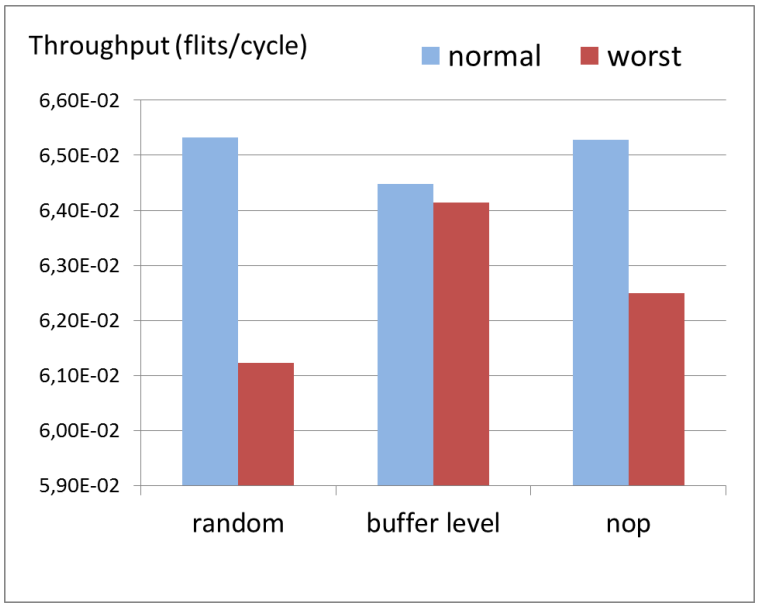


Figure 3.10: Selection-path strategy impact on throughput

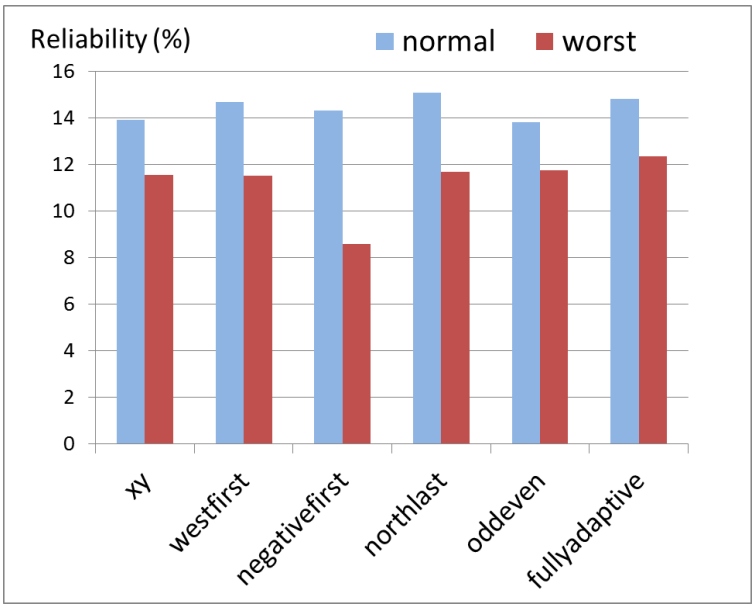


Figure 3.11: The impact of the routing algorithm on reliability

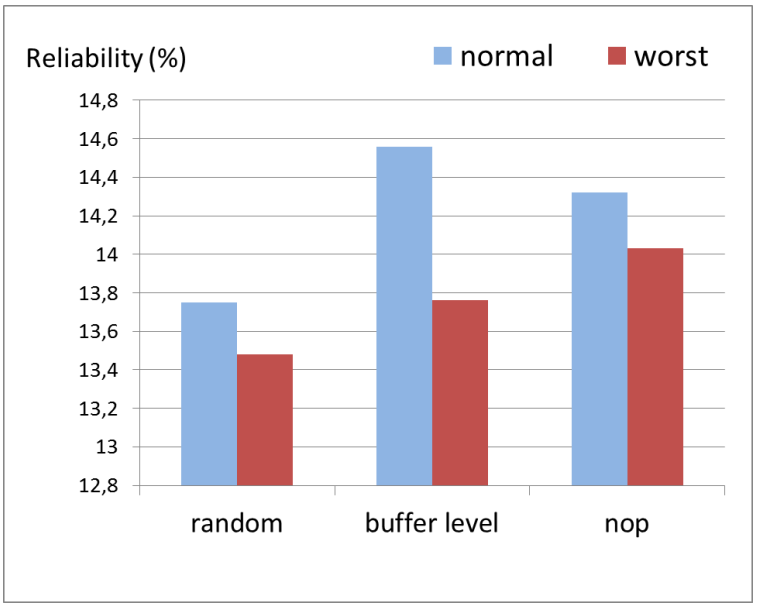


Figure 3.12: Selection path strategy impact on reliability

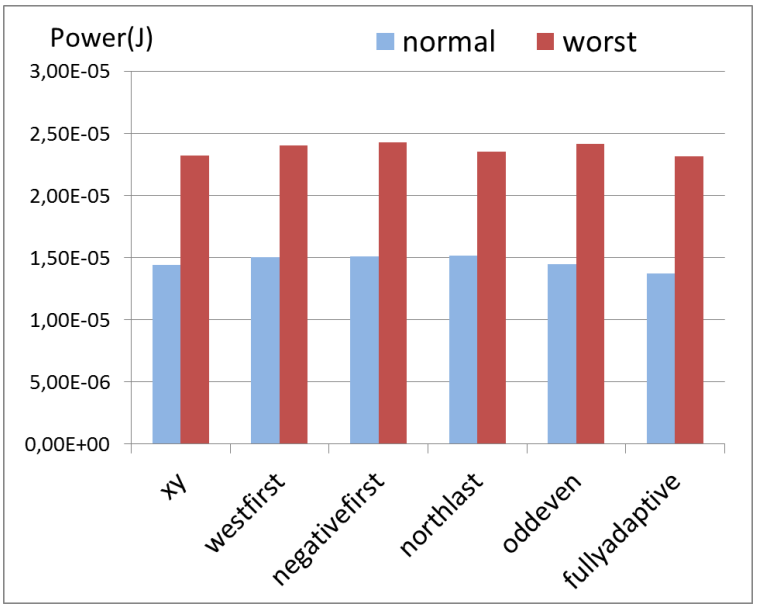


Figure 3.13: The impact of the routing algorithm on power consumption

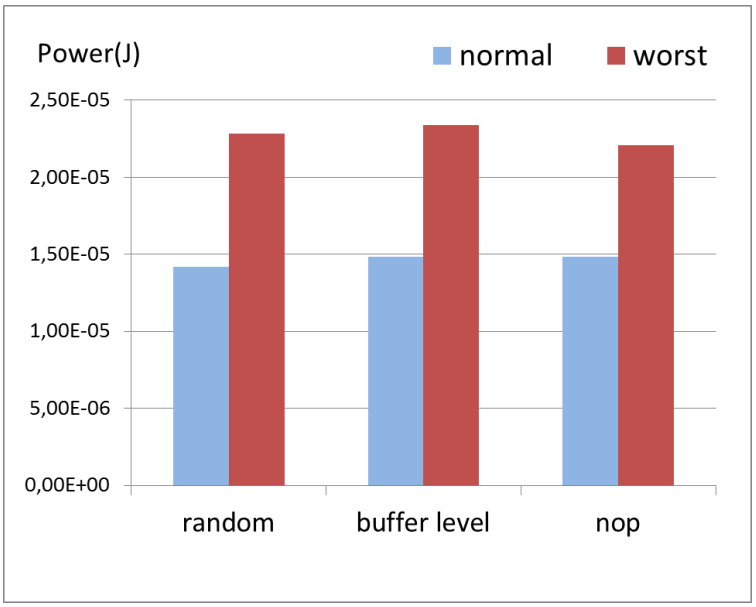


Figure 3.14: Selection path strategy impact on power consumption

3.5 Conclusions

Designing an adaptive NoC that can adapt the QoS needed by different application requirements is very time consuming. A challenge facing designers of SoCs containing NoC is to find NoC instances that balance the costs (e.g. area) and performances (e.g. latency and throughput). In this chapter, the worst condition scenarios, the evaluation of saturation network and the evaluation of the impact of NoC parameters design parameters on the performance have been presented. Worst condition is used as the first state of network before adjusting the parameters. While the combination of parameters that causing network saturation is avoided in the evaluation of impact parameter on performance. Some scenarios have been presented to see how big the impact of upgrading parameters to the performances was defined.

The evaluation shows that latency is the most influencing indicator of NoC performance to provide QoS over the other network performance. In simulation, to adapt the degradation of QoS caused by increasing latency, decreasing packet injection rate and decreasing packet size has biggest impact than increasing buffer size or decreasing num-

Table 3.10: Summarize the most influence parameter to NoC performances

Performance	Injection rate	Packet size	Buffer size	Number of resources
Latency	1st	2nd	4rd	3rd
Throughput	2nd	1st	3rd	4rd
Reliability	2nd	1st	3rd	-
Power consumption	2nd	3rd	4rd	1st

ber of resources as presented in Table 3.10. In worst case scenario, the Fully adaptive routing algorithm and NoP selection path strategy has the lowest latency than their counterparts.

In term of impact of parameter on the throughput performance, as presented in Table 3.10, increasing packet size have biggest impact on increasing throughput performance then followed by increasing injection rate. While increasing buffer size and number of resources not give big impact on throughput performance. For other parameter such as routing algorithm, Fully adaptive routing algorithm can adapt the decrease of throughput in worst condition.

In reliability performances, as presented in Table 3.10, decreasing injection rate can increase reliability performances compared to increasing injection rate or increasing buffer size. While Fully-adaptive routing algorithm and NoP selection path strategy can adapt the worst condition that cause decreasing reliability.

In term of power consumption, the fastest way to adapt the increasing power consumption is by decreasing the number of resources as presented in Table 3.10. Almost all routing algorithm type and selection path strategy have almost the same power consumption.

The results of this work can be used by NoC system designer as a guideline to estimate the system performance, related parameters and associated overhead in designing an adaptive NoC. In real-life application, all parameters we used in simulation (i.e. packet size, buffer size, routing algorithm and packet rate) are determined in a NoC router. Thus, reconfigurable router is needed to adapt the condition of network and the

performance degradation. In case of changing condition in network caused by fault, for example, the reconfigurable router can adapt the NoC by adjusting the packet size and packet rate, or change the type of its routing algorithm.

Chapter 4

Fault tolerant routing algorithm for 2D and 3D mesh network

As stated in chapter-1, deadlock packets will occurs when packet waits for resources that will never be released caused by the faulty link or node. *Gradient* and *Diagonal* algorithms handle the deadlock packet in the networks by choosing the direction which have more alternative path in avoiding the faults. Further, *Gradient* and *Diagonal* router implemented wormhole switching to have better latency and lower buffer use due to the header flits is forwarded without waiting next flit arrives.

This chapter presents adaptive fault-tolerant routing algorithms for 2D mesh called *Gradient* and 3D mesh called *Diagonal*. Both algorithms are designed to avoid multiple links and node failures that cause deadlock in mesh NoCs. The novel feature of these proposed algorithms is on how they classifies the destination node and how they chooses the decision sequence of alternative path while avoiding the faults in both 2D and 3D NoCs. Both Gradient and Diagonal algorithm has been simulated in systemC based Noxim simulator and then compared with existing related routing algorithm.

4.1 Gradient: Fault-tolerant Routing Algorithm for 2D Mesh Topology

Gradient considers sequences of alternative paths for packets when the main path fails. The proposed algorithm tolerates faults in worst condition traffic in NoCs. The main difference with existing 2D mesh routing algorithms relies to the fact that divide 2D coordinates into four destination zone based on vertical and horizontal lines, *Gradient* algorithm classify the destination address of packets in the network into eight zones based on a gradient line.

To evaluate the performance of the proposed algorithm, scenarios with various link-faults and node failures schemes were created. Hence the number of hops to destination nodes and the number of alternative paths in faulty network are determined and compared with other 2D mesh routing algorithms. Further, we implemented *Gradient* in the Noxim simulator to evaluate its latency and throughput performances.

4.1.1 Gradient algorithm

An accurate selection of alternative routes can avoid a large number of hops for packet to reach destination. Therefore, the proposed routing algorithm has more alternative routes with minimum hops than other existing fault-tolerant adaptive routing algorithm. Thus, the method proposed is adaptive and fault-tolerant for 2D mesh topology.

The algorithm classify the destination address of packets in the network into eight zones based on gradient line (M) in 2D coordinate as presented in Figure 4.1. Gradient line (M) is a number that represents the steepness of a straight line and is obtained from the value of destination address (Dx, Dy) and the current router address (Cx, Cy) as shown in equation 4.1

$$M = \frac{D_y - C_y}{D_x - C_x} \tag{4.1}$$

We then use the gradient line $|M| = 1$ to divides the 2D coordinates of network into eight zones as shown in Figure 4.1. Based on this relative positioning the destination address is assigned to a zone to determine the next hop (i.e. the next router in the

path).

For each zone of the destination node for the current packet, the *Gradient* algorithm defines one main route and three alternatives ones. The decision is then taken at run-time depending on network conditions. For example, for a destination in *Zone-1* (Figure 4.2-a), the main route is east (No.1) which provide the shortest path to the destination. The first alternative route is north (No.2) which can also ensure a shortest path to destination and finally south (No.3), if there is no possible hop in the first paths. The different possible decision depending on the zones is shown in Figure 4.2.

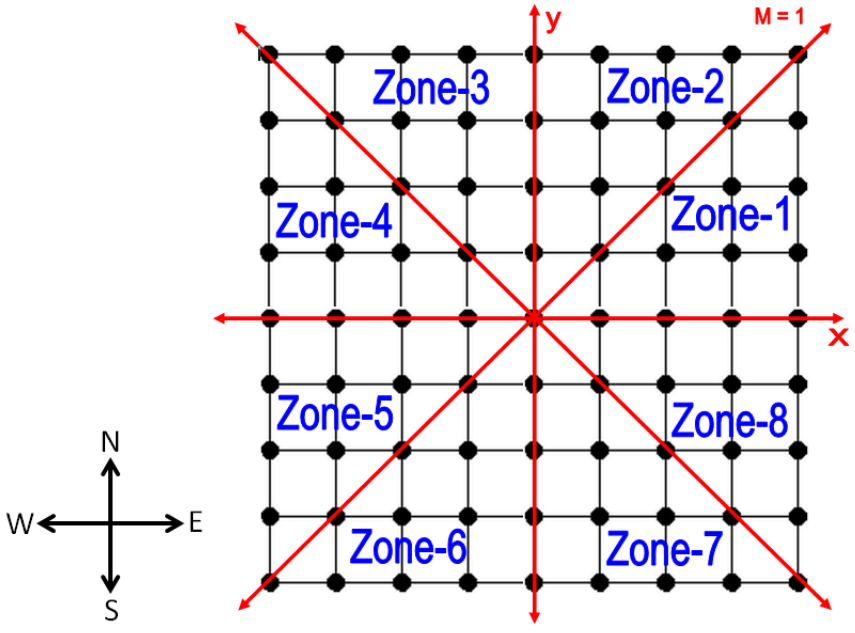


Figure 4.1: Gradient concept divide 2D coordinate into eight zones based on gradient line

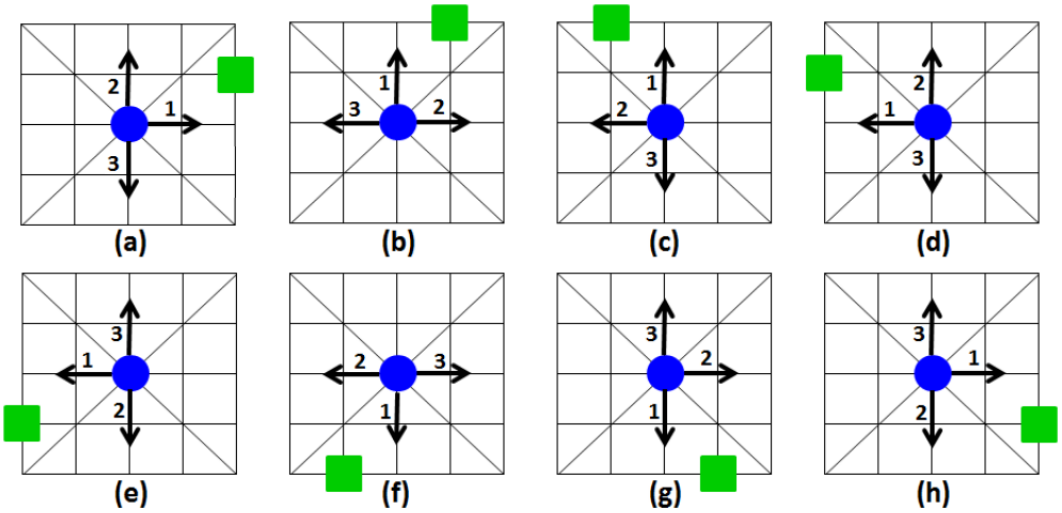


Figure 4.2: Sequence decision of *Gradient* algorithm for a destination in (a) Zone-1, (b) Zone-2, (c) Zone-3, (d) Zone-4, (e) Zone-5, (f) Zone-6, (g) Zone-7, (h) Zone-8

Classical fault-tolerant routing algorithms classify destination nodes into only four zones, based on vertical and horizontal lines in two dimensions. The number of alternative paths is then reduced. The main difference relies also on the choice sequence of the alternative routes. The main weakness of existing routing algorithms is that if there are faults in the main route and in the first alternative route the packet will choose a longer distance path to reach the destination node. Thus increasing the number of hops and degrading the network performances.

Gradient is independent of fault detection. The position of fault and the hotspot percentage of node in the network are fixed in our work. Fault detection and localization is out of the scope of the work. To avoid a hot-spot area or a faulty element in the network, *Gradient* chooses the shortest alternative route. The selection of this alternative path makes the distance to destination node closer than other existing fault-tolerant 2D mesh routing algorithm. The shortest distance of alternative path will avoid the degradation of network performance such as latency and throughput.

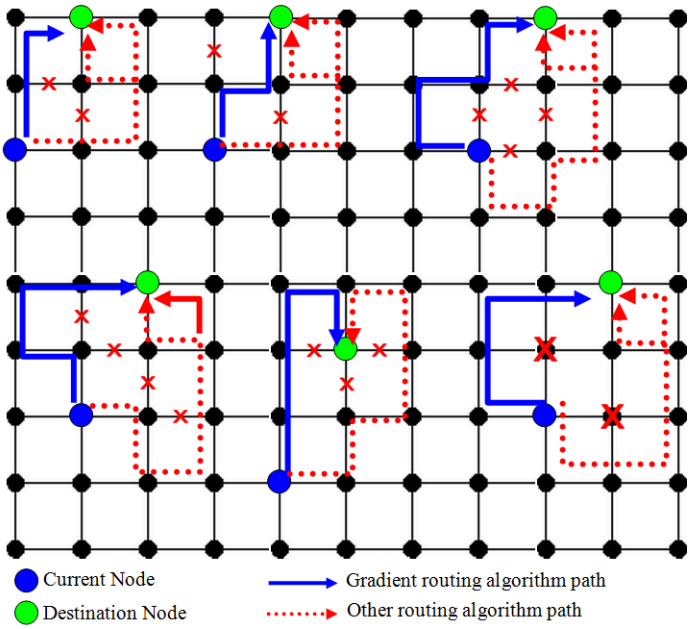
Figure 4.3 shows that dividing the network in eight zones leads *Gradient* to choose shortest path than turn model and fully adaptive routing algorithms. In Figure 4.3-a, the destination node is supposed to be in north-east of the current node. The *Gradient*

algorithm then classifies this destination node in Zone-2. In proposed algorithm the main route is north, while for the other algorithms (north-last, and fault-tolerant [11]) the main route is east. In case of different fault scenarios shown in Figure 4.3, the different decision on the main-route will affects the number of hops to reach the destination. Moreover, in case of link-faults on north and east port of current node, the alternative route of our algorithm is west but other algorithms will choose the south. This is the second advantage of *Gradient* decision because the packet chooses the correct direction to avoid the faulty-link. As we can see, the packet with *Gradient* has always the shortest distance path to reach destination node. This is due to the increased number of alternative paths available when there are problems on the main route. This in turn offers a more precise localization of destination and faults. We can show the same behavior on different zone, such as in the case of the destination node is on the south-west of current node as shown in Figure 4.3-b. Here also demonstrate that in worst-case Gradient operate the same than its counterpart.

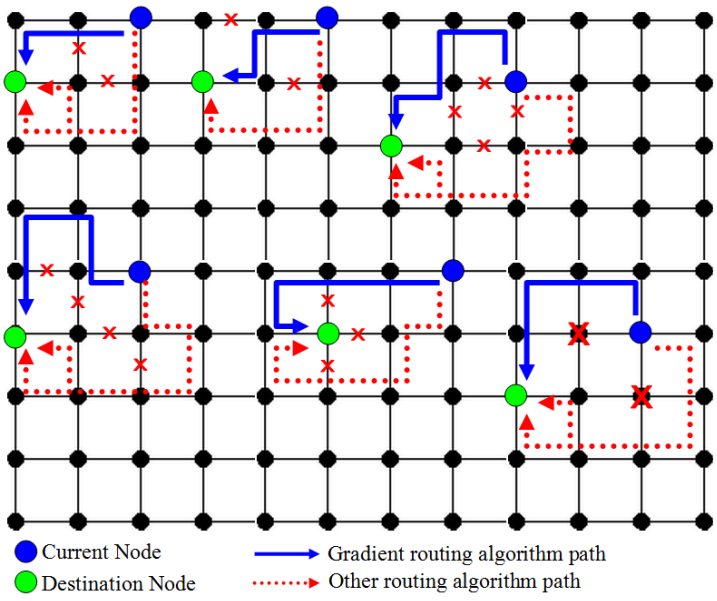
The pseudo code of the proposed algorithm is shown in Figure 4.4. According to the current position of the router and the address of the packet destination, it first divides the zone of destination node into the eight zone based on *Gradient* line M. The M line is obtained by using equation 4.1. In the algorithm, $abs.x$ is the absolute value of $Dx - Cx$ and $abs.y$ is the absolute value of $Dy - Cy$. The relation $abs.y > abs.x$ is representative of $|M| > 1$ and $abs.y < abs.x$ is representative of $|M| < 1$. Depending on the value of M and the relative position of the destination node, the algorithm then computes the next hop and propose a main route and two associated alternative paths.

4.1.2 Evaluation of minimum hops and alternative path

The objective of adaptive routing is to tolerate the faults that may cause packet deadlock and hence degrade the performance in NoC. But we also need to sustain network performance such as bandwidth and latency of communications. We then evaluate four measures of performance for the different routing algorithms: i) number of hops, which reflect the length of the route and penalty due to faulty link, ii) number of alternative



(a) Comparison decision path between Gradient algorithm and other routing algorithm possibility for destination packet in zone-2 of gradient zoning



(b) Comparison decision path between Gradient algorithm and other routing algorithm possibility for destination packet in zone-5 of gradient zoning

Figure 4.3: Alternative path of different routing algorithms in presence of faults in the network for different relative position of the destination node

```
Algorithm: Gradient
Input:   Coordinate of current node (current.x, current.y)
         Coordinate of destination node (destination.x, destination.y)
         Destination based on Gradient (abs_x, abs_y)
         abs_x = |destination.x - current.x|
         abs_y = |destination.y - current.y|  // abs_y ≥ abs_x  ( M ≥ 1)
Output:  Selected route
Procedure:
// destination on current node
if (destination.y == current.y && destination.x == current.x) {
                              main_route(DIRECTION_LOCAL);
// destination on zone-1
else if (destination.y > current.y && destination.x >= current.x && abs_x >= abs_y) {
                              main_route(DIRECTION_EAST);
                              first_alternative(DIRECTION_NORTH);
                              second_alternative(DIRECTION_SOUTH);}
// destination on zone-2
else if (destination.y >= current.y && destination.x > current.x && abs_x < abs_y) {
                              main_route(DIRECTION_NORTH);
                              first_alternative(DIRECTION_EAST);
                              second_alternative(DIRECTION_WEST);}
// destination on zone-3
else if (destination.y >= current.y && destination.x < current.x && abs_x < abs_y) {
                              main_route(DIRECTION_NORTH);
                              first_alternative(DIRECTION_WEST);
                              second_alternative(DIRECTION_EAST);}
// destination on zone-4
else if (destination.y > current.y && destination.x <= current.x && abs_x >= abs_y) {
                              main_route(DIRECTION_WEST);
                              first_alternative(DIRECTION_NORTH);
                              second_alternative(DIRECTION_SOUTH);}
// destination on zone-5
else if (destination.y < current.y && destination.x <= current.x && abs_x >= abs_y) {
                              main_route(DIRECTION_WEST);
                              first_alternative(DIRECTION_SOUTH);
                              second_alternative(DIRECTION_NORTH);}
// destination on zone-6
else if (destination.y <= current.y && destination.x < current.x && abs_x < abs_y) {
                              main_route(DIRECTION_SOUTH);
                              first_alternative(DIRECTION_WEST);
                              second_alternative(DIRECTION_EAST);}
// destination on zone-7
else if (destination.y <= current.y && destination.x > current.x && abs_x < abs_y) {
                              main_route(DIRECTION_SOUTH);
                              first_alternative(DIRECTION_EAST);
                              second_alternative(DIRECTION_WEST);}
// destination on zone-8
else {
                              main_route(DIRECTION_EAST);
                              first_alternative(DIRECTION_SOUTH);
                              second_alternative(DIRECTION_WEST);}
```

Figure 4.4: The abstraction of *Gradient* routing algorithm

paths, which represent the possibility of adaptive routing to avoid the faults or hot-spots in the network, iii) latency, related to the number of hop and iv) the throughput, to analyze performance of the network.

4.1.2.1 Scenarios

We have evaluated eight different routing algorithms, namely XY, West First, North Last, Negative First, Odd Even and Fully Adaptive that are implemented and supported by the Noxim simulator [51], the Fault Tolerant algorithm presented in [11] and *Gradient*. All these algorithms were evaluated under twenty different scenarios described in Figure 4.5.

These scenarios are designed to represents all fault position possibility and all destination node direction possibility in 2-D mesh topology. For example scenarios 1 to 4 in Figure 4.5, the destination nodes are in the west, east, north and south from the source node, thus they represents horizontal and vertical direction with a faulty link between them. Thus, these scenarios also represent horizontal and vertical possibility of fault position. In scenarios 5 to 20, the destination nodes are in west-north, east-north, west-south, and east-south of current node. Thus, its represents all possible positions in addition to vertical and horizontal direction.

As described in Figure 4.5, Scenarios 1 to 12 has only one faulty-link between current node and destination while scenarios 13 to 20 presents at least two faults on link or node on the route between the current node and the destination one. Thus the evaluation result of number of hops and the number of alternative paths of each routing algorithm is validated.

In chapter 2 has been stated that each routing algorithm has their own weakness in deciding the direction of packet to neighbour node. For example north-last algorithm, has no alternative route when the destination is in the north side of current node and there is a faulty link in between. A deadlock packet also occurs in west-first routing algorithm if the destination node is in the west side of current node with a fault in between. The objectives of these scenarios are to shows the comparison of different

path due to different routing decision.

4.1.2.2 Evaluation result

Table 4.1 summarizes the evaluation result of number of hops of each routing algorithm for scenario 1 to 12 as presented in Figure 4.5. In scenarios 1 to 12, we design one faulty link between source node and destination node in different direction. The result in Table 4.1 shows that west-first algorithm is deadlock for scenario 1, north-last for scenario 3, negative first and odd-even for scenario 4. While fully adaptive [51], fault tolerant [11] and Gradient routing algorithm have 3 numbers of minimum hops without deadlock in scenarios 1 to 4.

Classically, conventional routing algorithms divide 2D mesh topology into 4 zones: northeast, northwest, southwest and southeast, while our algorithm divides 2D mesh topology into 8 zones. In scenarios 5 to 12, we evaluated the path of packets for a destination node in the north-west, north-east, south-west, and south-east from the current node, with a link fault in horizontal and vertical line.

In scenario-1, algorithm North Last, Odd Even, Fully Adaptive, Fault Tolerant and *Gradient* has a minimum number of hops of 3 nodes, but algorithm XY, West First and Negative First cannot avoid the faulty link on the west side of the current node, thus causing packet deadlock and hence degradation of NoC performance. As expected, the deterministic algorithm fails in adapting the path depending on the position of the fault. Thus, it leads to deadlocks even with only one fault in the NoC. The results in color blue also show that, in scenarios 1 to 12, adaptive routing algorithms (fully-adaptive, fault-tolerant and *Gradient*) can avoid all faults (no deadlock) and have the same number of minimum hops.

We can see that in these scenarios there are three routing algorithms that have the same minimum number of hops. We then evaluate the number of alternative routes supported. We evaluate this metric only for adaptive routing, since the first algorithms are deterministic and offer only one path from a source to a destination. Table 4.2

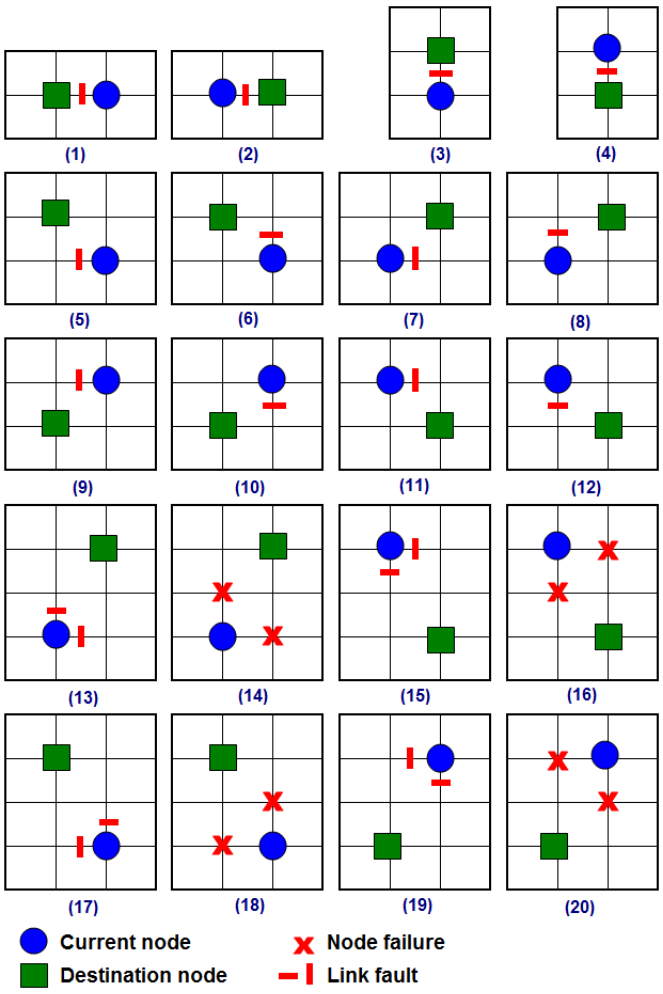


Figure 4.5: Twenty conditions of link-faults and node failures to evaluate the minimum hop and number of alternative path for seven different routing algorithms

Table 4.1: Number of hops for each algorithm in the presence of one fault in the network

Scenario	Turn model				Adaptive		
	West First	North Last	Neg. First	Odd Even	Fully Adaptive [51]	Fault Tolerant [11]	Gradient
1	DL	3	DL	3	**3**	**3**	**3**
2	3	3	3	DL	**3**	**3**	**3**
3	3	DL	3	DL	**3**	**3**	**3**
4	3	5	DL	DL	**3**	**3**	**3**
5	DL	3	DL	2	**2**	**2**	**2**
6	2	2	2	2	**2**	**2**	**2**
7	2	3	2	2	**2**	**2**	**2**
8	2	2	2	DL	**2**	**2**	**2**
9	DL	2	2	2	**2**	**2**	**2**
10	2	DL	2	2	**2**	**2**	**2**
11	2	3	2	2	**2**	**2**	**2**
12	2	DL	DL	2	**2**	**2**	**2**

DL = Deadlock; WF = West First; NL = North Last; NF = Negative First; OE = Odd Even; Gr = Gradient

shows that in scenarios 1 to 12, all adaptive routing algorithms have the same number of hops (same results than previously analyzed) and the same number of alternative paths. In presence of only one fault the algorithms are equivalent. However in scenarios 13 to 20, which correspond to two simultaneous faults in the network, the fully adaptive routing algorithms cannot avoid deadlock and then blocked communications appears. In scenarios 13, 15, 17 and 19, the *Gradient* algorithm has more alternative paths than the fault tolerant routing algorithm, however both of them have the same number of minimum hops. This shows that *Gradient* has more possibilities to tolerate faults than its counterpart. *Gradient* also has lower number of hops in scenario 14, 16, 18 and 20. Thus, *Gradient* algorithm has smallest number of hops and more alternative routes compared to other existing fault-tolerant routing algorithms. In the next chapter, the methodology, the simulation scenarios and the experimental result of Gradient routing algorithm are also presented. Further, the implementation of Gradient routing algorithm into on FPGA is also presented.

For network performances evaluation, we modified the Noxim simulator to support failures as representative of worst-case scenarios. By this method, we were able to evaluate and compare the performance metrics of latency and throughput of the studied algorithms.

4.2 Diagonal: Fault-tolerant routing algorithm for 3D mesh topology

The conventional 2D integrated circuit (IC) has limited floorplanning choices, and consequently, it limits the performance enhancements arising out of NoC architectures. Three-dimensional (3D) ICs are capable of achieving better performance, functionality, and packaging density compared to more traditional planar ICs [65].

In this work we propose an adaptive fault-tolerant routing algorithm for 3D mesh NoC. The novel feature of this algorithm is on how the algorithm chooses the sequence of alternative paths when the packet faces a faulty link or node in the 3D mesh NoCs.

Table 4.2: Comparison of number of minimum hops for the three adaptive routing algorithms and optimal calculation by hand

Scenario	Fully adaptive [51]		Fault Tolerant [11]		*Gradient*		Optimum calculated by hand	
	mh	nap	mh	nap	mh	nap	mh	nap
1	3	2	3	2	3	2	3	2
2	3	2	3	2	3	2	3	2
3	3	2	3	2	3	2	3	2
4	3	2	3	2	3	2	3	2
5	2	1	2	1	2	1	2	1
6	2	1	2	1	2	1	2	1
7	2	1	2	1	2	1	2	1
8	2	1	2	1	2	1	2	1
9	2	1	2	1	2	1	2	1
10	2	1	2	1	2	1	2	1
11	2	1	2	1	2	1	2	1
12	2	1	2	1	2	1	2	1
13	DL	DL	5	**1**	5	**3**	5	4
14	DL	DL	**7**	2	**5**	1	5	1
15	DL	DL	5	**1**	5	**3**	5	4
16	DL	DL	**7**	2	**5**	1	5	1
17	DL	DL	5	**1**	5	**3**	5	4
18	DL	DL	**7**	2	**5**	1	5	1
19	DL	DL	5	**1**	5	**3**	5	4
20	DL	DL	**7**	2	**5**	1	5	1

DL: deadlock, mh : minimum hops, nap: number of alternative path

The decision sequence of Diagonal is based on distances and directions estimation of destination node from the current node. It chooses the farthest destination for the main route or a second alternative and then chooses the shortest distance in opposite direction for third to fifth alternative routes.

Some works address the routing problem for 3D mesh networks other than using the basic dimension order routing approach. Most of the existing 3D mesh routing algorithms define the destination into horizontal and vertical layer. Some of them reuse existing 2D mesh routing algorithm for horizontal destination such as XY or YX. For example [61], proposed a deterministic routing scheme for choosing the 3D layer and it uses the XY routing within the 2D layers. The RPM routing algorithm [59] uses Z as vertical dimension and XY as "horizontal" dimensions. It first routes a packet in the minimal direction to a random intermediate Z position then routes the packet on the XY plane using either minimal XY or YX routing.

On 3D NoCs, [66] proposed a load balancing routing scheme in which a packet is sent to a random layer then use XY algorithm to reach horizontal destination and finally traverse vertical direction to final destination. An architecture called XNOTs [46] proposed an algorithm which is based on the idea of vertical switching. However, it requires large area switches, which are costly since crossbar area grows quadratically with the number of ports. In router, a 4NP-First routing scheme in [53] extend the 2D west-first, negative-first and north-last turn models into 2 negative-first (2N-first), 3 negative-first (3N-first), and 4 negative-first (4N-first) turn models in 3D. A Quadrant-XYZ dimension order routing algorithm [34] partitions the geometrical space of torus 3D NoC topology into quadrants and selects the nearest wrap-around edge to connect the destination node. A reconfigurable inter-layer routing mechanism (RILM) for 3D NoCs has been proposed in [62]. It can tolerate high number of vertical link failures by moving the message firstly inside the current layer to reach other vertical link near destination. An Elevator-First distributed routing algorithm [16] proposed a deadlock and live-lock free algorithm by using two virtual channels (one for ascending and the other for descending packets) for the case of deterministic and deadlock-free planar

routing schemes. The author in [74, 76] proposed a fault-tolerant routing algorithm by dividing the 3-D mesh into eight sub-networks.

A fully adaptive routing algorithm for 3D NoCs named DyXYZ [17] has been proposed as a solution of the degradation in performance in deterministic routing algorithm. This routing algorithm is similar to our proposed 3D routing algorithm called AdaptiveXYZ. In this work we also designed a simple adaptive routing algorithm for 3D mesh called adaptive-XYZ. This algorithm is the adaptive version of static dimension order routing algorithm XYZ to tolerate faulty network condition.

In adaptive-XYZ routing algorithm, a packet first traverses along the X dimension, then Y dimension and last in Z dimension. If the X dimension is faulty, the packet continues to Y then continue traverse X dimension again. If the X and Y dimension fail, the packet continue to Z dimension as shown on Figure 4.8-b. We use adaptive-XYZ algorithm and elevator first algorithm as comparisons to our proposed algorithm. We used AdaptiveXYZ routing algorithm in the scenario to compare the output performance with Diagonal routing algorithm.

In this work we propose a routing method to tolerate both faulty links and faulty routers for 3D mesh NoC. This method is based on distances and directions of destination node positioning in the 3D mesh NoC. It divides destination node position into 48 zones. The novel feature of this method is on how it classifies the destination node and the sequence decision of alternatives routes when the main route fails. The accuracy in selecting the alternative route can avoid increasing the required number of hops. Therefore, proposed routing algorithm has minimum hops compared to other existing 3D mesh routing algorithm. For this purpose, we extend Noxim[19] to support 3D mesh topology and then implement the proposed algorithm. We then evaluate and compare the latency performance using worst-case scenario condition. The results presented in this part take into account the performances of fault-tolerant routing algorithm. The latency performance is also evaluated based on simulation and is compared to adaptive XYZ routing algorithm, a modified algorithm developed for comparison purpose.

4.2.1 Diagonal algorithm

The proposed algorithm considers the distances and the directions of destination nodes based on the current node position. Figure 4.6 shows how we define the distances and the directions of destination-1 (D1) and destination-2 (D2) based on current address (C). The distances are obtained from the ratio between absolutes value of $\triangle X$, $\triangle Y$ and $\triangle Z$. Based on these distances, Diagonal choose the farthest distance as the first alternative and shortest distance as the last alternative. While the directions are obtained from the value of $\triangle X$, $\triangle Y$ and $\triangle Z$. As an example, if the value of $\triangle X$, $\triangle Y$ and $\triangle Z$ are positive, the algorithm will chooses X+, Y+, Z+ and vice versa. There are six possibilities to combine the distances of $|\triangle$X|, $|\triangle$Y|, $|\triangle$Z| and eight possibilities to combine the direction of X(+/-), Y(+/-) and Z(+/-) as shown on Table 4.3. We then combine the possibilities of distances and directions into a total of 48 possibilities as shown in Appendix A.

The algorithm chooses the farthest distance for the main route, the second farthest distance for the alternative-1 and third farthest for the alternative-2. In case of a fault on main route, the packet then chooses alternative-1. If the route of alternative-1 also fails, the packet then chooses alternative-2. The novel feature of this algorithm is on how the algorithm chooses the route for alternative-3. When the alternative-2 fails, the algorithm chooses the shortest distance with opposite direction for the alternative-3 as shown in Appendix A. Based on the defined zones, the algorithm has different decision sequence of main route and alternatives routes. The goal of this method is to have minimum hops and more alternative route for the packet to reach the destination node. For example, if the distance of destination-X from current-X ($|\triangle X|$) is farther than $|\triangle Y|$ and $|\triangle Y|$ is farther than $|\triangle Z|$, the algorithm will chooses X as the main route, Y as the first alternative, Z as the second alternative, -Z as the third alternative, -Y as the fourth alternative and finally -X as the fifth alternative route.

Table 4.3: Combination possibilities of (a) distances and (b) directions based

(a) Distances based

No	Combination possibilities
1	$\|\triangle X\| \geq \|\triangle Y\| \geq \|\triangle Z\|$
2	$\|\triangle X\| \geq \|\triangle Z\| \geq \|\triangle Y\|$
3	$\|\triangle Y\| \geq \|\triangle X\| \geq \|\triangle Z\|$
4	$\|\triangle Y\| \geq \|\triangle Z\| \geq \|\triangle X\|$
5	$\|\triangle Z\| \geq \|\triangle X\| \geq \|\triangle Y\|$
6	$\|\triangle Z\| \geq \|\triangle Y\| \geq \|\triangle X\|$

(b) Direction based

No	Combination possibilities
1	$\triangle X(+), \triangle Y(+), \triangle Z(+)$
2	$\triangle X(+), \triangle Y(+), \triangle Z(-)$
3	$\triangle X(+), \triangle Y(-), \triangle Z(+)$
4	$\triangle X(+), \triangle Y(-), \triangle Z(-)$
5	$\triangle X(-), \triangle Y(+), \triangle Z(+)$
6	$\triangle X(-), \triangle Y(+), \triangle Z(-)$
7	$\triangle X(-), \triangle Y(-), \triangle Z(+)$
8	$\triangle X(-), \triangle Y(-), \triangle Z(-)$

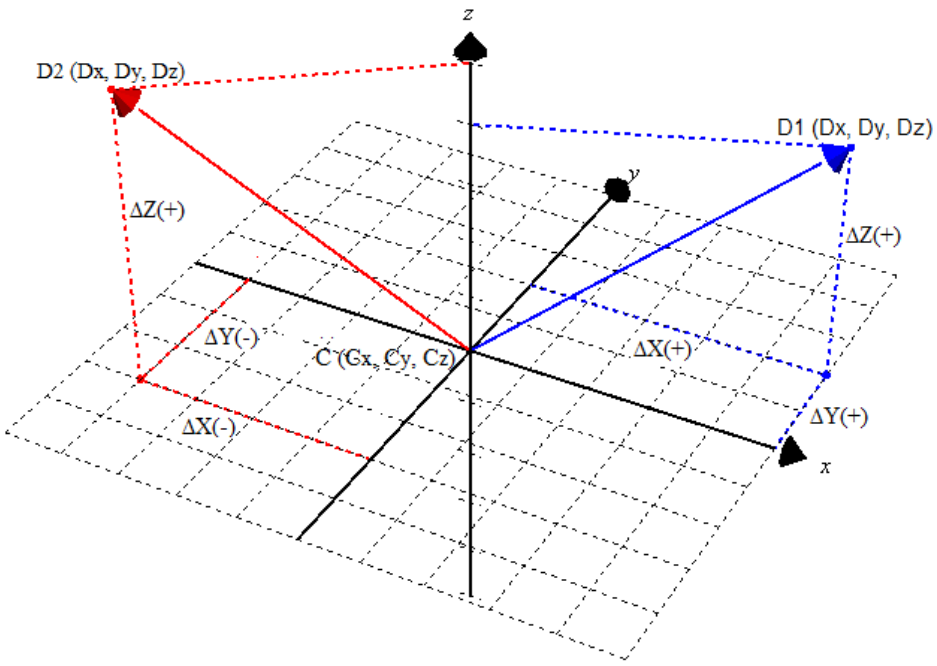
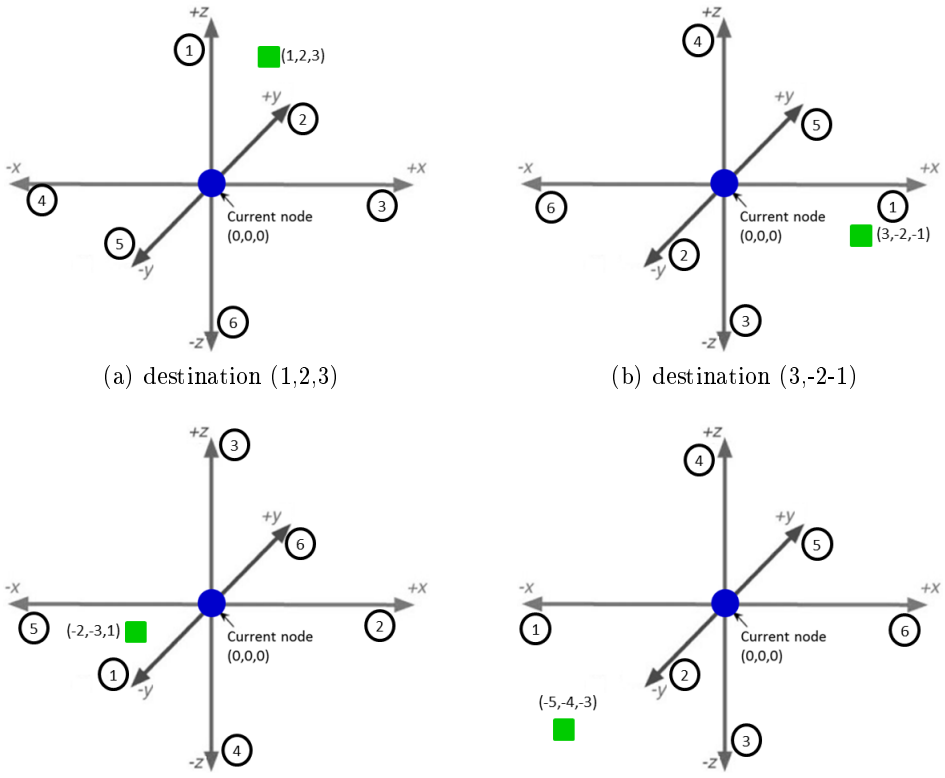


Figure 4.6: Addressing the position of node on 3D coordinate

Figure 4.7 presents some examples of sequence decision taking of *Diagonal* algorithm for some coordinates of destinations with coordinate of current node [0,0,0] in 3D



(a) destination (1,2,3)

(b) destination (3,-2-1)

(c) destination (-2,-3,1)

(d) destination (-5,-4,-3)

Figure 4.7: Sequence decision of *Diagonal* algorithm for destination coordinate on (a) [1,2,3], (b) [3,-2-1], (c) [-2,-3,1] and (d) [-5,-4,-3]

dimension. For destination node in coordinate [1,2,3] (Figure 4.7-a), the main route is Z-dimension which is the farthest distance than X-dimension (east) and Y-dimension. If the main route is faulty, the algorithm then chooses Y-dimension due to the fact that Y-dimension is farther than X-dimension. Another example of decision taking of *Diagonal* algorithm are shown in Figure 4.7-b (for coordinate destination [3,-2,-1]), Figure 4.7-c (for coordinate destination [-2,-3,1]), and Figure 4.7-d (for coordinate destination [-5,-4,-3]).

The advantages of the Diagonal routing algorithm (Figure 4.8-a) is it chooses the shortest alternative path when avoid the hotspot area or node failures. It makes the distance to destination address closer than adaptive-XYZ (4.8-b) or Elevator-first [16] routing algorithm. The shortest alternative path will avoid degradation on latency.

Algorithm: Diagonal
Input: C(x,y,z) //Coordinate current
D(x,y,z) //Coordinate destination
Output: Selected path
Process:
L(x,y,z) = D(x,y,z) - C(x,y,z)
if L(x,y,z) > 0 then direction_L(x,y,z) = positive //Direction positive
else direction_L(x,y,z) = negative //Direction negative
distance_L(x,y,z) = |L(x,y,z)| //Distance of Lx, Ly, Lz
Procedure:
//main
Choose the farthest distance_L(x,y,z) according to direction_L(x,y,z)
//Alt.1
Choose the second farthest distance_L(x,y,z) according to direction_L(x,y,z)
//Alt.2
Choose the third farthest distance_L(x,y,z) according to direction_L(x,y,z)
//Alt.3
Choose the shortest distance_L(x,y,z) in opposite direction_L(x,y,z)
//Alt.4
Choose the second shortest distance_L(x,y,z) in opposite direction_L(x,y,z)
//Alt.5
Choose the third shortest distance_L(x,y,z) in opposite direction_L(x,y,z)

(a) Diagonal routing algorithm

Algorithm: AdaptiveXYZ
Input: C(x,y,z) //Coordinate current
D(x,y,z) //Coordinate destination
Output: Selected path
Process:
L(x,y,z) = D(x,y,z) - C(x,y,z)
//Direction positive
if L(x,y,z) > 0 then direction_L(x,y,z) = positive
//Direction negative
else direction_L(x,y,z) = negative
Procedure:
//main
Choose X direction according direction_Lx
//Alt.1
Choose Y direction according direction_Ly
//Alt.2
Choose Z direction according direction_Lz

(b) AdaptiveXYZ routing algorithm

Figure 4.8: Comparison between (a) Diagonal routing algorithm and (b) AdaptiveXYZ routing algorithm

The proposed routing algorithm contributes to avoid the degradation performance of 3D mesh topology by tolerating link-fault and hotspot area. We have simulated the proposed algorithm using Noxim to evaluate the performance and compared it with adaptiveXYZ and elevator first routing algorithms.

4.2.2 Evaluation of minimum hops

This subsection presents the evaluation of number of hops for each routing algorithm. We have evaluated three different routing algorithms, namely adaptive-XYZ, Elevator first [16] and Diagonal algorithm in 3x3x3 mesh topology. These algorithms were evaluated under four different scenarios described in Figure 4.9. These scenarios are designed to evaluate the number of hops of each routing algorithm regarding different faulty condition. On the first scenario, we want to evaluate the number of hops for destination node in east-north-up direction of current node with different distances. Second scenario for direction west-north-down, third scenario for west-south-up and fourth scenario for west-north-up.

In scenario-1 as shows in Figure 4.9-a, algorithm adaptive XYZ and Elevator first has a minimum number of hops of 6 nodes while Diagonal needs only 4 nodes. The

Table 4.4: Number of minimum hops comparison of three adaptive routing algorithms

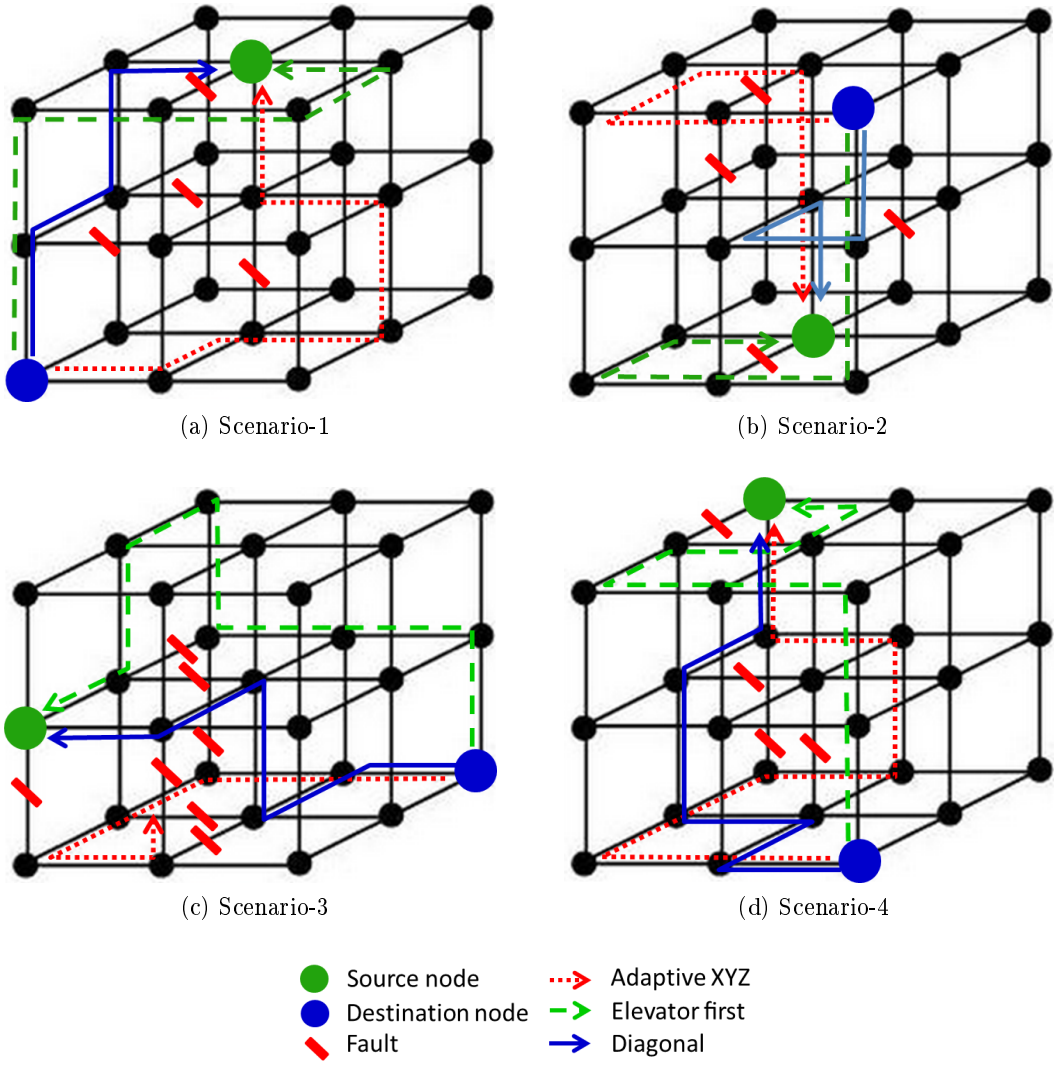
scenario	Source to destination	DyXYZ [17]	Elevator first [16]	Diagonal
1	000 to 112	6	6	4
2	202 to 110	6	6	4
3	220 to 001	deadlock	6	5
4	200 to 022	8	8	6

results also show that, in scenarios 3, adaptive-XYZ routing algorithms cannot avoid the fault that leads to deadlock while Vertical first have 6 hops and Diagonal only 5. Thus, *Diagonal* algorithm has smallest number of hops than adaptive-XYZ and Elevator-first [16] fault-tolerant routing algorithms. Table 4.4 summarizes the evaluation results of number of hops for each routing algorithm. *Diagonal* has lower number of hops in these four scenarios.

For network performances evaluation, we modified the router in Noxim simulator from 2D router to 3D router and support failures as representative of worst-case scenarios. By this method, we were able to evaluate and compare the performance metrics of latency and throughput of the studied algorithms. The methodology, the simulation scenarios and the experimental result of Diagonal routing algorithm are presented in the next chapter.

4.3 Conclusion of the chapter

Gradient and Diagonal are designed as fault-tolerant routing algorithm to avoid hotspots caused by faulty links or node failures in mesh NoCs. Gradient algorithm is designed for 2D mesh topology while Diagonal is for 3D mesh topology. The sequence decision on Gradient algorithm is based on the gradient line, while Diagonal algorithm is based on the combination of distances and directions of destination node from current node. The evaluations of number of hops, alternative path and its comparison with their counterparts are presented. In the next chapter, the evaluation of performance result



(a) Scenario-1 (b) Scenario-2

(c) Scenario-3 (d) Scenario-4

Figure 4.9: Routing path comparison between Diagonal, AdaptiveXYZ and Elevator first [16]

based on simulation are presented and compared. Further, the discussion regarding hardware cost of implementation Gradient routing algorithm into an FPGA is also presented in the next chapter.

Chapter 5

Experimental Results

The concept of Gradient and Diagonal routing algorithm has been presented in the previous chapter. The manual evaluation of number of hops and number of alternative path has also been presented. In this chapter, the methodology and the scenario used in simulation are also defined. The performances based simulation result of Gradient and Diagonal routing algorithm is presented. Hence, the implementation of Gradient routing algorithm into an FPGA is presented.

5.1 Gradient: Fault tolerant routing algorithm

As seen in chapter 2, latency and throughput are the most important network metrics to qualify a NoCs. We design three scenarios to evaluate our proposed fault-tolerant routing algorithms. For the first scenario, we evaluated Gradient performance using different faulty position in the network. The objective is to prove that Gradient has minimum hops in avoiding the faulty. Then, in second scenario, we increase the faulty node to see how big the impact on the performance of each routing algorithm. Finally, we evaluate Gradient algorithm in higher scalability of network by adjust the scalability of network.

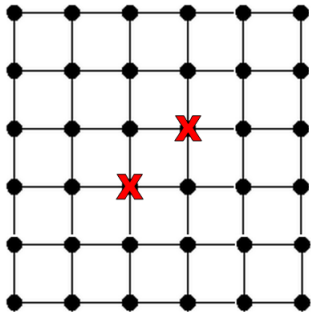
5.1.1 Evaluation based on the faulty position

In the first experiment, we evaluate the latency performance in a faulty network by increasing the packet injection rate. We considered in this evaluation a 6x6 mesh topology with two nodes failures in the center of the network (Figure 5.1a) and four nodes failures near the border of the network (Figure 5.1b). We used at least 6x6 mesh dimension scales in order to obtain a significant different of performance result between our proposed algorithm with its counterpart. The failure on the center of network permit to evaluate the performance of metrics packet sends from center side to border side and vice versa. While the faulty in border network is to evaluate the performance of metric packet send from border side to other border side. These scenarios are chosen as representative of other location failure.

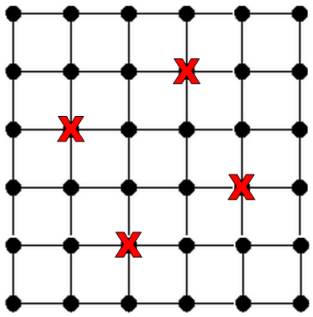
The results in Figure 5.2a shows that in packet injection rate between 5.10^{-4} to 2.10^{-3} almost all routing algorithm have almost the same value of average delay. Then, after $2.10^{-3}$the *Gradient* routing algorithm has lowest average delay.

5.1.2 Evaluation on increasing faulty node in network

We have also evaluated the performance of the different routing algorithms by increasing the worst condition of the network. For this purpose, we increased the percentage of node failures, i.e. the fault percentage of router (0%=normal, 100%=totally faulty). In this scenario we use a 5x5 (Figure 5.3a) and a 6x6 (Figure 5.3b) mesh network, with a packet injection rate of 0.005 packets/cycle/IP and a simulation time of 10.000 cycles. The results in Figure 5.3 show that our algorithm has lowest average delay in all cases and the difference increase as worst case conditions do.

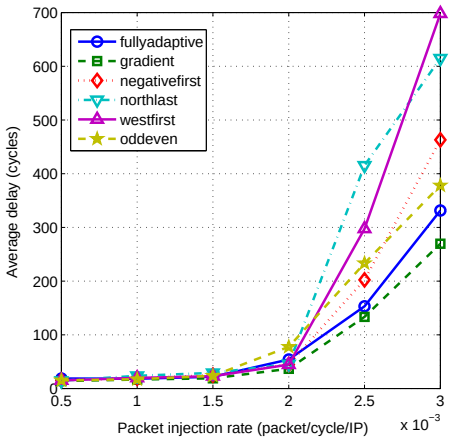


(a) 2-node failures in the centre of network

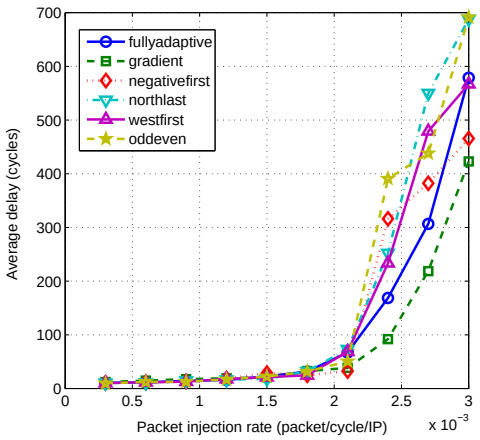


(b) 4-node failures in the near border of network

Figure 5.1: Two scenarios used in simulation: (a) 6x6 mesh with 2-nodes failure in the center of network and (b) 6x6 mesh with 4-nodes failure on near border of network



(a) Average delay result for 2-node failure

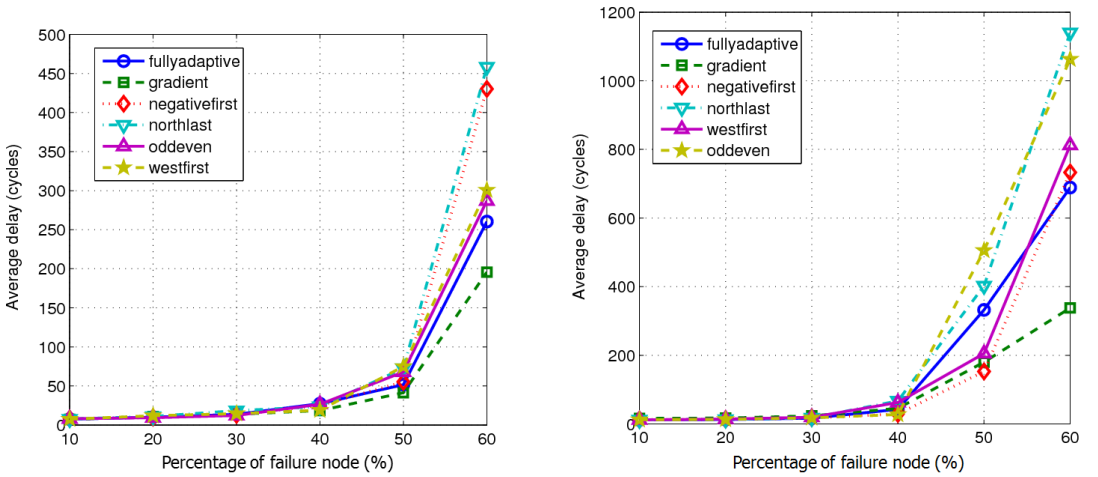


(b) Average delay result for 4-node failure

Figure 5.2: Average delay comparison on increasing packet injection rate for (a) first scenario and (b) second scenario

5.1.3 Evaluation on the scalability of network

In this scenario, we evaluated the scalability of the algorithm by increasing the number of nodes in the network. We increase the number of nodes by increasing the dimension size of the network. For each dimension size, we increase the number of faulty



(a) Average delay result in 5x5 mesh on increasing failure percentage

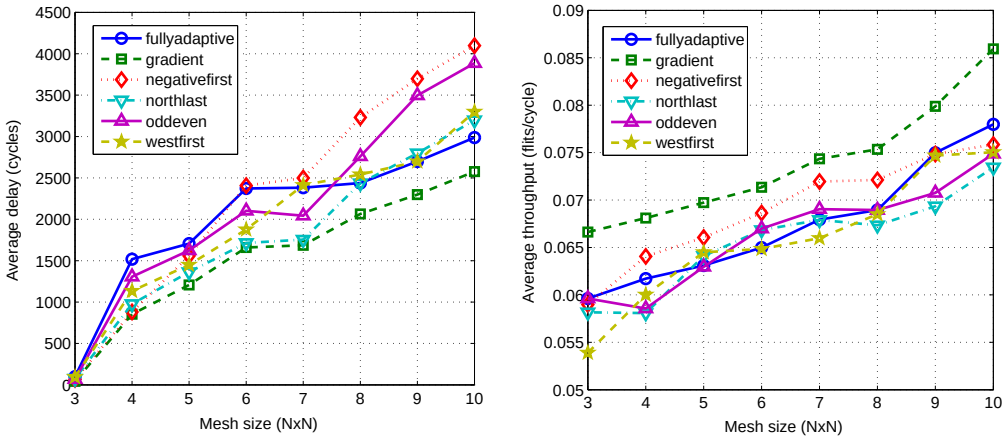
(b) Average delay result in 6x6 mesh on increasing faulty percentage

Figure 5.3: Average delay comparison on increasing faulty percentage with packet injection rate of 0.005 packets/cycle/IP for (a) 5x5 mesh and (b) 6x6 mesh topology

nodes, randomly dispatched in the network. We used a packet injection rate of 0.01 packets/cycle/IP and increased the mesh dimensions from 3x3 to 10x10. The number of faulty node in the network is respectively ranging from one to eight. The simulation result shows that the proposed routing algorithm has the lowest average delay and highest average throughput on all sizes of network as shown in Figure 5.4-a and 5.4-b.

5.1.4 Conclusion

Three scenarios have been presented to evaluate the performance of Gradient fault tolerant routing algorithm. The result shows that Gradient algorithm has lower delay and higher throughput than its counterparts. The result of lower latency in chapter 5 proof the result in chapter 4, that Gradient has minimum number of hops. While the result that Gradient has higher throughput proof that it have more alternative path. Thus, it is validated that Gradient has better performances than its counterpart and contributes to avoid the degradation of NoC performances in hot-spot and worst traffic condition.



(a) Average delay result in on increasing network size from 2x2 to 10x10 mesh

(b) Average throughput result in on increasing network size from 2x2 to 10x10 mesh

Figure 5.4: (a) Average delay and (b) average throughput on increasing number of nodes from 3x3 to 10x10 with packet injection rate 0,01 packets/cycle/IP

Table 5.1: Parameter use in simulation to evaluate Diagonal routing algorithm

Parameter	Value
Traffic pattern	Random
Buffer size	4 flits
Packet size	2 - 10 flits
Simulation time	10.000 cycles
Warm-up time	1.000 cycles

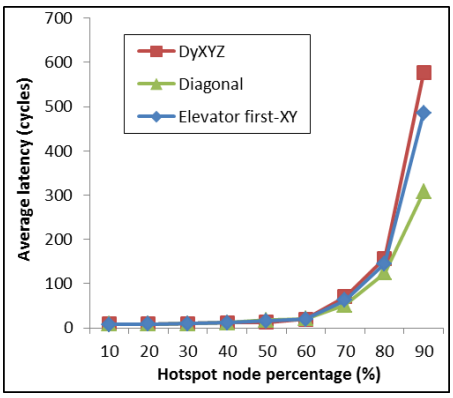
5.2 Diagonal: Fault tolerant routing algorithm

The number of hops and the number of alternative path of Diagonal routing algorithm have been evaluated in chapter 4. It shows that Diagonal has less number of hops and more alternative path than its counterpart. In this section, we evaluate the latency and throughput performance of Diagonal. For this purpose, we modified Noxim to support 3D mesh NoC and failures nodes. Then, we define network parameters value used in simulation as shown in Table 5.1.

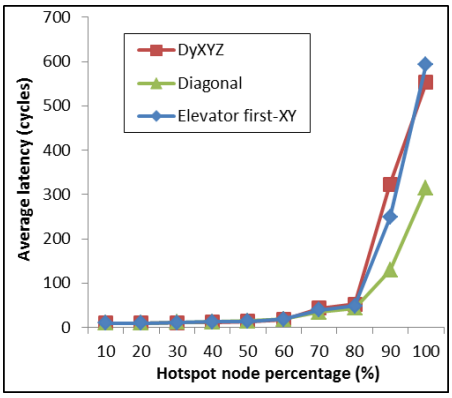
5.2.1 Evaluation on increasing faulty in network

As the first experiment, we evaluate the performance of Diagonal in a faulty network by increasing the packet injection rate. We consider the dimension of the network 3x3x3 with a packet injection rate of 0,004 (packets/cycle) and two faulty nodes in the 3D mesh NoC. We then increase the percentage of the faulty nodes and evaluate the average latency of packet traversal. The result shows that the proposed routing algorithm have lower average delay than DyXYZ [17] and routing algorithm as shown on Fig. 5.5-a.

In a second experiment, we used a 4x4x4 dimension size with a packet injection rate of 0,0015 (packets/cycle) and 3 faulty nodes in the network. Figure 5.5-b shows that proposed algorithm also have lower average delay as in previous scenario.



(a) Average delay comparison for 3x3x3 mesh with PIR 0,004

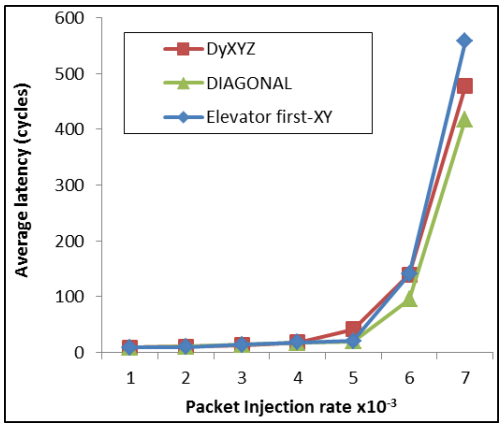


(b) Average delay comparison for 4x4x4 mesh with PIR 0,0015

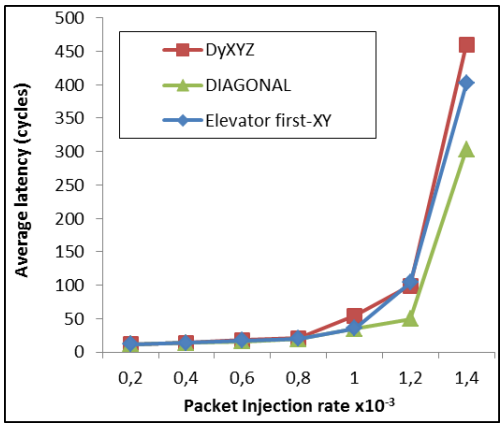
Figure 5.5: Average delay comparison on (a) 4x4x4 mesh with packet injection rate 0,004 and (b) 4x4x4 mesh with PIR 0,0015

5.2.2 Evaluation on increasing injection rate in network

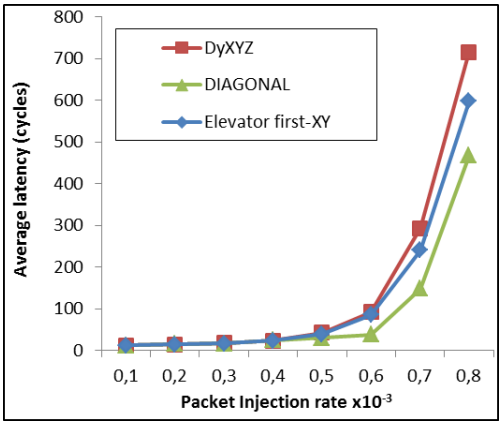
Further we modify the behavior of the network by increasing the packet injection rate on different mesh dimension size. In this simulation we defined different number of faulty node for each dimension size. We use two faulty nodes for 3x3x3x mesh, three faulty nodes for 4x4x4 mesh, four faulty nodes for 5x5x5 mesh and five faulty nodes for 6x6x6 mesh. The result on Figure 5.6 shows that the proposed algorithm has lower delay than adaptiveXYZ in all mesh dimension size.



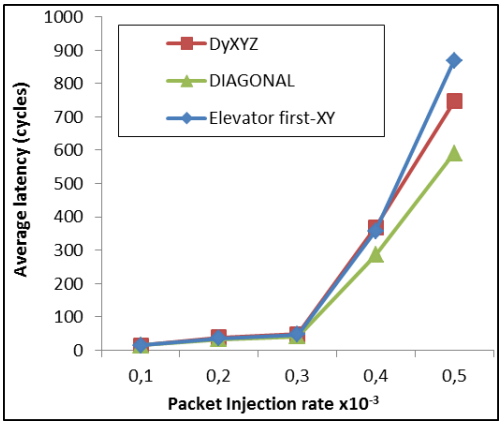
(a) Average delay comparison for 3x3x3 mesh



(b) Average delay comparison for 4x4x4 mesh



(c) Average delay comparison for 5x5x5 mesh



(d) Average delay comparison for 6x6x6 mesh

Figure 5.6: Average delay comparison on increasing packet injection rate from 0,0005 to 0,0035 packets/cycle/IP for (a) 3x3x3, (b) 4x4x4, (c) 5x5x5, (d) 6x6x6 mesh

5.2.3 Conclusion

We have evaluated the performances of Diagonal routing algorithm in different conditions. The result shows that Diagonal has lower average latency due to the fact that it has a minimum number of hops. Thus, Diagonal algorithm contributes to avoid the degradation of NoC performances in hot-spot and worst traffic condition.

5.3 Implementation on FPGA

The implementation of NoC depends on the complexity of routing algorithms, and affects the amount of hardware for the router and/or network interface. In this section we present an evaluation of the implementation of Gradient routing algorithm into a virtex-5 (xc5vsx50t-1ff665) FPGA. Further, we evaluate the impact of frequency to the performance of Gradient. The synthesis result were obtained using ISE 12.4 [77] tools while the metric performances are evaluated using Modelsim 6.4 [26].

5.3.1 Synthesis result

We implemented Gradient routing algorithm as a new switch control in HERMES NoC router [1] as shown in Figure 5.7. This HERMES-NoC router has been successfully prototyped onto Virtex-II xilinx FPGA [47]. The basic elements of this router are switch control logic and five bi-directional ports, connecting to four other switches and to a local IP core. The switch control logic consist of arbitration and routing algorithm. This router provides two flow control schemes (credit-based, handshake) and two scheduling schemes (round-robin, priority). The router parameters used for the evaluation are show in Table 5.2.

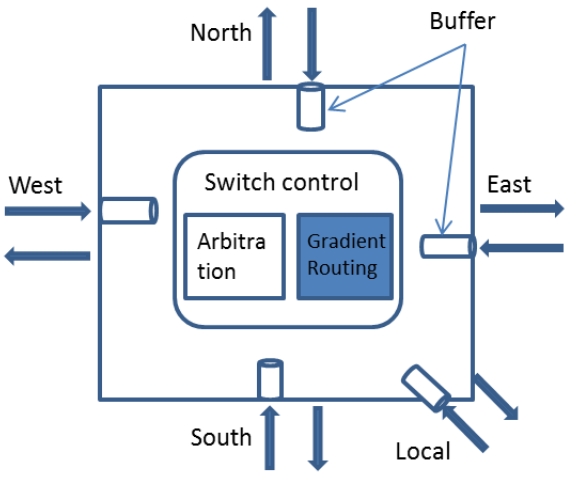


Figure 5.7: Implementing Gradient routing logic inside switch control block of HERMES NoC router

We used two metrics in order to evaluate the resources usage and the metric performances: latency and throughput. For the first experiment, we define the type of flow control, number of virtual channel and scheduling schemes to evaluate how big is the impact of the different routing algorithms schemes: XY, turn model and Gradient to the utilization of resources in targeted FPGA. We then define four scenarios as representative of different combination schemes as shows in Table 5.3. To obtain the synthesis result, we then implement each scenario into Virtex-5 FPGA.

Table 5.2: Router parameter

Architecture	Type/value
Dimension size	4x4 (16 Routers)
Flow control	Handshake, CreditBased
Scheduling	Round Robin, Priority
Virtual channel	0, 1, 2
Flit width	32 bits
Buffer depth	8 bits
Frequency	50 MHz
Routing type	XY, WestFirst, NorthLast, NegativeFirst, Gradient

The synthesis results of scenario-1 are shown in Table 5.4, scenario-2 in Table 5.5, scenario-3 in Table 5.6 and scenario-4 in Table 5.7. The synthesis results shows that XY routing algorithm use less resources of slice LUTs for scenario-1, scenario-3 and scenario-4 and also use less slice registers for scenario-4. However XY use less resource in FPGA than its counterpart but XY algorithm is not an adaptive routing. In general, all routing algorithm use almost the same resources in the targeted FPGA. Thus, Gradient

algorithm is better due to it has best network performance compared to its counterpart.

Table 5.3: Evaluation scenarios

Scenario	Flow Control	Virtual channel	Scheduling
1	Handshake	-	-
2	Credit Based	1	-
3	Credit Based	2	Round Robin
4	Credit Based	2	Priority

Table 5.4: FPGA resources needed for implementing handshake flow control without virtual channel nor scheduling

Routing	Slice Register	Slice LUTs	LUT Flip-Flop pairs	BUFG/BUFGCTRLs
XY	11%	52%	21%	6%
Turn model	11%	53%	21%	6%
Gradient	11%	53%	21%	6%

Table 5.5: FPGA resources need for implementing Credit Based flow control with 1 virtual channel and no scheduling

Routing	Slice Register	Slice LUTs	LUT Flip-Flop pairs	BUFG/BUFGCTRLs
XY	11%	38%	27%	5%
Turn model	11%	39%	27%	5%
Gradient	11%	39%	27%	5%

Table 5.6: FPGA resources need for implementing credit based flow control with 2 virtual channel and round robin scheduling

Routing	Slice Register	Slice LUTs	LUT Flip-Flop pairs	BUFG/BUFGCTRLs
XY	25%	89%	25%	5%
Turn model	25%	89%	25%	5%
Gradient	25%	89%	25%	5%

Table 5.7: FPGA resources need for implementing Credit Based flow control with 2 virtual channel and priority scheduling

Routing	Slice Register	Slice LUTs	LUT Flip-Flop pairs	BUFG/BUFGCTRLs
XY	13%	47%	25%	5%
Turn model	25%	93%	25%	5%
Gradient	25%	93%	25%	5%

5.3.2 Gradient performances in RTL level

As stated in chapter 2, designer can design a NoC system in high level abstraction, middle level or RTL level. In chapter 3, the evaluation of Gradient performance has been done in middle level. In this sub-section, we evaluate the performance of Gradient routing algorithm at the RTL level.

The implementation of Gradient algorithm as switch control in Hermes router is evaluated using Modelsim 6.4 [26]. The goal is to evaluate the metric performances and to shows that using Modelsim tool, Gradient algorithm also have better performances than its counterpart. The simulation results are then compared to existing algorithm such as west-first, negative-first and north-last. In this simulation result, we did not compared the performance of Gradient with other adaptive 2D mesh routing algorithm

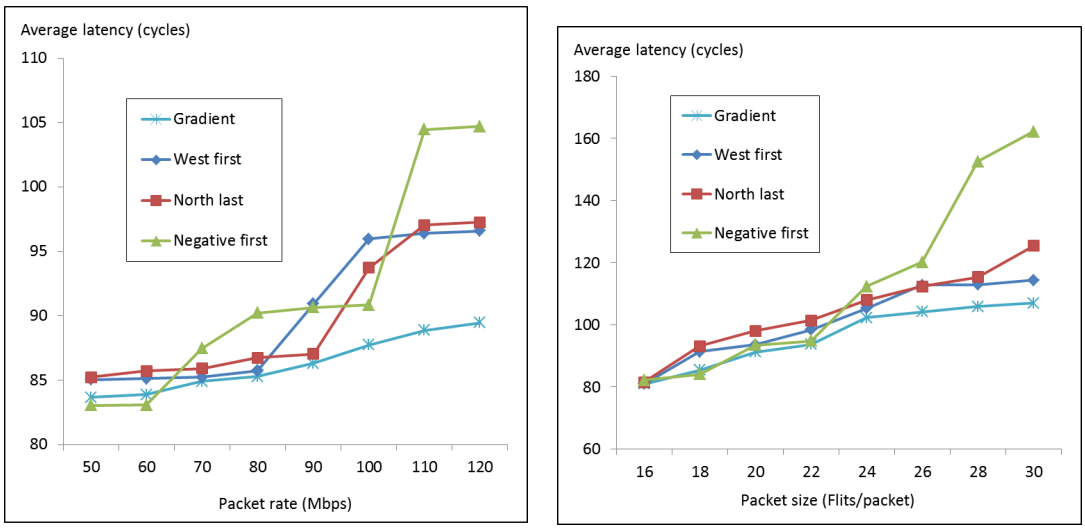
(i.e. odd-even and fully-adaptive routing algorithm) since we implemented Gradient to Hermes NoC [1] router. Hermes NoC router only provides XY, west-first, negative-first and north-last routing algorithm but not on other adaptive routing algorithm.

In this scenario, the network parameters such as frequency, target node, numbers of packet, packet size, traffic distribution, packet rate and simulation time are defined as shown in Table 5.8. Further, we adjust the packet rate and the packet size value (Table 5.8) to evaluate the output latency and throughput.

Table 5.8: Scenarios for simulation of network performances

Parameters	Traffic Scenario
Frequency	50 MHz
Target Node	Random
Number of Packet	1000 packets
Packet Size	10, 12, 14, 16, 18, 20, 22, 24 flits
Traffic Distribution	Uniform
Packet Rate	60, 80, 100, 120, 140, 160 Mbps
Simulation time	10.000 ns

The simulation results for the increasing packet rate scenario are presented in Figure 5.8-a while for the increasing packet size scenario are presented in Figure 5.8-b. The evaluation of latency in increasing packet rate (Figure 5.8-a) and packet size (Figure 5.8-b) shows that Gradient routing algorithm has lower latency than other routing algorithms.



(a) Average latency comparison on increasing packet rate

(b) Average latency comparison on increasing packet size

Figure 5.8: Average latency comparison of 2D mesh routing algorithm on increasing packet rate and packet size

5.4 Conclusion of the chapter

Gradient and Diagonal routing algorithms are designed to distribute traffic evenly among the paths to avoid failure and minimize contention. The scenarios and the methodology to evaluate the performances of Gradient and Diagonal routing algorithm are presented. The experimental results show that both algorithms has lower latency and higher throughput than it counterpart routing algorithm.

The new switch control that implement Gradient routing algorithm has been implemented in Hermes router then implemented into FPGA virtex-5. The synthesis result shows that Gradient use the same resources of slice LUTs as turn model routing algorithm. But, in the evaluation metric performances, Gradient has lower latency. However the XY algorithm has less resource but this algorithm is not adaptive. Thus, our proposed routing algorithm is the best adaptive routing due to its best performances and it use the same resources than other adaptive routing algorithms.

Chapter 6

Conclusions and perspectives

6.1 Conclusions

SoCs or the embedded system is widely used in consumer electronics such as cell phone, digital video camera, Global Positioning System (GPS) or portable game devices. Current applications for embedded systems are designed with high number of communicating tasks. To provide these requirements, embedded system will be designed with hundreds or thousands of processing element core or processor inside a chip as predicted by Moore's Law. This requires highly parallel and flexible communication architecture between each processing element in an embedded system.

The shared bus communication and direct point-to-point interconnections have been used for the communication between processing elements on a chip, however this communications cannot provide the intercommunication requirement in current and future MPSoCs. When the number of the element cores increase, bus arbitration bottleneck can increase the delay, while using direct point-to-point can become messy in wiring.

Network-on-Chips (NoCs) communication architecture has been proposed as a promising replacement to overcome many limitations of buses and point-to-point communication infrastructure by adopting the communication architecture of general network computer. NoC provides adaptivity and flexibility infrastructure to overcome performances degradation due to change in environment and QoS requirements of applications.

The ideal design of NoC is to get high throughput, minimum latency, minimum using resources, low power consumption and small area size. The quality of NoC can be measured from its network performance (i.e throughput, latency and reliability) and area size. To reach the maximum performance, the design parameters of NoC such as hardware architecture (i.e router micro-architecture, link architecture and topology), and software architecture (i.e application, security, traffic, transport protocol, packet size, header size, routing, flow control, switching, flit size, and buffer size) must be considered.

All NoC parameters have influence to NoC quality performances, thus the first objective of this PhD was to evaluate the impact of NoC parameters on its performance as a consideration in designing an adaptive NoC. The combination of NoC parameters values has been used to define network condition that represents minimum QoS of network. Then, adjusting the values of each parameter is used to see how big the impact of upgrading value on the performances.

The Noxim systemC based simulator has been used to evaluate the impact of NoC designed parameter to the performance since it can model hardware characteristics. The results show that the accuracy of choosing and adjusting the network parameters can avoid performance degradation. These results can be used to design control mechanism in an adaptive NoC to avoid the degradation of QoS when traffic condition change.

To improve the network performances of NoC, routing algorithm is designed to distribute traffic evenly among the paths to avoid hotspots and minimize contention. The major problems of deterministic routing algorithm arise when faulty link or failures node occurs in the networks. Designing a fault-tolerant routing algorithm to avoid link fault and load balancing is mandatory to achieve design of reliable NoCs.

The second objective of this PhD was to propose a fault-tolerant routing algorithm to avoid faults in the network. Gradient and Diagonal routing algorithm has been proposed as fault-tolerant routing algorithm to overcome the deadlock packet. Gradient is proposed for 2D mesh, while Diagonal is designed for 3D mesh topology.

In 2D mesh, compared to existing routing algorithms that divide 2D coordinates

into four destination zones, Gradient divide 2D coordinates into eight zones based on a gradient line. Gradient sequences choose the shortest alternative route to avoid a hot-spot area or a faulty element in the network. Thus, it avoids the degradation of network performances such as latency and throughput. In term of performances, the experimental results show that Gradient algorithm has lower latency and higher throughput than its counterpart.

The sequences decision chosen in Diagonal routing is based on the combination of distances and directions of destination node from current node in 3D NoC. It has a main route and four alternative paths. For the main route until second alternative path, the algorithm choose the farthest distance in the same direction, while for the third until five alternative, it choose the shortest distance in opposite direction. The experimental results show that Diagonal algorithm have smallest number of hops, more alternative possibilities paths to tolerate faults, lower latency and higher throughput than its counterpart.

The accuracy in selecting the alternative route can avoid increasing the number of hops of packets. Therefore, Gradient and Diagonal routing algorithm has minimum hops compared to existing fault-tolerant routing algorithm for 2D and 3D mesh NoC. Proposed algorithms contributes to avoid the degradation of NoC performances in hot-spot and worst traffic conditions.

Finally, the implementation of Gradient routing algorithm into an FPGA has been presented. A new Gradient switch control has been designed and implemented in HERMES router. The synthesis results shows that however Gradient routing algorithm have better performance than its counterpart, it use the same resources amount of FPGA with other turn model adaptive routing algorithm such as west-first, negative-first and north-last.

6.2 Perspectives

In chapter 3, the evaluation of the impact of NoC design parameters on the performance as a consideration in designing an adaptive NoC has been presented. In the near future,

the researchers can continue this work by making a simple software application as a tool to give an overview of the impact of NoC design parameter on performance.

The concept and the performance of Gradient and Diagonal fault tolerant routing algorithm have been presented. The implementation of Gradient routing algorithm onto FPGA has been done, while Diagonal not. Thus, in the next future, Diagonal routing algorithm should be implemented into FPGA.

The performance evaluation and the implementation onto FPGA of Gradient routing algorithm have been presented in chapter 5. In the next future, we propose to continue this work by designing a dedicated prototype of Gradient router in a 4x4 mesh NoC.

Increasing NoC application that require different QoS, make NoC designer consider the network management such as fault detection, fault management, monitoring system or network control architecture. Integrating fault detection module in every NoC router can helps Gradient and Diagonal routing to choose the optimal routing path to destination. It can be local fault detection or global network fault detection. This module then can be used as fault control network that receive and send information or signal to all component in the system. Thus, all components in the network can react and adapt fastly with the current condition of network. Further, a fault management protocol is needed to handle growing number of fault information in network. Fault management protocol will be responsible for detecting, identifying faults in the network then taking corrective actions.

Adding monitoring system module is also part of future work. Thus, all information such as current performance, behavior of each traffic application can be monitored then sent to all components in the network. With this monitoring, the information in NoC system can be viewed at transaction level. Thus, the fault detection and error identification fault process can be more quick and easy.

Finally, an intelligent NoC router is proposed as future work to react and adapt the information from fault management and network monitoring module. The intelligent NoC router must be automatically reconfigurable and user manageable.

Bibliography

[1] Atlas noc simulator documentation https://corfu.pucrs.br/redmine/projects/atlas.

[2] The network simulator (ns-2) source and documentation. http://www.isi.edu/nsnam/ns/.

[3] Nirgam simulator sorces and user guide http://nirgam.ecs.soton.ac.uk/.

[4] Omnet++ network simulator documentation http://www.omnetpp.org.

[5] D. Atienza, F. Angiolini, S. Murali, A. Pullini, L Benini, and G.D. Micheli. Network-on-chip design and synthesis outlook. *Integration, the VLSI Journal*, 41(3):340 – 359, 2008.

[6] N. Bagherzadeh and M. Matsuura. Performance impact of task-to-task communication protocol in network-on-chip. In *Fifth International Conference on Information Technology: New Generations (ITNG), 2008*, pages 1101 –1106, 2008.

[7] T. Bjerregaard and S. Mahadevan. A survey of research and practices of network-on-chip. *ACM Computing Surveys (CSUR)*, 38:1–51, 2006.

[8] T. Bjerregaard and S. Mahadevan. A survey of research and practices of network-on-chip. *ACM Comput. Surv.*, 38(1), June 2006.

[9] R.V. Boppana and S. Chalasani. Fault-tolerant wormhole routing algorithms for mesh networks. *IEEE Transactions on Computers*, 44(7):848 –864, jul 1995.

[10] F. Chaix, D. Avresky, N. Zergainoh, and M. Nicolaidis. Fault-tolerant deadlock-free adaptive routing for any set of link and node failures in multi-cores systems. In *9th IEEE International Symposium on Network Computing and Applications (NCA), 2010*, pages 52 –59, july 2010.

[11] F. Chaix, D. Avresky, N.-E. Zergainoh, and M. Nicolaidis. A fault-tolerant deadlock-free adaptive routing for on chip interconnects. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, pages 1 –4, march 2011.

[12] G.M. Chiu. The odd-even turn model for adaptive routing. *IEEE Transactions on Parallel and Distributed Systems*, 11(7):729 –738, jul 2000.

[13] W.J. Dally and C.L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, C-36(5):547 –553, may 1987.

[14] W.J. Dally and B. Towles. Route packets, not wires: on-chip interconnection networks. In *Design Automation Conference (DAC), 2001*, pages 684 – 689, 2001.

[15] M. Daneshtalab and M. Ebrahimi. A systematic reordering mechanism for on-chip networks using efficient congestion-aware method. *Journal of Systems Architecture*, 2012.

[16] F. Dubois, A. Sheibanyrad, F. Pe?trot, and M. Bahmani. Elevator-first: A deadlock-free distributed routing algorithm for vertically partially connected 3d-nocs. *Computers, IEEE Transactions on*, 62(3):609–615, 2013.

[17] Masoumeh Ebrahimi, Xin Chang, Masoud Daneshtalab, Juha Plosila, Pasi Liljeberg, and Hannu Tenhunen. Dyxyz: Fully adaptive routing algorithm for 3d nocs. In *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*, pages 499–503, 2013.

[18] M. Elhajji, B. Attia, A. Zitouni, R. Tourki, S. Meftali, and J.-L. Dekeyser. Feronoc: Flexible and extensible router implementation for diagonal mesh topology. In *Design and Architectures for Signal and Image Processing (DASIP), 2011 Conference on*, pages 1 –8, nov. 2011.

[19] F. Fazzino, M. Palesi, and D. Patti. Noxim: Network-on-chip simulator. http://noxim.sourceforge.net.

[20] B.S. Feero and P.P. Pande. Networks-on-chip in a three-dimensional environment: A performance evaluation. *Computers, IEEE Transactions on*, 58(1):32 –45, jan. 2009.

[21] D. Fick, A. DeOrio, G. Chen, V. Bertacco, D. Sylvester, and D. Blaauw. A highly resilient routing algorithm for fault-tolerant nocs. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2009.*, pages 21 –26, april 2009.

[22] S. Foroutan, Y. Thonnart, R. Hersemeule, and A. Jerraya. An analytical method for evaluating network-on-chip performance. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, pages 1629 –1632, march 2010.

[23] Fayez Gebali, Haytham Elmiligi, and Mohamed Watheq El-Kharashi. *Networks-on-Chips: Theory and Practice*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2009.

[24] P. Gehlot and S.S. Chouhan. Performance evaluation of network on chip architectures. In *International Conference on Emerging Trends in Electronic and Photonic Devices Systems*, pages 124 –127, 2009.

[25] C.J. Glass and L.M. Ni. The turn model for adaptive routing. In *The 19th Annual International Symposium on Computer Architecture, 1992*, pages 278 –287, 1992.

[26] Mentor graphics. *ModelSimő LE/PE UserŠs Manual 6.4*, 2009.

[27] F. Guo, Y Solihin, L. Zhao, and R. Iyer. A framework for providing quality of service in chip multi-processors. In *40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2007*, pages 343 –355, 2007.

[28] H. Hossain, M. Ahmed, A. Al-Nayeem, T.Z. Islam, and M. Akbar. Gpnocsim - a general purpose simulator for network-on-chip. In *International Conference on Information and Communication Technology (ICICT), 2007.*, pages 254 –257, march 2007.

[29] J. Hu and R. Marculescu. Dyad - smart routing for networks-on-chip. In *Design Automation Conference, 2004.*, pages 260 –263, july 2004.

[30] S. Jamuna and V.K. Agrawal. Implementation of bistcontroller for fault detection in clb of fpga. In *Devices, Circuits and Systems (ICDCS), 2012 International Conference on*, pages 99 –104, march 2012.

[31] Axel Jantsch and Hannu Tenhunen, editors. *Networks on chip*. Kluwer Academic Publishers, Hingham, MA, USA, 2003.

[32] G. Becker D. Towles B. Jiang, N. Michelogiannakis and W.J. Dally. Booksim 2.0 user guide.

[33] A.B. Kahng, Bin Li, Li-Shiuan Peh, and K. Samadi. Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration. In *Design,*

Automation Test in Europe Conference Exhibition (DATE), 2009, pages 423 –428, april 2009.

[34] M.A. Khan and A.Q. Ansari. Quadrant-based xyz dimension order routing algorithm for 3-d asymmetric torus routing chip (atrc). In *International Conference on Emerging Trends in Networks and Computer Communications (ETNCC), 2011*, pages 121 –124, april 2011.

[35] H.S. Kia and C. Ababei. A new fault-tolerant and congestion-aware adaptive routing algorithm for regular networks-on-chip. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 2465 –2472, june 2011.

[36] R. Koch, T. Pionteck, C. Albrecht, and E. Maehle. An adaptive system-on-chip for network applications. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, page 8, 2006.

[37] Tim Kogel, Rainer Leupers, and Heinrich Meyr. *Integrated system-level modeling of network-on-chip enabled multi-processor platforms.* Kluwer, 2006.

[38] T. Kranich and M. Berekovic. Noc switch with credit based guaranteed service support qualified for gals systems. In *Digital System Design (DSD), 2010.*, pages 53–59, 2010.

[39] S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani. A network on chip architecture and design methodology. In *IEEE Computer Society Annual Symposium on VLSI, 2002*, pages 105 –112, 2002.

[40] M. Li, Q.A. Zeng, and W.B. Jone. Dyxy - a proximity congestion-aware deadlock-free dynamic routing method for network on chip. In *Design Automation Conference, 2006 43rd ACM/IEEE*, pages 849 –852, 0-0 2006.

[41] Ye Lu, J. McCanny, and S. Sezer. The impact of global routing on the performance of nocs in fpgas. In *Reconfigurable Computing and FPGAs (ReConFig), 2011 International Conference on*, pages 369–374, 2011.

[42] Zhonghai Lu, Rikard Thid, Mikael Millberg, and Axel Jantsch. Nnse: Nostrum network-on-chip simulation environment. In *In Proc. of SSoCC*, 2005.

[43] Peter S. Magnusson, Magnus Christensson, Jesper Eskilson, Daniel Forsgren, Gustav Hållberg, Johan Högberg, Fredrik Larsson, Andreas Moestedt, and Bengt Werner. Simics: A full system simulation platform. *Computer*, 35(2):50–58, February 2002.

[44] T. Mak, K.P. Lam, P. Cheung, and W. Luk. Adaptive routing in network-on-chips using a dynamic programming network. *IEEE Transactions on Industrial Electronics*, PP(99):1, 2010.

[45] Milo M. K. Martin, Daniel J. Sorin, Bradford M. Beckmann, Michael R. Marty, Min Xu, Alaa R. Alameldeen, Kevin E. Moore, Mark D. Hill, and David A. Wood. Multifacet's general execution-driven multiprocessor simulator (gems) toolset. *SIGARCH Comput. Archit. News*, 33(4):92–99, November 2005.

[46] H. Matsutani, M. Koibuchi, and H. Amano. Tightly-coupled multi-layer topologies for 3-d nocs. In *International Conference on Parallel Processing (ICPP), 2007*, page 75, sept. 2007.

[47] F. Moraes, N. Calazans, A. Mello, L. Moller, and L. Ost. Hermes: an infrastructure for low area overhead packet-switching networks on chip. *Integration, the VLSI Journal*, 38:69–93, 2004.

[48] A. Narasimhan, O. Kumaravelu, and R. Sridhar. An investigation of the impact of network parameters on performance of network-on-chips. In *48th Midwest Symposium on Circuits and Systems, 2005*, pages 1617 –1620 Vol. 2, 2005.

[49] V.-D. Ngo and H.-W. Choi. Analyzing the performance of mesh and fat-tree topologies for network on chip design. *Computeur Science*, 3824:300–310, 2005.

[50] Umit Y. Ogras, Paul Bogdan, and Radu Marculescu. An analytical approach for network-on-chip performance analysis. *Trans. Comp.-Aided Des. Integ. Cir. Sys.*, 29:2001–2013, December 2010.

[51] M. Palesi, D. Patti, and F. Fazzino. *Noxim - the NoC Simulator - User Guide.* University of Catania, 2010.

[52] H. Park and D.P. Agrawal. Generic methodologies for deadlock-free routing. In *Parallel Processing Symposium, 1996., Proceedings of IPPS '96, The 10th International*, pages 638–643, 1996.

[53] S. Pasricha and Y. Zou. A low overhead fault tolerant routing scheme for 3d networks-on-chip. In *12th International Symposium on Quality Electronic Design (ISQED), 2011*, pages 1 –8, march 2011.

[54] M. Pastrnak, P.H.N. de With, and J. van Meerbergen. Realization of qos management using negotiation algorithms for multiprocessor noc. In *2006 IEEE International Symposium on Circuits and Systems (ISCAS), 2006.*, page 4, 2006.

[55] V.F. Pavlidis and E.G. Friedman. 3-d topologies for networks-on-chip. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 15(10):1081 –1090, oct. 2007.

[56] L.S Peh and N.E. Jerger. *On-Chip Networks*. Morgan and Claypool Publishers, 1st edition, 2009.

[57] S.G. Pestana, E. Rijpkema, A. Radulescu, K. Goossens, and O.P. Gangwal. Cost-performance trade-offs in networks on chip: a simulation-based approach. In *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, volume 2, pages 764 – 769 Vol.2, feb. 2004.

[58] V. Puthal, G.M.K. Singh and V. Laxmi. C-routing: An adaptive hierarchical noc routing methodology. In *2011 IEEE/IFIP 19th International Conference on VLSI and System-on-chip*, october 2011.

[59] R.S. Ramanujam and B. Lin. Near-optimal oblivious routing on three-dimensional mesh networks. In *IEEE International Conference on Computer Design, 2008. ICCD 2008*, pages 134 –141, oct. 2008.

[60] J. Riihimaki, E. Salminen, K. Kuusilinna, and T. Hamalainen. Parameter optimization tool for enhancing on-chip network performance. In *IEEE International Symposium on Circuits and Systems, 2002. ISCAS 2002*, volume 4, pages IV–61 – IV–64 vol.4, 2002.

[61] A.V.V. Rose, R. Seshasayanan, and G. Oviya. Fpga implementation of low latency routing algorithm for 3d network on chip. In *2011 International Conference on Recent Trends in Information Technology (ICRTIT)*, pages 385 –388, june 2011.

[62] C. Rusu, L. Anghel, and D. Avresky. Rilm: Reconfigurable inter-layer routing mechanism for 3d multi-layer networks-on-chip. In *On-Line Testing Symposium (IOLTS), 2010 IEEE 16th International*, pages 121–126, 2010.

[63] Claudia Rusu, Lorena Anghel, and Dimiter Avresky. Adaptive inter-layer message routing in 3d networks-on-chip. *Microprocessors and Microsystems*, 35(7):613 – 631, 2011.

[64] A. Sheibanyrad, F. Petrot, and A. Jantsch. *3D Integration for NoC-based SoC Architectures*. Springer Verlag, 2010.

[65] Abbas Sheibanyrad, Frdric Ptrot, and Axel Jantsch. *3D Integration for NoC-based SoC Architectures*. Springer Publishing Company, Incorporated, 1st edition, 2010.

[66] R. Sunkam Ramanujam and B. Lin. A layer-multiplexed 3d on-chip network architecture. *Embedded Systems Letters, IEEE*, 1(2):50 –55, aug. 2009.

[67] A. W. Topol, D. C. La Tulipe, L. Shi, D. J. Frank, K. Bernstein, S. E. Steen, A. Kumar, G. U. Singco, A. M. Young, K. W. Guarini, and M. Ieong. Three-dimensional integrated circuits. *IBM Journal of Research and Development*, 50(4.5):491 –506, july 2006.

[68] Brian Towles and William J. Dally. Worst-case traffic for oblivious routing functions. In *Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures*, SPAA '02, pages 1–8, New York, NY, USA, 2002. ACM.

[69] M. Valinataj and S. Mohammadi. A fault-aware, reconfigurable and adaptive routing algorithm for noc applications. In *VLSI System on Chip Conference (VLSI-SoC), 2010 18th IEEE/IFIP*, pages 13 –18, sept. 2010.

[70] Freek Verbeek and Julien Schmaltz. On necessary and sufficient conditions for deadlock-free routing in wormhole networks. *IEEE Transactions on Parallel and Distributed Systems*, 22(12):2022–2032, 2011.

[71] J.C. Villanueva, J. Flich, J. Duato, H. Eberle, N. Gura, and W. Olesinski. A performance evaluation of 2d-mesh, ring, and crossbar interconnects for chip multiprocessors. In *2nd International Workshop on Network on Chip Architectures, 2009. NoCArc 2009*, pages 51 –56, dec. 2009.

[72] Wang. *System-on-Chip Test Architectures: Nanometer Design for Testability*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008.

[73] Y.Q. Wang, Zhang M.X., and Xiao C.W. A fault tolerant adaptive routing algorithm in 2d mesh network on chip. In *IEEE 3rd International Conference on Communication Software and Networks (ICCSN), 2011*, pages 140 –144, may 2011.

[74] J. Wu. A fault-tolerant adaptive and minimal routing approach in 3-d meshes. In *Seventh International Conference on Parallel and Distributed Systems, 2000*, pages 256 –263, 2000.

[75] J. Wu. A fault-tolerant and deadlock-free routing protocol in 2d meshes based on odd-even turn model. *IEEE Transactions on Computers*, 52(9):1154 – 1169, sept. 2003.

[76] J. Wu. A simple fault-tolerant adaptive and minimal routing approach in 3-d meshes. *J. Comput. Sci. Technol.*, 18:1–13, January 2003.

[77] Xilinx. *Synthesis and Simulation Design Guide - ISE 12.4*, 2008.

[78] Z. Zhang, A. Greiner, and S. Taktak. A reconfigurable routing algorithm for a fault-tolerant 2d-mesh network-on-chip. In *Proceedings of the 45th annual Design Automation Conference*, DAC '08, pages 441–446, New York, NY, USA, 2008. ACM.

[79] Aamir Zia, Sachhidh Kannan, H. Jonathan Chao, and Garrett S. Rose. 3d {NOC} for many-core processors. *Microelectronics Journal*, 42(12):1380 – 1390, 2011.

Personal Publication

1. **I. Pratomo** and S. Pillement. Impact of design parameters on performance of adaptive network-on-chips. In High Performance Computing and Simulation (HPCS), 2012 International Conference on, pages 724 –725, july 2012.

2. **I. Pratomo** and S. Pillement. Gradient - An Adaptive Fault-tolerant Routing Algorithm for 2D Mesh Network-on-Chips. In Design & Architectures for Signal & Image Processing (DASIP), International Conference, October 2012.

Abbreviations

ASIC	Application-Specific Integrated Circuit
BE	Best Effort
CN	Computer Networks
DBFC	Dimensional Bubble Flow Control
DOR	Dimension Order Routing
DSP	Digital Signal Processor
FPGA	Field Programmable Gate Array
GLB	Global Congestion Information
GpNoCsim	General Purpose Simulator for Network-on-Chip
GPS	Global Positioning System
GS	Guaranteed Service
IC	Integrated Circuit
IO	Input Output
IP	Intellectual Property
ISE	Integrated Software Environment

LUT	Look Up Table
MPSoC	Dynamic Multi-Processor System-on-Chip
NI	Network Interface
NoC	Network-on-Chip
NoP	Neighbors-on-Path
Noxim	Network on Chip Simulator
NS	Network Simulator
PE	Processing element
QoS	Quality of Services
RF	Radio Frequency
RILM	Reconfigurable Inter-layer Routing Mechanism
RNI	Resource-Network-Interfaces
RTL	Register Transfer Level
SAF	Store-and-Forward
SEU	Single Event Upsets

SoC	System-on-Chip
TCP	Transport Control Protocol
TSV	Through Silicon vias
TTL	Time To Live
UDP	Unit Data Protocol
ULSI	Ultra Large Scale Integration
VC	Virtual Chanel
VHDL	VHSIC Hardware Description Language
VHSIC	Very-High-Speed Integrated Circuits
VLSI	Very Large Scale Integration

Appendix A

Path line of routing algorithm in 20 case scenario

Figures in Appendix A shows the decision path of XY (Figure A.1), west first (Figure A.2), North-last (Figure A.3), Negative first (Figure A.4), Full-adaptive (Figure A.5), Fault tolerant (Figure A.6) and Gradient (Figure A.7) routing algorithm for the destination in west (1), east (2), north (3), south (4), north-west (5, 6, 13, 14), north-east (7, 8, 15, 16), south-west (9, 10, 17, 18) and south-east (11, 12, 19, 20) from current node.

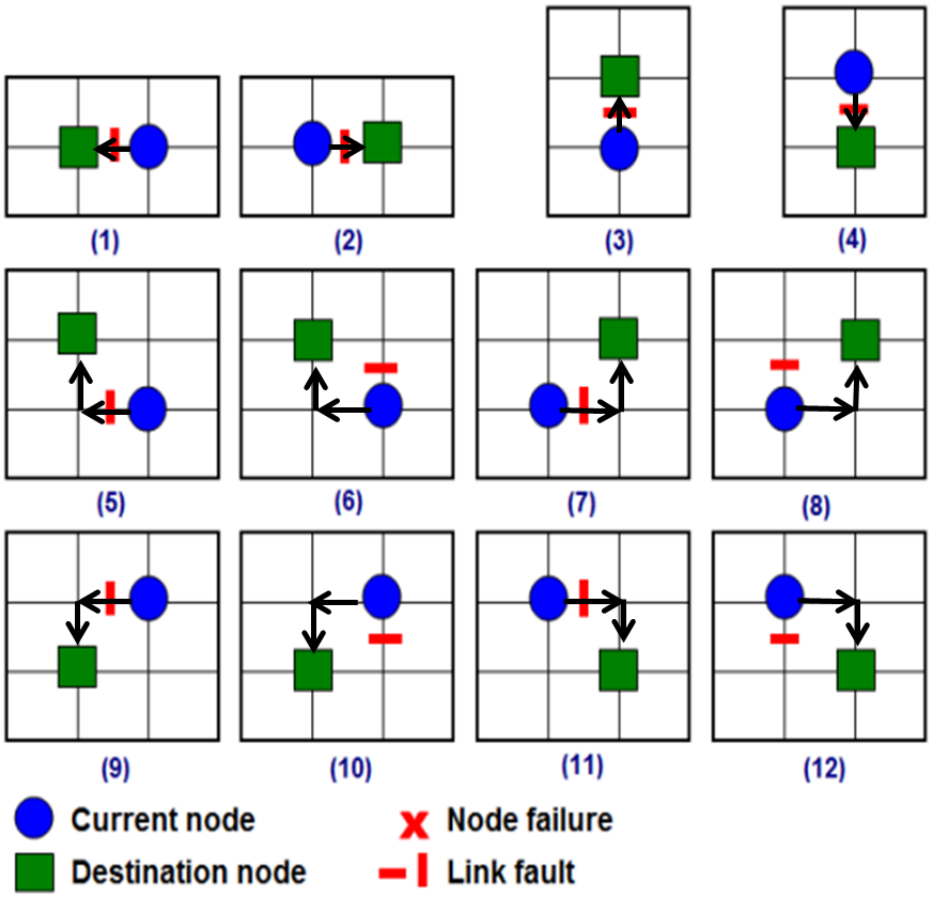


Figure A.1: Decision path of XY routing algorithm in one fault condition between current and destination node

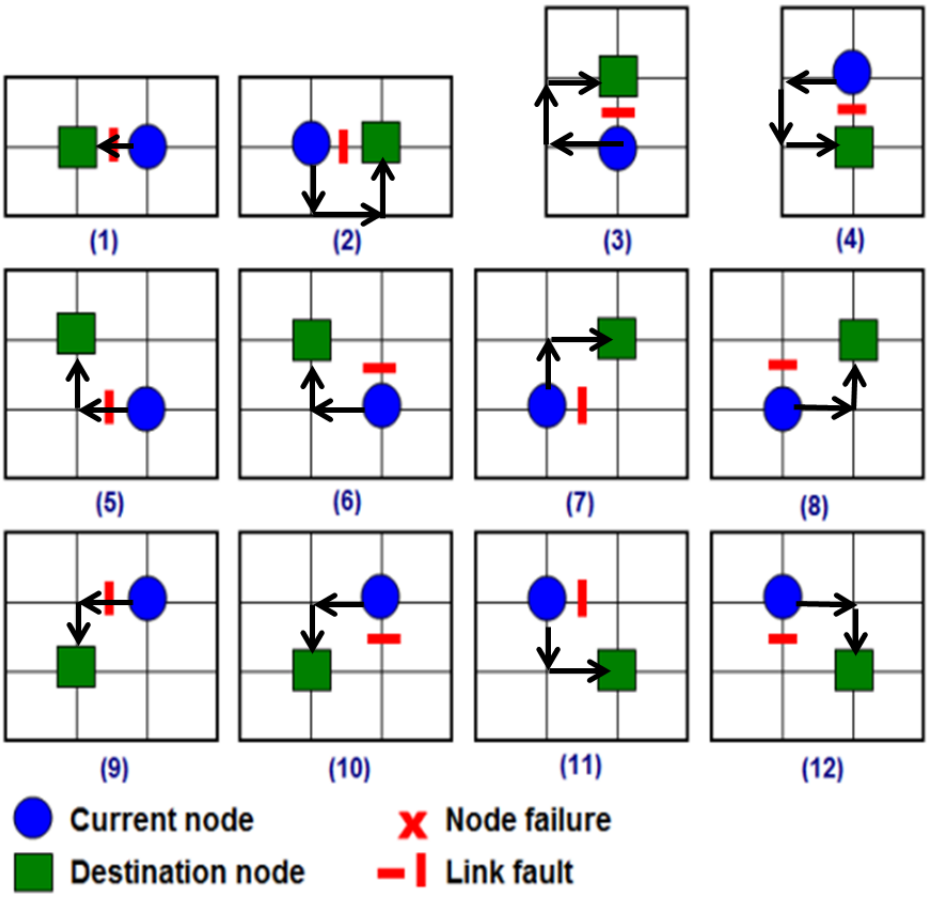


Figure A.2: Decision path of West-first routing algorithm in one fault condition between current and destination node

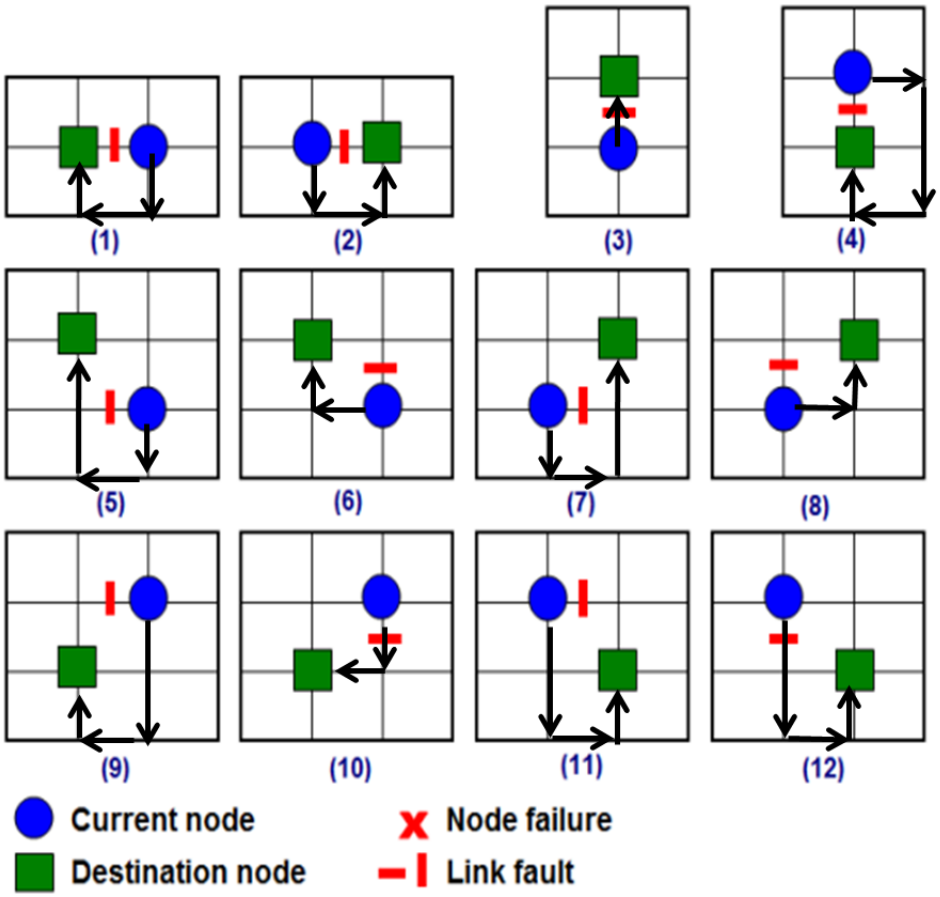


Figure A.3: Decision path of North-last routing algorithm in one fault condition between current and destination node

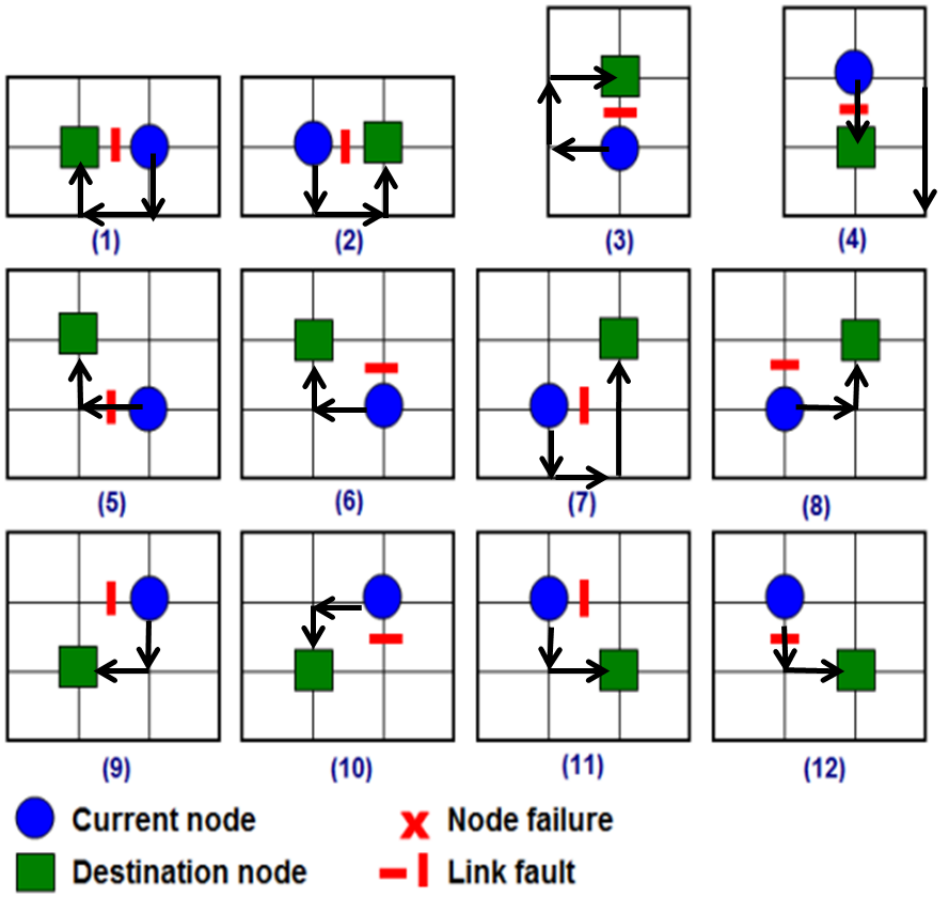


Figure A.4: Decision path of Negative-first routing algorithm in one fault condition between current and destination node

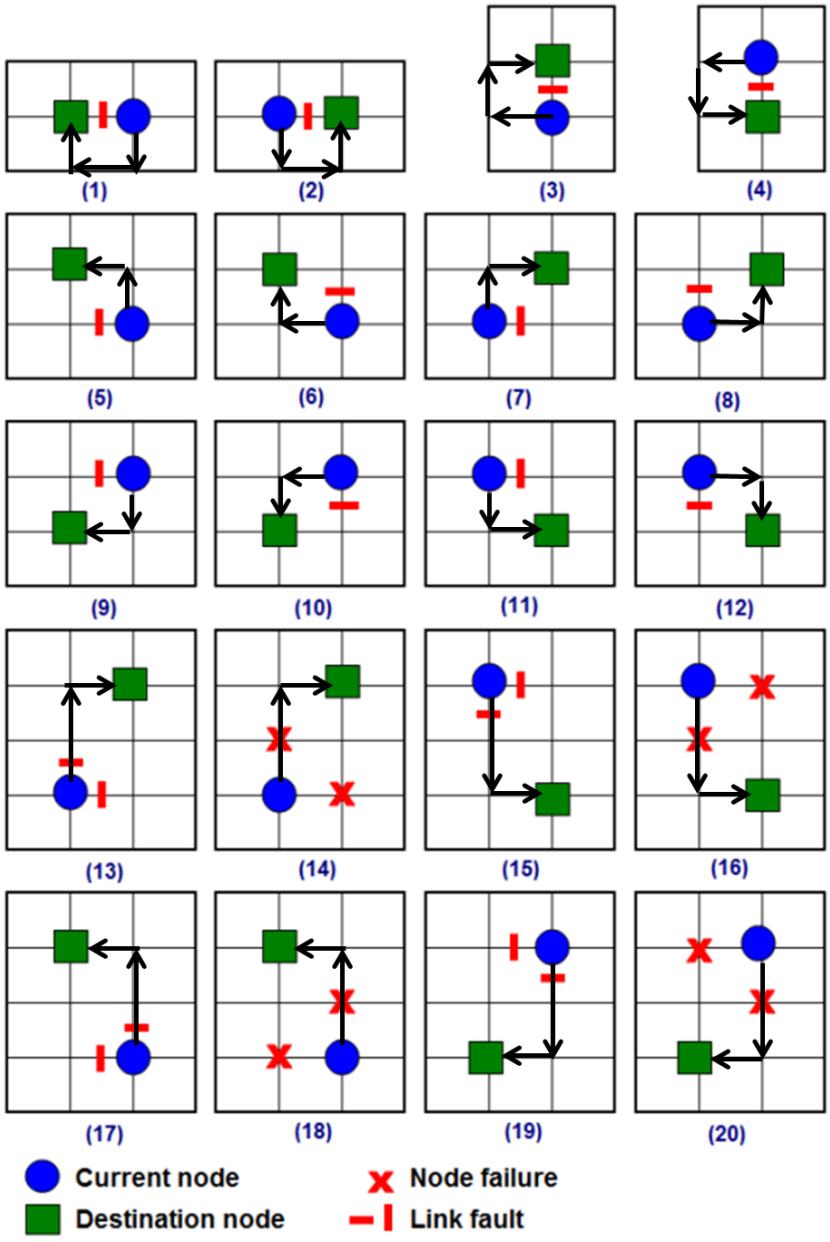


Figure A.5: Decision path of Fully-adaptive routing algorithm in 20 condition between current and destination node

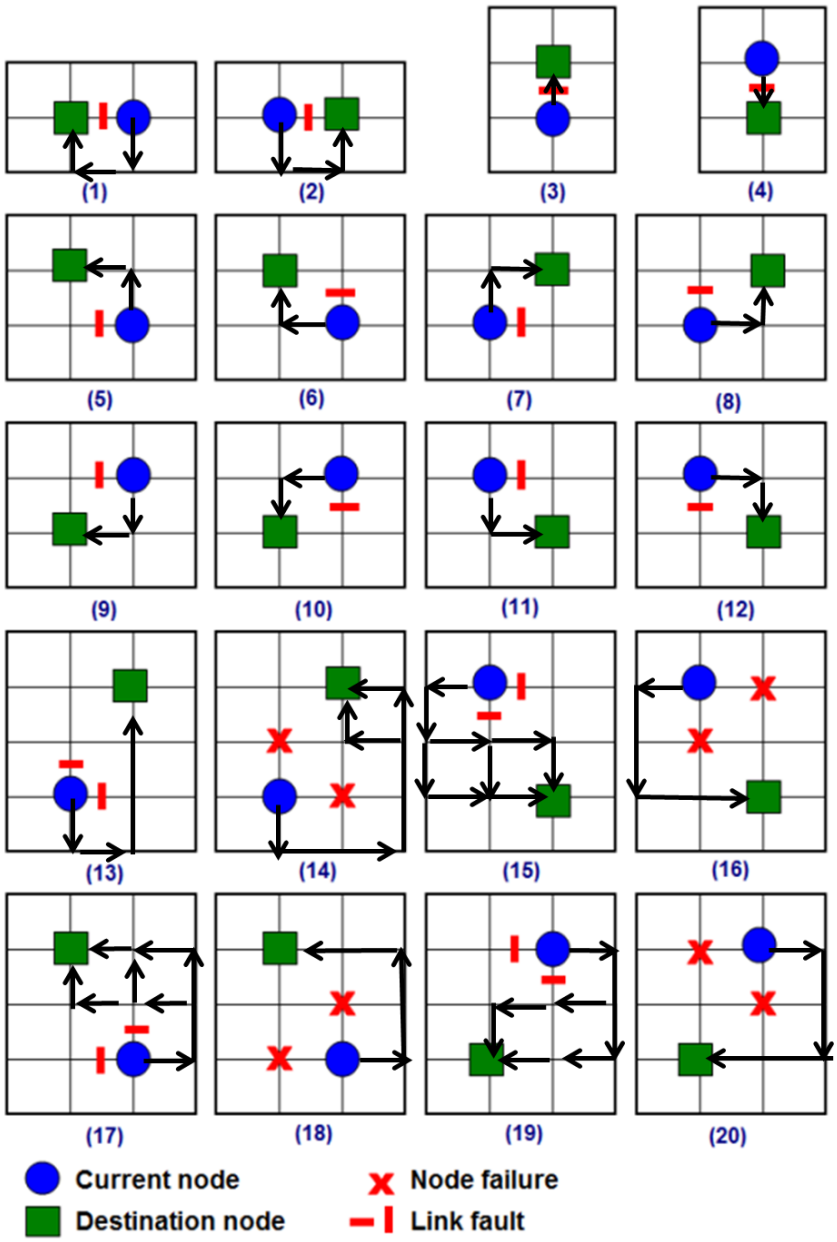


Figure A.6: Alternative path of Fault-tolerant routing algorithm [11] in minimum number of hops

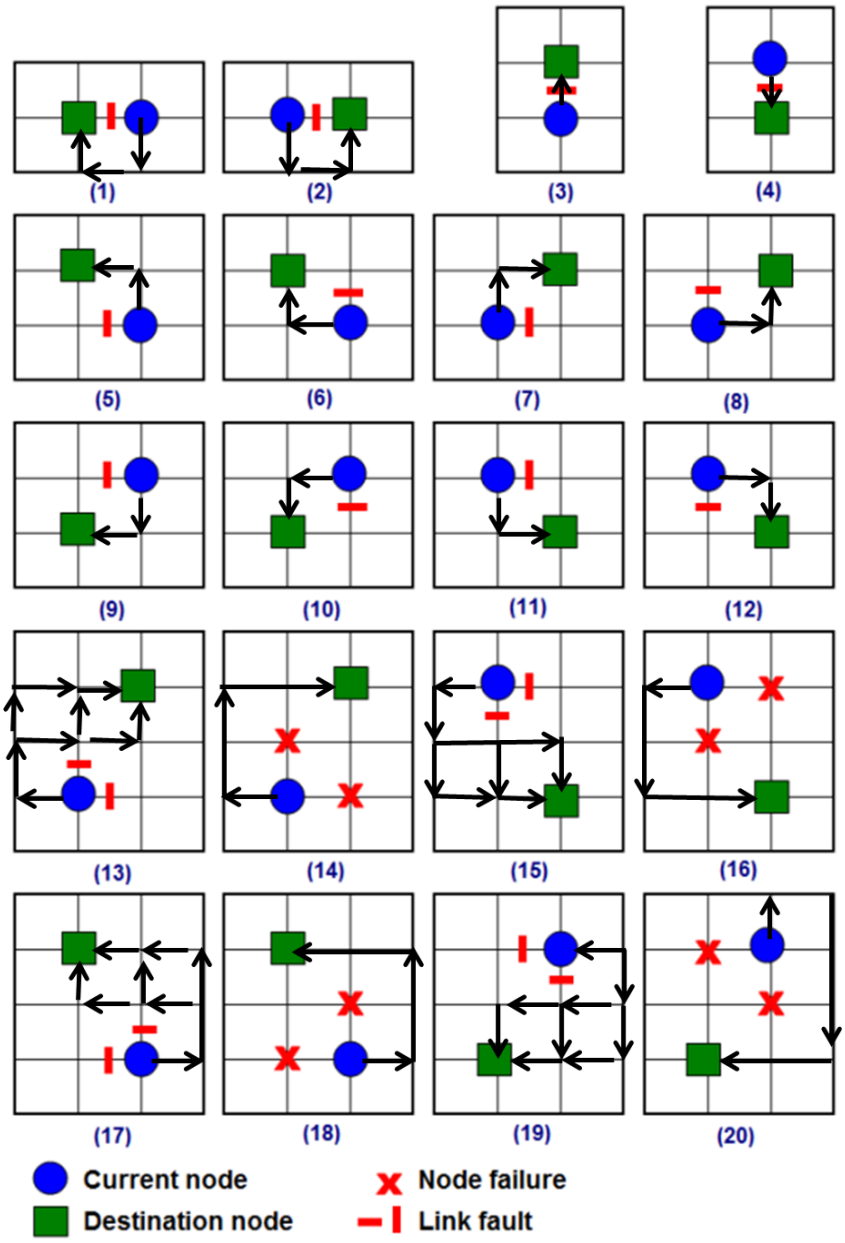


Figure A.7: Alternative path of Gradient routing algorithm in minimum number of hops

Appendix B

Tabel combination of Diagonal algorithm

This table below presented the destination zone and the sequence decision of Diagonal routing algorithm. The classification of the destination zone is based on the distance and the direction of destination address from current address. As an example, if the destination address is (3,2,4) and the current address is (1,1,1) then the ($\triangle$X,$\triangle$Y,$\triangle$Z) is (2,1,3). Thus $\triangle X = +, \triangle Y = +, \triangle Z = +$ and $|\triangle Z| \geq |\triangle X| \geq |\triangle Y|$. So, the destination can be classify in zone $|\triangle Z|+ \geq |\triangle X|+ \geq |\triangle Y|+$ as shown in row No.5. The sequence decision for this destination zone is Z+ for the main route, X+ as first alternative, Y+ as second alternative, Y- as third alternative, X- as fourth alternative and Z- as fifth alternative.

No	Decision based	Decision					
	distances and directions	main	al.1	al.2	al.3	al.4	al.5
1	$\|\triangle X\|+ \geq \|\triangle Y\|+ \geq \|\triangle Z\|+$	X+	Y+	Z+	Z-	Y-	X-
2	$\|\triangle X\|+ \geq \|\triangle Z\|+ \geq \|\triangle Y\|+$	X+	Z+	Y+	Y-	Z-	X-
3	$\|\triangle Y\|+ \geq \|\triangle X\|+ \geq \|\triangle Z\|+$	Y+	X+	Z+	Z-	X-	Y-
4	$\|\triangle Y\|+ \geq \|\triangle Z\|+ \geq \|\triangle X\|+$	Y+	Z+	X+	X-	Z-	Y-
5	$\|\triangle Z\|+ \geq \|\triangle X\|+ \geq \|\triangle Y\|+$	Z+	X+	Y+	Y-	X-	Z-
6	$\|\triangle Z\|+ \geq \|\triangle Y\|+ \geq \|\triangle X\|+$	Z+	Y+	X+	X-	Y-	Z-

No	Decision based	Decision					
	distances and directions	main	al.1	al.2	al.3	al.4	al.5
7	|△X|+≥|△Y|+≥|△Z|-	X+	Y+	Z-	Z+	Y-	X-
8	|△X|+≥|△Z|-≥|△Y|+	X+	Z-	Y+	Y-	Z+	X-
9	|△Y|+≥|△X|+≥|△Z|-	Y+	X+	Z-	Z+	X-	Y-
10	|△Y|+≥|△Z|-≥|△X|+	Y+	Z-	X+	X-	Z+	Y-
11	|△Z|-≥|△X|+≥|△Y|+	Z-	X+	Y+	Y	X	Z
12	|△Z|-≥|△Y|+≥|△X|+	Z+	Y+	X+	X-	Y-	Z-
13	|△X|+≥|△Y|-≥|△Z|+	X+	Y-	Z+	Z-	Y+	X-
14	|△X|+≥|△Z|+≥|△Y|-	X+	Z+	Y-	Y+	Z-	X-
15	|△Y|-≥|△X|+≥|△Z|+	Y-	X+	Z+	Z-	X-	Y+
16	|△Y|-≥|△Z|+≥|△X|+	Y-	Z+	X+	X-	Z-	Y+
17	|△Z|+≥|△X|+≥|△Y|-	Z+	X+	Y-	Y+	X-	Z-
18	|△Z|+≥|△Y|-≥|△X|+	Z+	Y-	X+	X-	Y+	Z-
19	|△X|+≥|△Y|-≥|△Z|-	X+	Y-	Z-	Z+	Y+	X-
20	|△X|+≥|△Z|-≥|△Y|-	X+	Z-	Y-	Y+	Z+	X-
21	|△Y|-≥|△X|+≥|△Z|-	Y-	X+	Z-	Z+	X-	Y+
22	|△Y|-≥|△Z|-≥|△X|+	Y-	Z-	X+	X-	Z+	Y+
23	|△Z|-≥|△X|+≥|△Y|-	Z-	X+	Y-	Y+	X-	Z+
24	|△Z|-≥|△Y|-≥|△X|+	Z-	Y-	X+	X-	Y+	Z+
25	|△X|-≥|△Y|+≥|△Z|+	X-	Y+	Z+	Z-	Y-	X+
26	|△X|-≥|△Z|+≥|△Y|+	X-	Z+	Y+	Y-	Z-	X+
27	|△Y|+ ≥|△X|-≥|△Z|+	Y+	X-	Z+	Z-	X+	Y-

No	Decision based	Decision					
	distances and directions	main	al.1	al.2	al.3	al.4	al.5
28	|△Y|+≥|△Z|+≥|△X|-	Y+	Z+	X-	X+	Z-	Y-
29	|△Z|+≥|△X|-≥|△Y|+	Z+	X-	Y+	Y-	X+	Z-
30	|△Z|+≥|△Y|+≥|△X|-	Z+	Y+	X-	X+	Y-	Z-
31	|△X|-≥|△Y|+≥|△Z|-	X-	Y+	Z-	Z+	Y-	X+
32	|△X|-≥|△Z|+≥|△Y|+	X-	Z-	Y+	Y-	Z+	X+
33	|△Y|+ ≥|△X|-≥|△Z|-	Y+	X-	Z-	Z+	X+	Y-
34	|△Y|+≥|△Z|-≥|△X|-	Y+	Z-	X-	X+	Z+	Y-
35	|△Z|-≥|△X|-≥|△Y|+	Z-	X-	Y+	Y-	X+	Z+
36	|△Z|-≥|△Y|+≥|△X|-	Z-	Y+	X-	X+	Y-	Z+
37	|△X|-≥|△Y|-≥|△Z|+	X-	Y-	Z+	Z-	Y+	X+
38	|△X|-≥|△Z|+≥|△Y|-	X-	Z+	Y-	Y+	Z-	X+
39	|△Y|- ≥|△X|-≥|△Z|+	Y-	X-	Z+	Z-	X+	Y+
40	|△Y|-≥|△Z|+≥|△X|-	Y-	Z+	X-	X+	Z-	Y+
41	|△Z|+≥|△X|-≥|△Y|-	Z+	X-	Y-	Y+	X+	Z-
42	|△Z|+≥|△Y|-≥|△X|-	Z+	Y-	X-	X+	Y+	Z
43	|△X|-≥|△Y|-≥|△Z|-	X-	Y-	Z-	Z+	Y+	X+
44	|△X|-≥|△Z|-≥|△Y|-	X-	Z-	Y-	Y+	Z+	X+
45	|△Y|- ≥|△X|-≥|△Z|-	Y-	X-	Z-	Z+	X+	Y+
46	|△Y|-≥|△Z|-≥|△X|-	Y-	Z-	X-	X+	Z+	Y+
47	|△Z|-≥|△X|-≥|△Y|-	Z-	X-	Y-	Y+	X+	Z+
48	|△Z|-≥|△Y|-≥|△X|-	Z-	Y-	X-	X+	Y+	Z+

Appendix C

Decision path comparison of 3D mesh topology for 4 fault scenarios

Figures in Appendix B shows the comparison of decision path of AdaptiveXYZ, Elevator first and Diagonal routing algorithm for fault scenario-1 (B.1), fault scenario-2 (B.2), fault scenario-3 (B.3) and fault scenario-4 (B.4)

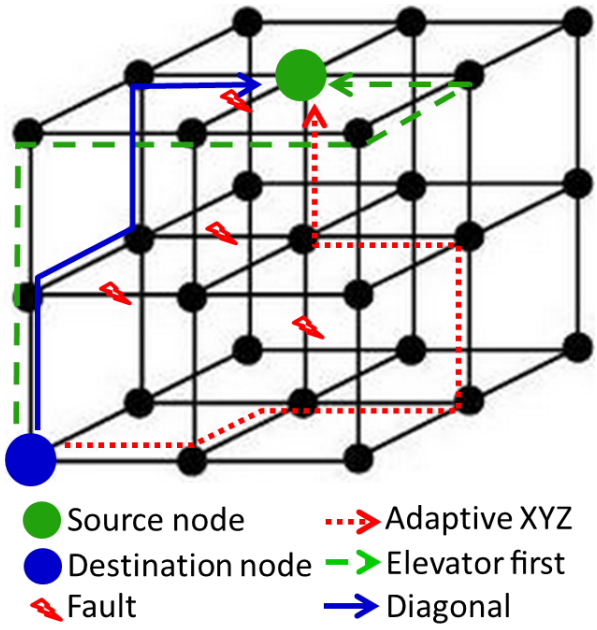


Figure C.1: The comparison of decision path of AdaptiveXYZ, Elevator first and Diagonal routing algorithm for fault scenario-1

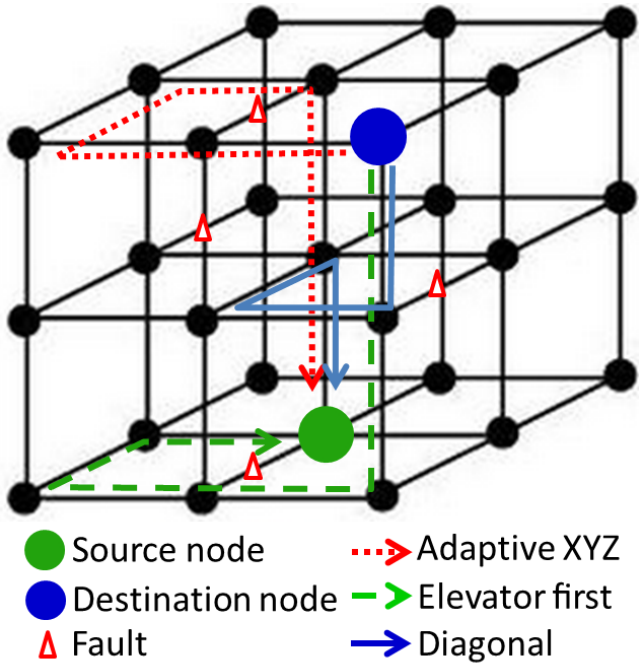


Figure C.2: The comparison of decision path of AdaptiveXYZ, Elevator first and Diagonal routing algorithm for fault scenario-2

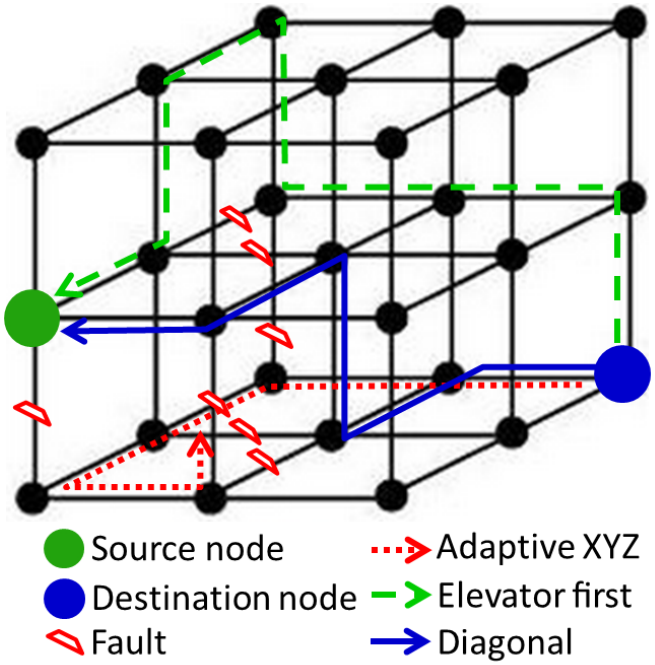


Figure C.3: The comparison of decision path of AdaptiveXYZ, Elevator first and Diagonal routing algorithm for fault scenario-3

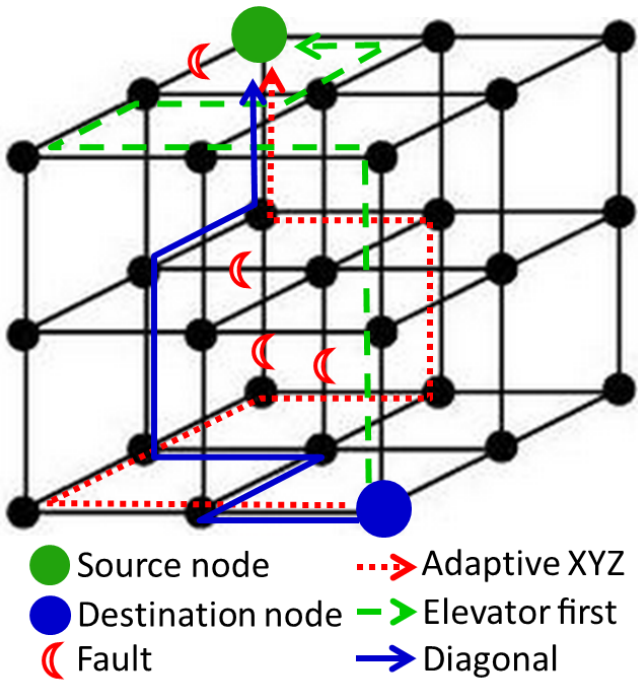


Figure C.4: The comparison of decision path of AdaptiveXYZ, Elevator first and Diagonal routing algorithm for fault scenario-4

NoC Adaptatif pour SoC Reconfigurable

22 octobre 2013

1 Introduction

Les nouveaux systèmes embarqués intègre des milliards de transistors et des composants hétérogènes intégrés ensemble dans une puce appelée système multiprocesseur sur puce (MPSoC). Les bus et les interconnexions point-à-point ne permettent plus de supporter les communications entre toutes ces ressources. Pour soutenir efficacement ces communications, il est alors nécessaire d'implémenter un un réseau d'interconnexion très parallèle régulier et flexible.

Les réseaux sur puce (NoC-pour Network-on-Chip) ont été proposés comme une solution de communication dans les MPSoC. L'architecture NoC fournit l'infrastructure de communication qui est adaptable et offre différente qualité des services (QoS) basée sur les besoins des applications. En outre, les NoC peuvent aussi fournir une infrastructure souple de communications [1].

Une architecture à base de NoC est constituée de ressources de traitement tels que des processeurs, des DSP ou des unités de stockage et des commutateurs, ou routeurs, qui sont connectés par des canaux de sorte qu'ils sont en mesure de communiquer les uns avec les autres en s'envoyant des messages. En outre les routeurs peuvent stocker localement les messages en transit dans des buffers.

La topologie maillée (mesh) est la plus utilisée pour les NoC en raison de sa simplicité et de sa grande évolutivité [2]. Dans un circuit 2D générique les ressources sont connectés dans une grille formant une maille homogène, en 3D le maillage consiste à empiler des mailles 2D. Les architectures 3D offrent de meilleures performances réseau par rapport aux architectures 2D [3] en raison de la complexité croissante des circuits et de la longueur des interconnexions dans les topologies 2D.

L'introduction de la qualité de service dans un réseau sur puce nécessite des mécanismes de réservation afin de garantir le temps de latence et le débit des paquets proposés. Les interconnexions NoC ont beaucoup de paramètres qui influent sur leurs performances. Dans ce sens, un concepteur de NoC doit choisir plusieurs paramètres (et leurs valeurs) qui ont chacun un impact sur les performances du réseau. La précision de ces choix et l'ajustement de la valeur de ces paramètres peuvent éviter la dégradation des performances du réseau. Dans cette thèse nous avons d'abord proposer une évaluation de l'impact des paramètres de conception d'un NoC sur son rendement. Nous avons ensuite fait varier leurs valeurs afin d'évaluer leur impact sur la variation des performances. L'objectif est d'évaluer quelle est

l'incidence de la mise à niveau d'une valeur d'un paramètre donné sur les performances. Les résultats obtenus peuvent être considérés comme une base pour la conception de mécanisme de contrôle d'un NoC adaptatif pour éviter la dégradation de la QoS dans des conditions d'opérations changeantes.

L'utilisation des technologies submicroniques profondes dans un système embarqué cause une augmentation de la susceptibilité au "Single Event Upset" (SEU) qui peuvent diminuer la fiabilité des circuits. Un SEU se produit lorsque un rayonnement provoque le changement de valeur d'un bit dans certaines bascules du circuit. Cette modification indésirable peut causer le dysfonctionnement de l'architecture. Un autre problème qui mène à l'apparition de défauts permanents dans le circuit est le " vieillissement ". Dans le cas d'un défaut permanent l'élément fautif doit être éliminé de la communication. Une autre solution est de re-router le paquet en évitant le composant défectueux, ainsi un routage tolérant aux pannes est nécessaire. Nous avons ainsi conçu des algorithmes de routage tolérant aux fautes pour des réseaux mesh en 2D et en 3D. Ces deux algorithmes prennent en compte des séquences de voies alternatives pour les paquets lorsque la voie d'acheminement principale tombe en panne. Les algorithmes proposés permettent d'éviter plus de fautes et tolèrent des défaillances multiples dans le pire état du trafic.

2 Etat de l'Art

L'idée d'une infrastructure de communication de type NoC est de séparer les problèmes d'infrastructure de communication de l'application. Ainsi, un réseau peut être évolutif et configurable. Grâce à cette infrastructure de communication, une ressource matérielle peut se connecter en envoyant des messages à d'autres ressources dans le réseau. Une infrastructure générique de NoC est la combinaison de divers éléments matériels (éléments de traitement, commutateurs, liens) et de protocoles de communication (routage, commutation) qui déterminent l'architecture de communication.

L'algorithme de routage décide du chemin sur lequel les données doivent être transmises. Les algorithmes de routage peuvent être classer comme étant déterministe, semi-adaptatif (oblivious) ou adaptatif. Dans un routage déterministe, tout paquet prends le même chemin à partir d'un nœud source vers un nœud destination. Dans le routage semi-adatatif, la décision du chemin de routage est prise localement en fonction de sans prendre en compte l'état du réseau. Le routage adaptatif prends en plus l'état du réseau en compte afin d'éviter des zones encombrées (hot-spots) ou fautives.

La plupart des algorithmes de routage sont basés sur la technique de commutation "whormhole" en raison de sa simplicité.

La performance d'un algorithme de routage dépend principalement de deux facteurs : i) le nombre de hop entre deux nœuds et ii) la distribution des chemins. Sur la base du nombre de sauts requis (hop-count) les algorithmes de routage peuvent être classés en routage minimal et non-minimal. Un algorithme de routage minimal attribue des chemins minimaux entre des paires source-destination, tandis que l'algorithme de routage non minimal permet de prendre des chemins non-minimaux.

L'adaptativité permet d'alterner des chemins entre la même paire de nœuds source et de

destination, cette propriété permet de supporter de la tolérance aux pannes, ou de l'optimisation de performance en évitant des zones du réseau qui seraient surchargée.

l'idée de la technologie 3D est de rendre la distance entre les nœud plus près par rapport à une topologie 2D en empilant des circuits les uns sur les autres [3]. Les interconnexions par "Through Silicon Via" (TSV) conduisent à des distances plus courtes entre les deux couches.

Lors de la conception d'un SoC le compromis entre performances du réseau (c.-à latence, débit, charge de la communication), la consommation d'énergie et le coût silicium est primordial. Ainsi la qualité d'un NoC, peut être mesurée à partir de sa consommation d'énergie, de la taille de ses ressources et de ses performances. Les performances du réseau NoC sont sa latence, son débit et sa fiabilité. Ainsi, le concepteur de NoC doit prendre en compte l'impact de chaque paramètre sur les performances du résultat.

3 Impacts des paramètres de conception NoC sur les performances de transmission

Le défi à relever pour les concepteurs de SoC contenant un NoC est de trouver l'équilibre entre le coût et les performances (par exemple, la latence et le débit) du réseau. Nous avons dans un premier temps identifié les paramètres importants pour un NoC ainsi que ses performances principales. Nous avons ensuite établis différents scénarios de fonctionnement afin d'évaluer l'impact de la variation de chaque paramètre sur les performances du réseau. Pour évaluer ces scénarios nous avons utilisé le simulateur Noxim basé sur SystemC [7].

Nous avons identifié deux parties pour l'évaluation d'un NoC : la performance du réseau et le coût de mise en œuvre conformément à la figure 1. Les performances du réseau se composent de la latence, du débit et de la fiabilité tandis que le coût de mise en œuvre inclus la consommation d'énergie, la surface silicium du réseau et le nombre de ses ressources.

Lors de nos expérimentations nous avons identifié que la latence, le débit, la fiabilité et la consommation d'énergie sont principalement influencés par la capacité des ressources, la capacité du canal, de la topologie et la complexité de la tâche comme indiqué sur la 1. La capacité des ressources indique le maximum d'informations traitées par Processing Element (PE). Il représente la performance du routeur ou du processeur. La capacité du canal indique la quantité maximum d'information de données qui peut être transmis de manière fiable dans un canal.

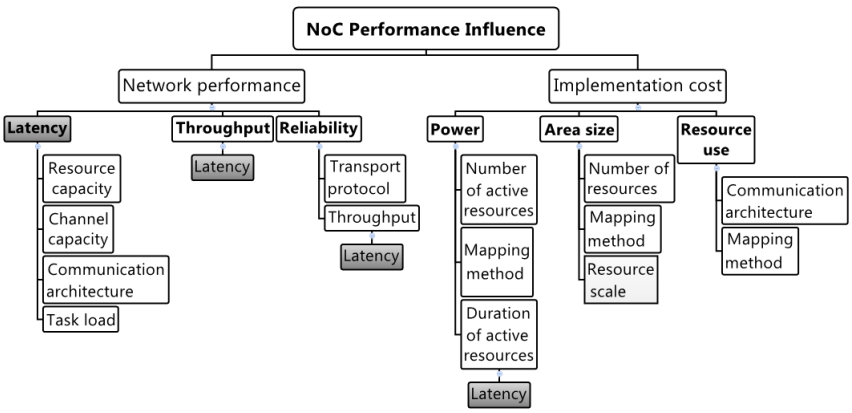


FIGURE 1 – Classification des performances des NoC et impact des paramètres.

Nous pouvons noter que la micro-architecture des ressources a un impact sur la surface et la consommation du réseau. L'augmentation du nombre de ressources augmente aussi la taille du chemin le plus long et donc la latence du réseau.

La méthode d'allocation des ressources et le positionnement des dites ressources dans le réseau influence la consommation et les performances du réseau.

La latence est l'une des performances requise pour permettre une grande qualité de service. Lors de conditions limités d'opérations la latence permet de représenter aussi le débit, la fiabilité et la consommation d'énergie.

scénarios de pires conditions En application réelle, les pires conditions se produisent lorsque la performance se dégrade à la limite de QoS. Ainsi, le système doit s'adapter en ajustant les paramètres qui ont un impact significatif sur l'augmentation des performances.

Le pire état arrive lorsque les performances du réseau sont au maximum, donc la mise à niveau des paramètres ne peuvent pas donner d'impact significatif sur la performance ou d'un réseau saturé. En condition normale, nous ne pouvons pas définir quel paramètre peut donner le plus d'impact sur les performances.

Nous avons évalué et définis le pire état en combinant huit paramètres : le nombre de hotspot, la taille de la mémoire tampon, la taille des paquets, le taux d'injection de paquets, le type de routage, la distribution de la circulation, de la stratégie de parcours de sélection et le nombre de ressources. Nous avons évalué les combinaisons de paramètres qui provoque les pires conditions.

Nous avons ainsi pu évaluer l'impact de chaque paramètre sur les performances du réseau.

scénarios de saturation réseau Chaque communication de réseau a une limite de performances de connexion connu comme limite de bande passante . Nous avons donc testé le réseau afin de déterminer cet état de saturation. L'objectif dans ce cas est de déterminer les combinaisons de paramètres provoquant la saturation. Dans les conditions testée nous n'avons pas saturé le réseau ce qui permet par la suite de valider les intervalles de valeurs des paramètres testés.

3.1 Incidence des paramètres sur les performances

Nous avons dans un deuxième temps évalué l'ampleur de l'impact de l'amélioration de la valeur des paramètres sur les performances du NoC et identifier le meilleur jeu de paramètre à adapter en cas de détérioration des communications. Les résultats montrent que la latence est la performances la plus sensible et un très bon indicateur de la qualité de service du réseau. Il est apparu que la diminution de la taille des paquets et du taux d'injection donne de bon résultat afin de maintenir la latence. L'algorithme de routage a lui aussi un impact important sur cette métrique.

En terme d'impact sur le débit, l'augmentation de la taille des paquets et du taux d'injection ont un grand impact. Ce qui va à contre sens de l'optimisation de la latence. Les autres paramètres n'ont pas un grand impact sur le débit.En ce qui concerne la fiabilité la diminution du taux d'injection permet d'améliorer les choses, mais le paramètre le plus important pour cette performance reste l'algorithme de routage. Finalement la consommation de puissance et les ressources utilisées dépendent grandement de la taille des buffers dans les routeurs, du nombre de ressources et de l'algorithme de routage.

4 Algorithmes de routage tolérants aux pannes pour réseaux mesh 2D et 3D

L'originalité de nos algorithmes de routage vient de la façon dont ils classifie le nœud de destination et comment ils choisissent la séquence de chemins alternatifs.

4.1 algorithme de routage Gradient pour mesh 2D

4.1.1 Algorithme

Les algorithmes de routage tolérants aux pannes classiques classent les nœuds destination dans quatre zones, en fonction de lignes verticales et horizontales passant par le nœud courant. *Gradient* classifie l'adresse de destination en huit zones basées sur la ligne gradient (M) en coordonnées 2D tel que présenté dans la figure 2. La ligne gradient (M) est obtenue grâce à l'équation 1

$$M = \frac{D_y - C_y}{D_x - C_x} \tag{1}$$

avec (Dx, Dy) l'adresse de destination et (Cx, Cy) l'adresse du routeur actuel.

Nous utilisons alors la ligne de gradient $left|M\ right| = 1$ pour diviser le réseau en huit zones (Fig. 2). Sur la base de ce positionnement relatif de l'adresse de destination une zone est assignée pour déterminer le saut suivant (c'est à dire le routeur suivant dans la chemin).

Pour chaque zone du nœud de destination pour le paquet courant, l'algorithme *Gradient* définit un itinéraire principal et trois variantes. La décision du chemin utilisé est alors prise à l'exécution en fonction des conditions du réseau. Par exemple, pour une destination dans la *Zone- 1*, la route principale est Est qui fournit le chemin le plus court vers la destination.

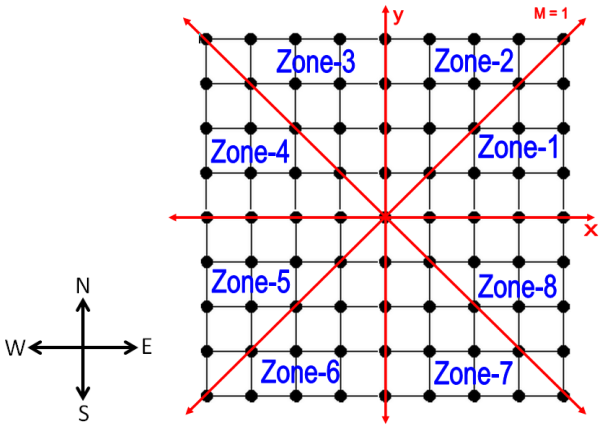


FIGURE 2 – Gradient divise les coordonnées 2D en huit zones

Le premier itinéraire alternatif est au Nord qui peut également assurer chemin le plus court et enfin au Sud, si il n'y a pas de saut possible dans les premiers chemins.

Outre le nombre de zone permettant de mieux localiser la destination, la différence repose également sur la séquence de choix de la route alternative. La séquence intégrée dans *Gradient* permet de minimiser la taille des routes alternatives en cas de fautes sur le chemin principal.

Gradient est indépendant de la méthode de détection de défaut. La position des "hots-pots" et des fautes sont définis statiquement dans notre travail. Afin d'éviter une zone de "hotspot" ou un élément défectueux dans le réseau, *Gradient* choisit la voie alternative la plus courte. le choix de ce chemin plus court permet d'éviter la dégradation de la performance du réseau tels que la latence et le débit.

4.1.2 Evaluation des performances

Nous avons évalué huit algorithmes de routage différents, à savoir XY, West-first, North-last, Negative-first, Odd-Even, fully-adaptive qui sont mis en place et intégré dans le simulateur Noxim [8], l'algorithme à tolérance de panne présentée dans [9] et *Gradient*. Tous ces algorithmes ont été évalués selon différents scénarios. ces scénarios sont conçus pour évaluer le nombre de sauts et le nombre de chemins alternatifs de chaque algorithme de routage dans différentes conditions. Le résultat montre que notre algorithme a le plus petit nombre de sauts et le plus grand nombre de routes alternatives par rapport aux autres algorithmes existants.

Nous avons évalué la latence dans un réseau défectueux en augmentant la le taux d'injection de paquets. Nous avons utilisé dans cette évaluation une topologie mesh 6x6 avec deux nœuds fautifs dans le centre du réseau et quatre nœuds fautifs près de la frontière du réseau.

Les résultats montrent que pour des taux d'injection de paquets entre $5,10^{-4}$ à $2{,}10^{-3}$ la quasi-totalité des algorithmes de routage ont des performances similaires. A partir d'un taux d'injection de $2,10^{-3}$ l'algorithme *Gradient* a le plus faible retard moyen. Nous avons également évalué les performances des différents algorithmes de routage en augmentant le

pourcentage de défaillance des nœuds, soit le pourcentage de défauts des routeurs variant de 0 % = normal, à 100 % = totalement défectueux. Dans ce scénario, nous utilisons un réseau mesh de taille 5x5et un autre de taille 6x6, avec un taux d'injection de paquets de 0,005 paquets/cycle/IP. Les résultats montrent que notre algorithme a la plus faible retard moyen dans tous les cas et la différence augmente avec l'aggravation des conditions d'opérations des réseaux. Enfin, nous avons évalué l'évolutivité de l'algorithme en augmentant le nombre de nœuds dans le réseau. Le résultat montre que l'algorithme proposé a un nombre minimum de saut, un plus grand nombre de chemins alternatifs, des latence inférieures et un débit supérieur que les autres algorithmes de routage adaptatifs.

4.1.3 Mise en œuvre sur FPGA

Algorithme de routage gradient a été mis en œuvre dans un routeur du réseau HERMES [11]. Ce réseau bien connu a été prototypé avec succès sur Virtex-II FPGA Xilinx [12]. Les éléments de base de ce routeur sont une logique de commande de commutation et cinq ports bi-directionnels (connexion à quatre autres commutateurs et une connexion locale au PE). La logique de commande de commutation est constituée de l'arbitrage et de l'algorithme de routage que nous avons modifié.

Nous avons intégré un réseau mesh de dimension 4x4. La synthèse a été réalisée avec les outils ISE 12.4 [13] pour un FPGA Virtex-5. Les résultats de synthèse (tableau 1) montrent que l'algorithme de routage XY utilise moins de ressources cependant cet algorithme n'est pas adaptatif. Dans les autres cas les algorithmes de routage adaptatif tel que le notre utilisent presque les mêmes ressources du FPGA ciblé. Ainsi, l'algorithme Gradient est préférable car il nécessite les mêmes nombre de ressources, mais a une meilleure performance réseau par rapport aux autres algorithmes.

TABLE 1 – Ressources FPGA nécessaires pour mettre en œuvre le contrôle de flux handshaking sans canal virtuel ni ordonnancement

Routing	Slice Register	Slice LUT	LUT Flip- Flop	BUFG / BUFGCTRLs
XY	0,11	0,52	0,21	0,06
Turn Model	0,11	0,53	0,21	0,06
Gradient	0,11	0,53	0,21	0,06

4.2 Algorithme de routage pour réseau mesh 3D

4.2.1 Algorithme

Comme pour Gradient, l'originalité de *Diagonal* est la façon dont l'algorithme choisit la séquence de chemins alternatifs lorsque le paquet est confronté à un lien en défaut dans les NoC en technologie 3D. L'algorithme proposé considère la distance et la direction du nœud de destination en fonction de la position du nœud courant.

Les distances sont obtenus à partir du rapport entre les valeurs absolus de $\triangle X$, $\triangle Y$ et $\triangle Z$. Alors que les directions sont obtenues directement à partir de ces valeurs.

L'algorithme choisit la distance la plus longue comme route principale, la seconde distance la plus éloignée comme première alternative et la troisième plus éloignée pour l'alternative-

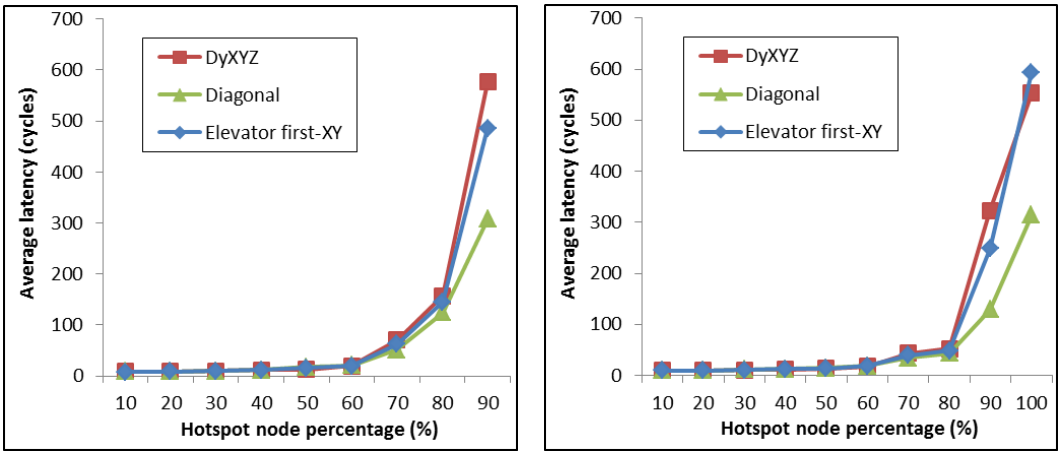
2. Pour l'alternative-3 l'algorithme choisit la route l aplus courte dans le sens opposé à la distance. Sur la base des zones définies, notre algorithme présente un minimum de hop et plus de voie alternative pour atteindre le nœud de destination. Par exemple, si la distance de destination-Xà partir de courant-X ($|\triangle X|$) est plus loin que $|\triangle Y|$ et $|\triangle Y|$ est plus loin que $|\triangle Z|$, l'algorithme choisira X comme voie principale, Y comme première alternative, Z comme la seconde variante, - Z pour troisième variante, -Y en quatre et enfin -X en tant que cinquième itinéraire alternatif.

4.2.2 Evaluation des performances

Dans ce paragraphe, nous présente l'évaluation du nombre de sauts pour chaque algorithme de routage. Nous avons évalué trois algorithmes de routage différents, adaptative-XYZ, Elevator-first [10] et *Diagonal* pour une topologie mesh 3x3x3. Les résultats de l'évaluation du nombre de sauts pour chaque algorithme de routage montre que *Diagonal* à le plus petit nombre de sauts requis dans tous les scénarios testés.

Afin d'évaluer Diagonal, nous avons modifié Noxim pour intégré les NoC mesh 3D et des nœuds fautifs. Nous avons alors dégradé l'état du réseau en augmentant le nombre de nœuds défectueux et le taux d'injection de paquets sur différentes taille de réseau.

Dans un premier lieu, nous prenons un réseau de 3x3x3 avec un taux d'injection de 0004 (paquets/cycle) et deux nœuds défectueux. Nous augmentons ensuite le pourcentage de nœuds fautifs et nous évaluons la latence moyenne des paquets. Le résultat montre que l'algorithme de routage proposé a un délai moyen inférieur que les autres algorithmes de routage comme illustré sur la figure 3.a. Ce résultat reste vrai pour une taille de réseau plus grande, un taux d'injection de 0,0015 (paquets / cycle) et trois nœuds défectueux dans le réseau (Fig. 3.b) montre que l'algorithme proposé ont également délai moyen inférieur comme dans le scénario précédent.



(a) comparaison moyenne de retard pour un mesh 3x3x3 avec PIR 0004

(b) comparaison moyenne de retard pour un mesh 4x4x4 avec PIR 0,0015

FIGURE 3 – Comparaison moyen de retard sur (a) mesh 3x3x3 avec un taux d'injection de paquets 0004 et (b) mesh 4x4x4 avec PIR 0,0015

5 Conclusion et perspectives

L'évaluation de l'impact des paramètres de conception d'un NoC sur sa performance a été présenté. Les résultats peuvent être utilisés pour concevoir un mécanisme de commande afin d'éviter la dégradation de la qualité de service lorsque les conditions de communications change dans un NoC adaptatif. Dans l'avenir, ce travail sera la base d'une application logicielle permettant l'évaluation au cours de la conception de l'impact des choix du concepteur sur la performance du réseau.

Les algorithmes de routage *Gradient* et *Diagonal* ont été proposés comme algorithme de routage à haute disponibilité pour surmonter le blocage de paquets pour des réseaux mesh en technologie 2-D et 3-D. Les séquences de choix d'itinéraires alternatif dans nos algorithmes permettent de choisr le plus court chemin pour éviter des zones de contention ou des éléments fautifs dans le réseau. L'implémentation dans un FPGA a montré que nos algorithmes ne nécessitent pas plus de ressources que les autres algorithmes tout en supportant de meilleure performance dans un environnement avec des éléments fautifs.

Références

[1] R. Koch, T. Pionteck, C. Albrecht, and E. Maehle, "An adaptive system-on-chip for network applications," in *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, 2006, p. 8.

[2] W. Dally and C. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Transactions on Computers*, vol. C-36, no. 5, pp. 547 –553, may 1987.

[3] V. Pavlidis and E. Friedman, "3-d topologies for networks-on-chip," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 10, pp. 1081 –1090, oct. 2007.

[4] A. W. Topol, D. C. L. Tulipe, L. Shi, D. J. Frank, K. Bernstein, S. E. Steen, A. Kumar, G. U. Singco, A. M. Young, K. W. Guarini, and M. Ieong, "Three-dimensional integrated circuits," *IBM Journal of Research and Development*, vol. 50, no. 4.5, pp. 491 –506, july 2006.

[5] W. Dally and B. Towles, "Route packets, not wires : on-chip interconnection networks," in *Design Automation Conference (DAC), 2001*, 2001, pp. 684 – 689.

[6] D. Fick, A. DeOrio, G. Chen, V. Bertacco, D. Sylvester, and D. Blaauw, "A highly resilient routing algorithm for fault-tolerant nocs," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2009.*, april 2009, pp. 21 –26.

[7] F. Fazzino, M. Palesi, and D. Patti, "Noxim : Network-on-chip simulator," http ://noxim.sourceforge.net.

[8] M. Palesi, D. Patti, and F. Fazzino, *Noxim - the NoC Simulator - User Guide*, University of Catania, 2010.

[9] F. Chaix, D. Avresky, N.-E. Zergainoh, and M. Nicolaidis, "A fault-tolerant deadlock-free adaptive routing for on chip interconnects," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, march 2011, pp. 1 –4.

[10] F. Dubois, A. Sheibanyrad, F. Pe ?trot, and M. Bahmani, "Elevator-first : A deadlock-free distributed routing algorithm for vertically partially connected 3d-nocs," *Computers, IEEE Transactions on*, vol. 62, no. 3, pp. 609–615, 2013.

[11] Atlas noc simulator documentation https ://corfu.pucrs.br/redmine/projects/atlas.

[12] F. Moraes, N. Calazans, A. Mello, L. Moller, and L. Ost, "Hermes : an infrastructure for low area overhead packet-switching networks on chip," *Integration, the VLSI Journal*, vol. 38, pp. 69–93, 2004.

[13] Xilinx, *Synthesis and Simulation Design Guide - ISE 12.4*, 2008.