UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM MICROELETRÔNICA

CAROLINE MARTINS CONCATTO

Coping with Permanent Faults in NoCs by using Adaptive Strategies based on Router Design-level and Routing Algorithm-level

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Microeletrônica

Prof. Drª. Fernanda G. de L. Kastensmidt
Orientadora

Porto Alegre, Novembro de 2009.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Concatto, Caroline Martins

Coping with Permanent Faults in NoCs by using Adaptive Strategies based on Router Design-level and Routing Algorithm-level / Caroline Martins Concatto – Porto Alegre: Programa de Pós-Graduação em Microeletrônica, Novembro de 2009.

69 p.:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Microeletrônica. Porto Alegre, BR – RS, 2009. Orientadora: Fernanda Gusmão de Lima Kastensmidt.

1.NoC. 2.Fault Tolerance 4.Microeletronics. I. Kastensmidt, Fernanda G. de L.. II.Título.

NoC, Fault-tolerance, adaptability, performance, microeletronics

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. Carlos Alexandre Netto
Vice-Reitor: Prof. Rui Vicente Oppermann
Pró-Reitor de Pós-Graduação: Prof. Aldo Bolten Lucion
Diretor do Instituto de Informática: Prof. Flávio Rech Wagner
Coordenador do PPGMicro: Prof. Ricardo Reis
Bibliotecária-Chefe do Instituto de Informática: Beatriz Regina Bastos Haro

AGRADECIMENTOS

Gostaria de agradecer à minha mãezinha que amo, minha irmã querida e ao meu maridão Alex, esses fizeram parte do meu cotidiano fora da UFRGS, me tiraram um pouco do mundo acadêmico ou me fizeram estudar mais uns 5 minutos.

Dos professores do PGMicro e PPGC, principalmente à minha orientadora (a Fe), e ao Luigi que acredita mais em mim do que eu mesma (e há quem diga que ele é muito exigente). À profa. Erika por tudo que me ajudou até aqui. Ao prof. Luba e à nossa adorável e bela secretária Zig. Aos meus queridos colegas de laboratório e em especial, minha ex-colega de faculdade, colega de mestrado e parceira de artigos mil, Débora.

À toda coordenação do PGMicro, encabeçada pelos professores Ricardo Reis e Flávio Horowitz. Ao Instituto de Informática e à UFRGS em geral.

E também, sem deixar o lado espiritual de lado, gostaria de agradecer às forças sobre naturais que fazem com que a gente continue estudando e indo até o final.

A todos os meus mais sinceros Agradecimentos.

CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS

AVF	Area Vulnerable to Fault
ATE	Automatic Test Equipment
BIST	Built-In-Self-Test
CPU	Central Process Unit
CMOS	Complementary Metal Oxide Semiconductor
DBP	Default Backup Path
ECC	Error Correction Code
EDRAM	Embedded Dynamic Random Access Memory
EW	East to West
FIFO	First-In-First-Out
FSM	Finite State Machine
FTRA	Fault Tolerant Routing Algorithm
FTAR	Fault Tolerant Adaptive Router
IC	Integrated Circuit
IoCs	In-order Cores
I/O	Input/Output
IP	Intellectual Property
MPEG4	Moving Picture Experts Group 4
MWD	Multi-Window Display
NoC	Network-on-Chip
NS	North to South
OoCs	Out-of-order Cores
QoS	Quality of Service
AR	Adaptive Router
SoC	System-on-Chip
TDG	Test Data Generators
TED	Test Error Detection

TMR	Triple Modular Redundancy
TR	Time Redundancy
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VOPD	Video Object Plane Decoder

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Nowadays, networks-on-chip (NoCs) have been used as an alternative communication architecture inside complex system on-chip. They offer better scalability and performance than the traditional bus. However, the growing number of interconnects that have to be inserted using smaller transistors means that NoCs have a growing number of faults, either from manufacturing or due to aging. In future systems-on-chip (SoCs), the fault rate will be around 20 to 30% of the contact and transistors of integrated circuits. Therefore, even in the presence of a fault, it is still desirable that NoCs properly work.

The main idea of this work is to implement adaptive mechanisms to protect NoCs against permanent faults. The main advantage of such mechanism is to manage failures based on data from the testing and diagnosing phase. The mechanisms are adapted in each router in order to sustain performance, increasing the system yield and reliability even in the presence of failures.

Even if one adds extra blocks for replacement, the occurrence of permanent faults in a NoC might preclude the replacement or repair of a faulty component within the SoC. In such case, fault-tolerant NoCs are able to reduce manufacturing costs, increase yield and the lifetime of the chip.

Keywords: NoC, Fault-tolerance, adaptability, performance, microeletronics.

Cobrindo falhas permanentes em Redes intrachip usando técnicas adaptativas nos roteadores em um nível de projeto e em um nível de algoritmo

RESUMO

Hoje em dia, as redes intra chip (NoC) são cada vez mais utilizadas como uma arquitetura de comunicação alternativa para sistemas complexos, pois estas permitem flexibilidade e desempenho da comunicação. Porém, o grande número de interconexões da rede, aliado à diminuição das dimensões dos transistores fabricados nas tecnologias nanométricas, fazem com que a NoC possa ter um grande número de falhas durante sua fabricação, ou por desgaste durante sua vida útil. Sabe-se que, em futuras tecnologias os circuitos integrados terão uma taxa de falhas permanentes de 20 a 30%. Entretanto, mesmo na presença de falhas, é desejável que a NoC permaneça funcionando corretamente.

A partir do diagnóstico das falhas, a NoC deve ser capaz de buscar alternativas para manter a comunicação entre os núcleos, evitando os canais e os roteadores com falhas. O objetivo deste trabalho é propor mecanismos adaptativos de proteção contra falhas permanentes.

Mesmo quando são adicionados componentes extras para a substituição em SoCs, a ocorrência de falhas permanentes na rede intrachip impede a substituição ou reparo de um componente no sistema intrachip. Portanto a tolerância a falhas na NoC será crucial para reduzir custo de manufatura, e aumentar o rendimento e o tempo de vida do circuito integrado.

O mecanismo proposto é capaz de evitar falhas sabendo anteriormente, na fase de teste e diagnóstico, a localização especifica da falha. Portanto, as técnicas se adaptam em cada roteador para evitar as falhas permanentes, sempre buscando manter desempenho, aumentar o rendimento e a confiabilidade do sistema.

Palavras-Chave: NoC, Tolerância a Falhas, adaptabilidade, performance, microeletrônica.

1 INTRODUCTION

Network-on-Chip (NoC) is an efficient communication architecture for Systems-on-Chip. It allows the integration of a huge number of computational blocks in only one integrated circuit. NoC allows the reuse of blocks, which provide a high degree of parallelism and it may present high improvement in performance (BENINI and DE MICHELI, 2002). The increasing need of NoC has been driven by the demand for high bandwidth in communication. According to Moore's law, transistor integration doubles in approximately eighteen months, but as the delay of the global wires has increased, physical connection in an IC (Integrated Circuit) is the performance bottleneck (HOROWITZ et al., 2000). NoC can provide short interconnections, which can reduce the wire delays.

The traditional methods (end-to-end, bus-based and others) are not a good solution to integrate many IP cores in System-on-Chips (SoCs), because they do not scale well as more components are added to the system. Ad hoc solutions that use multiple buses, crossbar and dedicated wiring can achieve near optimal performance for a specific application but they are hard to reuse and have a high development and verification cost (PANDE et al., 2003). The use of Network-on-chip (NoC) to replace the traditional methods of interconnection has advantages for performance and modularity. With the use of a NoC, it was observed that electrical properties can be optimized, power consumption can be decreased by 10 times and speed increased by 3 times when compared to bus-based interconnections (DALLY and TOWLES, 2001).

A basic NoC infrastructure is composed of routers, links and network interfaces. The routers route the data through the links. The network interfaces adapt the data coming from or going to the core of the network. The topology defines the layout of the network. The routing algorithm decides the path between source and destination. Figure 1.1 exemplifies a 3x3 Grid NoC where 'acc' is accumulator, 'mem' is memory, 'FPU' is the Floating Point Unit, 'I/O' is the input and output block and 'CPU' is the processor.

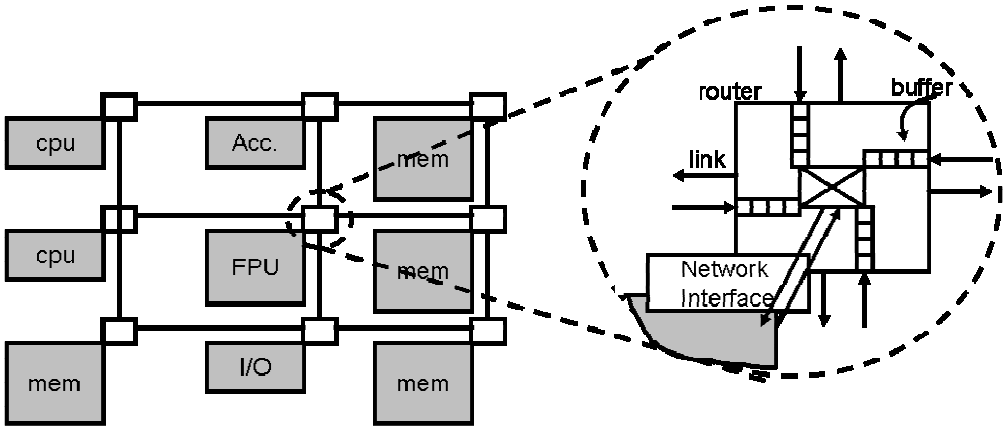


Figure 1.1: NoC 3x3 2-D Grid with 9 routers.

With the advance in technology, as the number of interconnections increases and the transistor dimension shrinks, NoC will probably have a high number of faults in its interconnections and logics (during the manufacture, or during its lifetime). According to the projections of INTEL (FURBER, 2006) "*within a decade we will see 100 billion transistor chips*", however, "*20 billion of those transistors will fail in manufacture and a further 10 billion will fail in the first year of operation*". Therefore, 20% to 30% will be the fault rate in a System-on-Chip. The effect of faults can be permanent, intermittent and transient. Here in this Thesis, we address permanent faults.

In the future of any SoC architecture, it is very likely that in order to be fault-tolerant, a SoC will contain a large number of replicated components and will employ a fault-tolerance approach to deactivate the defective component, i.e. microprocessor or memory, routers and wires. Once a faulty component has been detected, the system software application can be used to initiate the remaining hardware components.

To apply a fault tolerant technique in the NoC, first it is necessary to test and diagnose the communication infrastructure. However, the test has some challenges (how to detect the fault, how to identify the faulty components: routers, wires, cores and how to use efficiently the remaining hardware components that are fault free). Faults can be detected by inserting input vectors capable of manifesting them at the output. The test can be performed by the use of ATE (Automatic Test Equipment) or BIST (Built In Self Test). Cota et al. (2004), has proposed the use of a NoC as a way to apply these vectors to test the cores of a SoC, performing functional testing. This work has been extended to detect faults in the NoC links by Cota et al. (2008). More recently, Herve et al. (2009) has proposed a method of diagnosing the location of faults in the NoC links. The information of the faults location can be used by a fault tolerance technique to isolate the faults and to determine the fault free components.

In this Thesis, we propose two different techniques to cope with permanent faults in the network. We assume that faults can occur in the routers and in the links. Both strategies aim to increase system reliability by isolating and avoiding known faults. The first strategy is based on the idea of designing the router with extra components so that faults in the router can be tolerated. The idea is to ensure the NoC behavior by using the router with a different configuration providing minimum performance and power degradation. This technique is a design level strategy. At this level, we have used techniques to protect the NoC against permanent faults in the router buffers, in router

FSM, links and crossbar. However, as the number of faults increases, this technique becomes less cost effective.

So, this brings the necessity of an algorithm level strategy. In this case, the routing algorithm is adapted in the presence of permanent faults in the links. With the algorithm level strategy, faulty links and routers are avoided. According to the NoC size, the performance degradation can be minimal.

By using the solutions proposed in this Master Thesis, the yield can be increased because once faulty, a SoC can be used as the faults can be isolated. In the two proposed strategies, we show results in area, frequency, power.

1.1 Organization of Text

This thesis is organized as follows:

- Chapter 2 – The case study platform is presented.
- Chapter 3 – The problem of coping with permanent faults in NoC is defined and some state-of-the art works to solve the problem are presented.
- Chapter 4 – The first scheme of fault tolerance is proposed, where results are shown in terms of performance, area, power and frequency.
- Chapter 5 – Adaptive Routing Algorithm is presented. Results in power, frequency, area and performance are presented.
- Chapter 6 – The proposed strategies are compared to two of the state of the art techniques.
- Chapter 7 - Conclusion and future work are discussed.

2 NOC PLATFORM

This chapter presents the SoCiN NoC platform used in this thesis (System-on-Chip, Interconnection Networks) proposed by Zeferino and Susin (2003). Its topology can be configured as 2D-Grid or Torus. The Network SoCIN has a wormhole packet switching, a deterministic routing algorithm, input buffering, handshake control flow and round-robin arbiter. This chapter presents details of the topology, switching, routing, buffering, control flow and others characteristics.

The main block of SoCIN is the router RASoC (Router Architecture for Systems-on-Chip). RASoC is a VHDL soft-core which has tree parameters of change: channel width, buffer depth and the information of the routing in the header packet, also the number of ports can be configured. These decisions are made in design time (channel width, buffer size and number of ports).

The topology most common of SoCIN is 2D Grid, shown in fig. 1.1. In fig. 1.1 the white boxes are the routers, the gray boxes are the cores and the arcs between the routers represent the communication links. One feature of a 2D Grid is that it can be easily extended to 2D torus, adding arcs on the edged routers. The torus network has the advantage of decreasing the diameter of the network. Nevertheless, in torus topology the size of the wires are larger than in Grid topology, consequently there is a lower frequency due to the capacity in the long wires.

The links of the SoCIN are unidirectional channels, called simples (fig 2.1). The simplex channel has n bits of data and 2 bits for control and so n+2 is the size of the channel. The 2 bits for control are used to inform if the flits are header, data or tail. The parameter n is defined at design time according to the need of the system. In addition to the n+2 bits, the simplex channel needs 2 other signals (*val* and *ack*) to do the handshake protocol, these two signals are explained together with the input buffering.

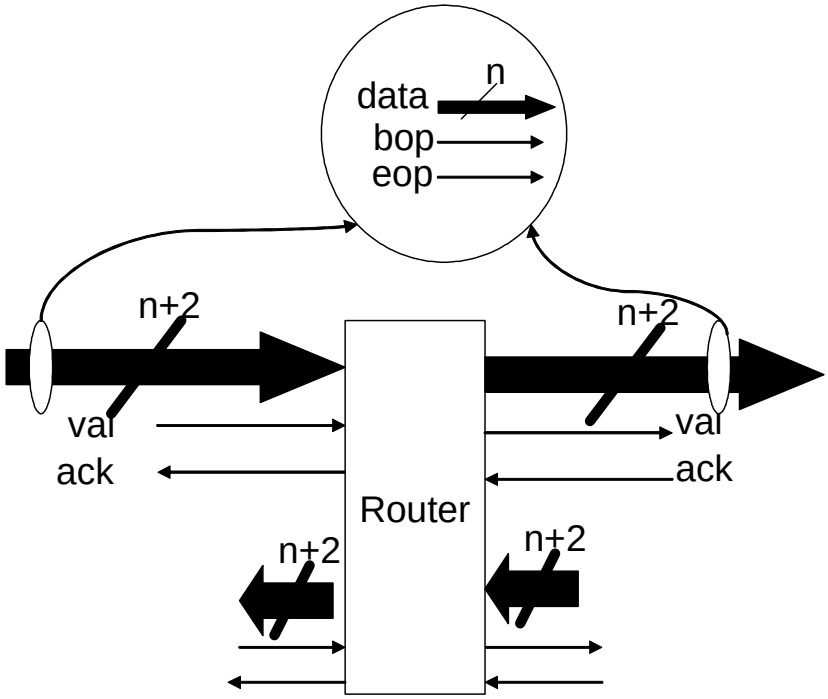


Figure 2.1: Link configuration of SoCIN.

The switching is wormhole. The packets are made by flits and each flit has the same size as the channel, in this case *n*+2 bits. The first flit of the packet is the header. The header flit has the information needed to make the path between the source and the destination. The following flits have the data and they follow the header flit through the network like a pipeline. Fig. 2.2 shows a packets split in flits, the first flit is the header flit (H) the framing signals are “10”, this flit is followed by the data flits (P) where the framing signals are “00”.The Tail flit (T) comes to release the path and the framing signals are “01”.Each router of the network has the ability to store a few flits in the input channel.

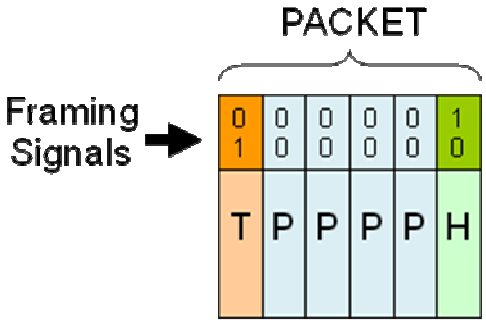


Figure 2.2: A Packet.

The routing algorithm is deterministic, that is, the path used for the packet is determined by the source and the destination and the same path is always chosen. The XY routing algorithm is deadlock free and it has a low cost of implementation, however its performance is lower when the traffic or fault rates are high.

XY routing algorithm first routes the packet in the X axis until the packet reaches the desired column and after the packet is routed in the Y axis. Once the Y axis is taken

the packet cannot be routed in the X axis again. The XY routing algorithm is deadlock free because there is not a cyclic dependency between the channels, a packet will never be routed from Y axis to X axis and this can be seen in fig 2.3.

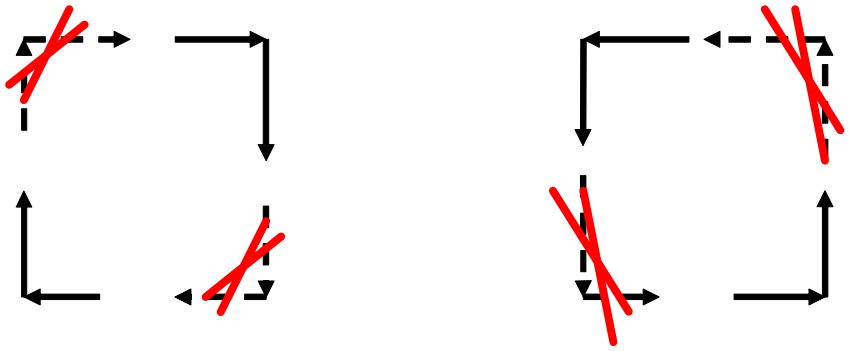


Figure 2.3: Permitted turns and non permitted in XY routing algorithm.

SoCIN has input buffering, that is, each input channel has a buffer to store the incoming flits. In SoCIN each input channel has FIFOs to store and to read flits before being sent to the output channel. Each FIFO buffer has the capability to store *p* flits of *n*+2 bits, where *p* is the parameter chosen at design time to satisfy the needs of the system. In this thesis we choose the value of *p* equal to 4.

The control flow used for SoCIN is the handshake. The source informs the destination the intention to send a packet. The destination answer given is “yes” or “not”. If the answer is “yes” the source will send flits to be stored in the FIFO of the destination, for “no” the flit is not sent and stays stored at the source´s FIFO. The answer is “yes” when the destination has space in its FIFO buffer and “no” when FIFO is full. The signals *val* and *ack*, used to do the handshake protocol, are signals coming from the FIFO. These signals inform when the flit can be stored or not in the FIFO buffer.

3 STATE-OF-THE-ART TECHNIQUES TO COPE WITH FAULTS IN NOCS

3.1 Faults in Network-on-Chip

With the technology scaling and the transistors dimension shrinking, fault tolerance in the systems is becoming a key challenge for designing NoCs and research in the field has gained significant momentum. Fig.3.1 and Fig. 3.2 show the fault location in a NoC-based SoC. In fig 3.1, a core with permanent fault is shown. The permanent fault can affect the correct functionality of the NoC, that is, it can happen in the router, or in the channel. In the worst case the fault can affect the routers and channels, as depicted in fig 3.2 (c). In the router, the fault can occur in all components of the router (crossbar, handshake protocol, buffers and others). The fault can happen in the channels, resulting in a shortcut between different channels, between the same channel, between the link and the VDD or between the link and the ground. The permanent fault can result in a wire-and, wire-or, stuck-at 0 or stuck-at 1.

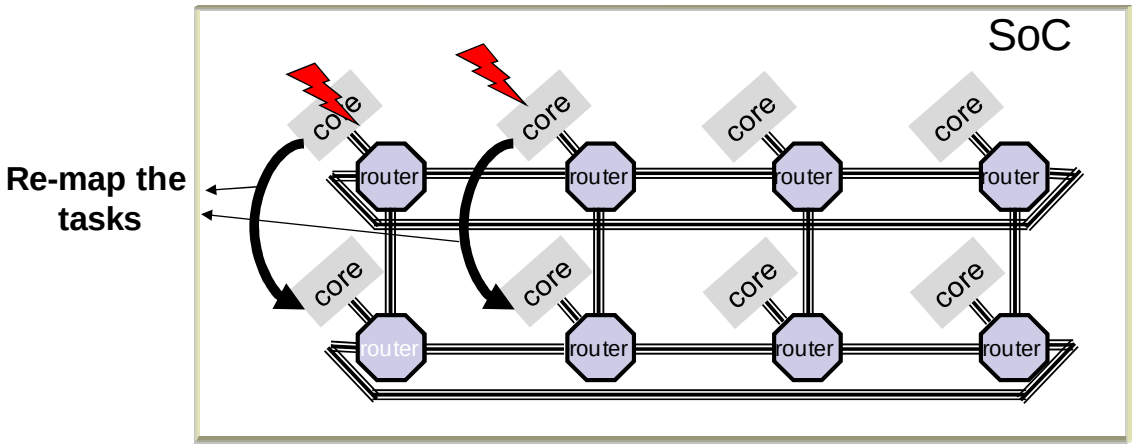
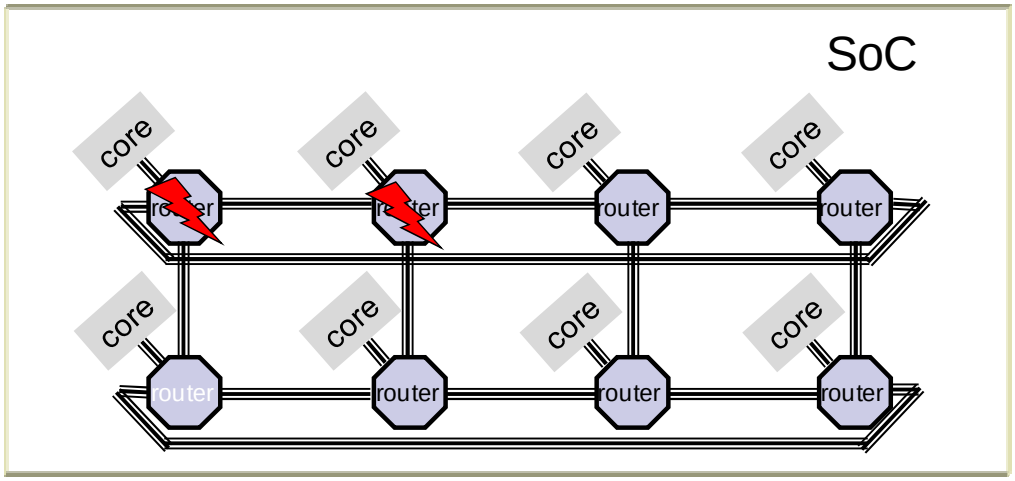
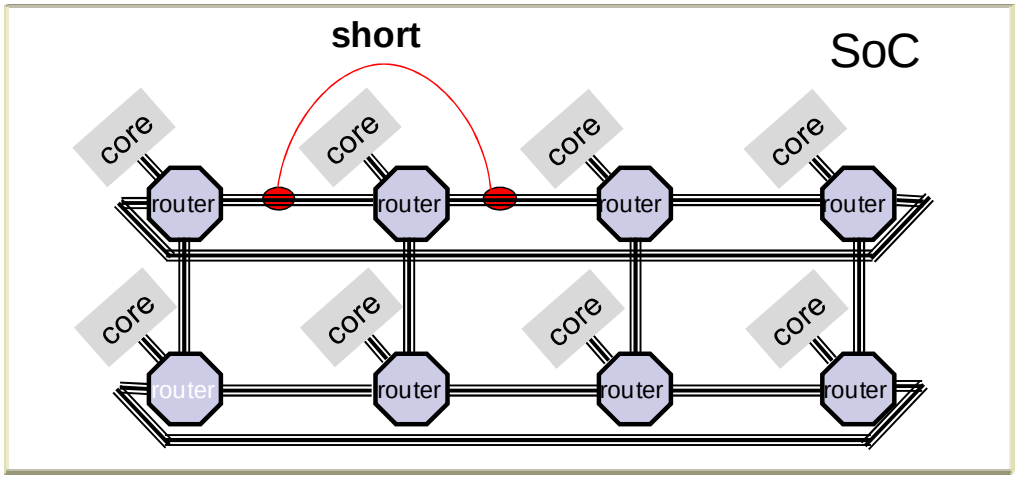


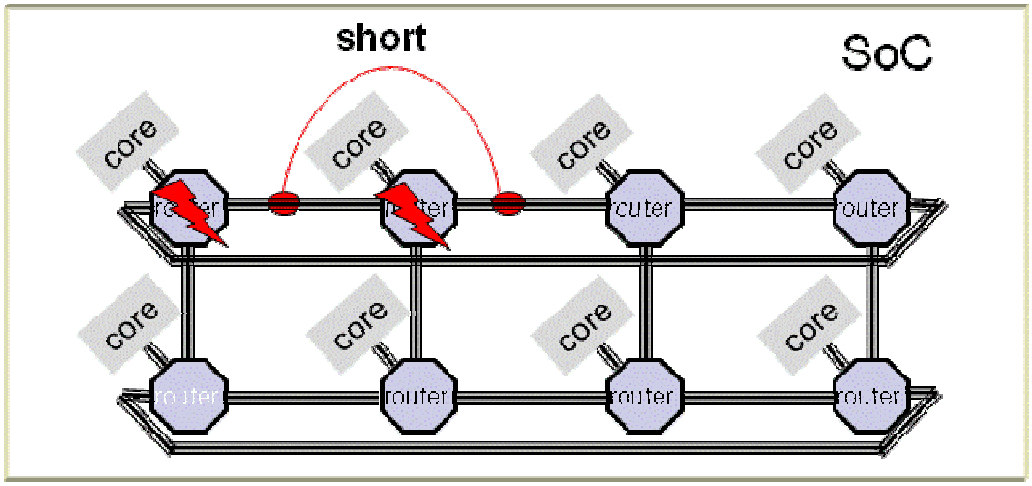
Figure 3.1: Fault Location in a SoC NoC-based of the cores.



(a)



(b)



(c)

Figure 3.2: Fault Location in a SoC NoC-based (a) routers; (b) channels; (c) router and channels.

3.2 Related Works able to cope with Faults in NoC

To protect the NoC infrastructure some works have proposed fault tolerant solutions with well-know techniques, such as Triple Modular Redundancy (TMR) or Time Redundancy (TR) (NICOLAIDIS, 2005) to protect the NoC against faulty components. However, most of these works based on hardware redundancy severely increase the area and power consumption to protect the NoC router, because when one triplicates the hardware, one also triplicates the power consumption. On the other hand, Time Redundancy decreases the performance of the NoC, since all information needs to be retransmitted or have a delay.

Hamming code is a linear error-correcting code. It can detect up to two simultaneous bit errors and correct single-bit errors. In (Frantz et al., 2006) the use of Hamming Code on the input buffers has been proposed to protect the FIFO data. The incoming data is encoded before being stored in the buffers and decoded when it leaves the buffers. This work protects the buffers. To protect the link another technique is used. The hamming code is implemented using a chain of XORs to code and decode the data, thus it shows huge area and performance overheads, around 50% increase in the area. Frantz's works do not show results of performance, but Bo Fu and Ampadu (2008) shows a decrease of 30% in the throughput. Therefore, one can conclude that the use of hamming code is too expensive to be applied in all NoC components.

The most common fault tolerance technique known is redundancy, as shown in (YU and AMPADU, 2008) and (ROSSI et al., 2005). In Yu and Ampadu (2008) the authors use adaptability to select the ECC (Error Correction Code) scheme. When data comes from the neighbor router, the router chooses the most effective ECC scheme to send the data trough the links. However, using the fault tolerance technique proposed in (YU and AMPADU, 2008) and (ROSSI et al., 2005) only the links are protected and, furthermore, in Yu and Ampadu this technique is specially used to avoid faults caused by noise, such as interference. In both methods no technique was used to protect the routers.

To protect the link of the NoC the most common strategy used is to change the routing algorithm. The latency of the routing algorithm strategy is lower than re-transmission and the cost is less than TMR. In Frazzetta et al. (2008), series of routing strategies to utilize NoC systems with partially faulty links are proposed.

Frazzeta et al. (2008) proposes to use partially faulty links when the traffic in the network is high. The idea is to distribute the traffic uniformly between links. The links capacity can be split in 25%, 50%, 75% or 100%, according to the faults in the link. The names of the three strategies are Faulty Link Elimination Strategy, Partially Fault Links Usage Strategy and Partially Faulty Links Usage with Look-ahead strategy. In the first one, the faulty link is never used, in the second the faulty link can be used and in the last one the faulty link is used. Fig. 3.3 shows the architecture of the router. It has a block to verify the network congestion and an input and output register in the links to store pieces of the link before being used by the router.

The analogy of Frazzeta's work is with roads in real life. For instance, when two roads go to the same place: one is empty of cars (no one is using the road), but only half of the road can be used; the second road is free of faults, but a lot of cars are using this road. Which one must be chosen? For Frazzeta, the first road is empty and is chosen.

The same is proposed for the network, if one part of the link can be used and the channel has low congestion, this channel must be used.

With this proposal, Frazzetta et al. (2008), save, in the best case, 12% of energy consumption, because it can choose the minimum path even in the presence of a fault. However, the area overhead for the same strategy is almost 40%, but on the other hand the delay of the network is reduced by 50% for the same reasons that can reduce the energy consumption. In Frazzeta, only the links are protected, that is, the routers are vulnerable to fault detection.

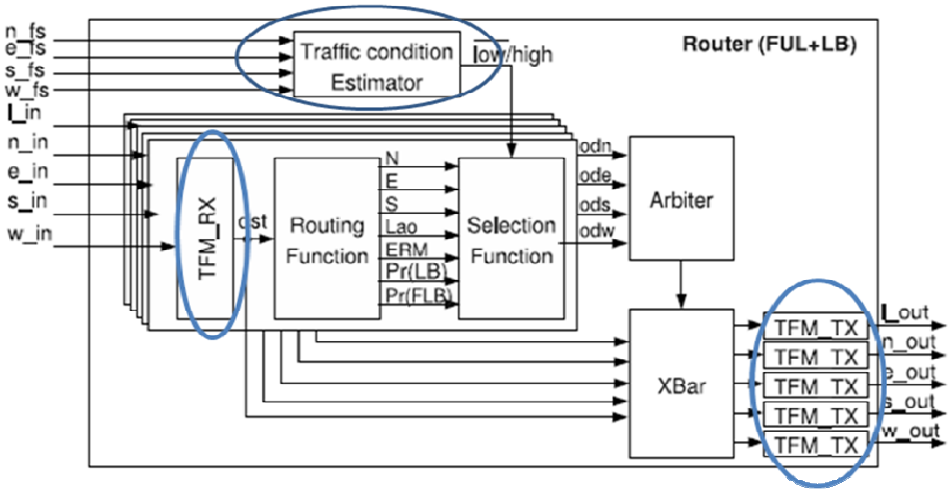


Figure 3.3: Frazzeta´s router.

In Schonwald et al., (2007) a table is used that has the information about the number of hops needed in each port to send the packet from the source to the destination. This table is defined when the networks are initialized, each packet sends the number of hops and its address to the neighbor. In Fig. 3.4 the algorithm is used to avoid faults in a 5x5 NoC. The faulty routers have the following coordinates: (1, 3) (2, 3) and (3, 3). For instance, if a core connected in router (2, 0) wants to send packets to the core that is connected in router (3, 4), the flit of the packets are sent to the following coordinates (2, 1) and (2, 2). When the flit reaches the router in the position (2, 2), the router sends the flit to the router (3, 2), according to the algorithm. The router (2, 3) cannot respond, because it is faulty, thus the router (2,2) selects the router in the position (3, 2), from then on the routing flows normally, because there are not faults in following routers.

The main drawback of Schonwald (2007) is that the table grows n^2, where n is the number of routers in the network, since each position in the table needs four more memory positions - stored in each router - to address the new neighbor. Besides, time is needed to form the routing table, that is, all routers send information to all routers to store the right number of hops in the table, however this table does not use a virtual channel so the implementation is easier.

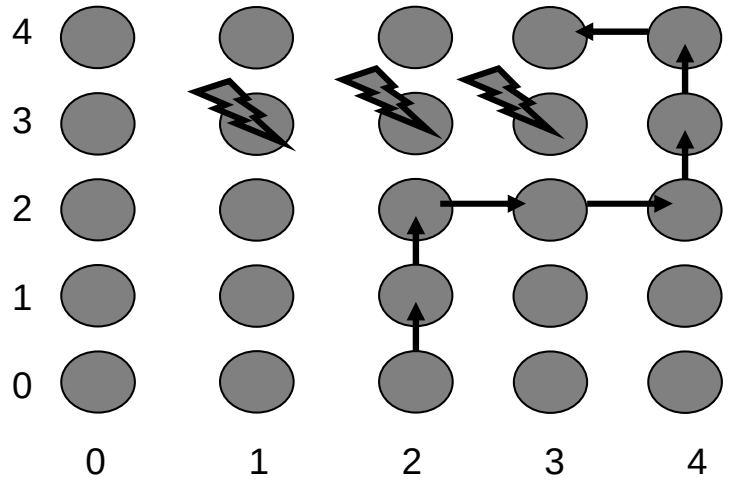


Figure 3.4: Faulty 5x5 Mesh with re-routing algorithm.

KOIBUCHI et al., (2008) propose a Default Backup Path (DBP), which is a unidirectional path that connects all routers in only one way like a circle. When the router has a fault or the link is faulty, the DBP is enabled and used to send and receive the packets to/from the neighbors. The DBP is connected to the network interface. In order to use the default backup path and connect the routers to the network, a virtual channel is needed to forbid some turns.

A set of wires of the DBP are used to send the packets in one direction and the other sets are used to send the packets in the opposite direction, although the use of a virtual channel makes the control of the NoC more complex and more prone to faults. One can observe that the DBP increases the power consumption in the links and the latency of the network, because when the DBP disables a router, the router components, such as FIFO, crossbar and arbiters are not used, only the link is used. So, by using the DBP path the packet takes more time being routed than in the original path.

Table 3.1 shows all related works with the techniques used and the network part that are protected. The first four related works protect the NoC adding more hardware, they change the architecture to be fault tolerant. The last three present the routing algorithm as a fault tolerant technique, in this case the routing algorithm isolates the fault component. In the table 3.1 one notices that Frantz et al. (2006) can protect the link and the router at design level. In algorithm level there are two related works that protect the router and the link, however Schönwald does not use a virtual channel, such as the strategy proposed in this Thesis.

Table 3.1: Related Works

Authors	Protection of the link	Protection of the router	Technique used	Adaptive Technique
YU and AMPADU	yes	no	Error Correction Code	Yes, parametrical ECC
ROSSI	yes	no	Error Correction Code	No
FRANTZ	yes	yes	Hamming Code in the router buffers. TMR in the links	No
NICOLAIDIS	no	yes	TMR, buffers	No
FRAZZETTA	yes	no	Routing Algorithm/ table	Yes
KOIBUCHI	yes	yes	Routing algorithm/ virtual channel	Yes
SCHONWALD	yes	yes	Routing algorithm/ table	Yes

The first four related works use redundancy techniques to protect the NoC against fault. They work at design level and use one technique to protect one part of the NoC, router or links or both. When comparing these techniques with the design level based strategy proposed in this thesis, our proposal does not use only one fault tolerant technique to protect the network. The proposal uses an adaptive scheme to isolate the faulty buffers and the fault column or row of the crossbar. TMR and Hamming Code are used to protect the FSMs and the links, respectively.

The last three related works use an adaptive routing algorithm to protect the NoC against fault. However all these related works use a routing table or a virtual channel to implement the adaptive algorithm. This increases the area and the power consumption of the network. Our proposal at algorithm level does not use a virtual channel or routing tables to implement the adaptive routing algorithm, we use the advantage of the NoC-Torus to re-route the packets through the fault-free paths.

Another aspect is related to fig 3.5, in this figure the related works are split according to the part of the network that is protected and the strategy level used. Since Frazzeta et al. (2008) has the option of using adaptive routing and part of the link, it uses two level strategies, algorithm and design level. In fig.3.5 we are able to see that

Frantz and Schönwald protect links and routers, Frantz at design level and Schönwald at algorithm level. These two works have the same aim and work at the same levels as the proposals of this thesis.

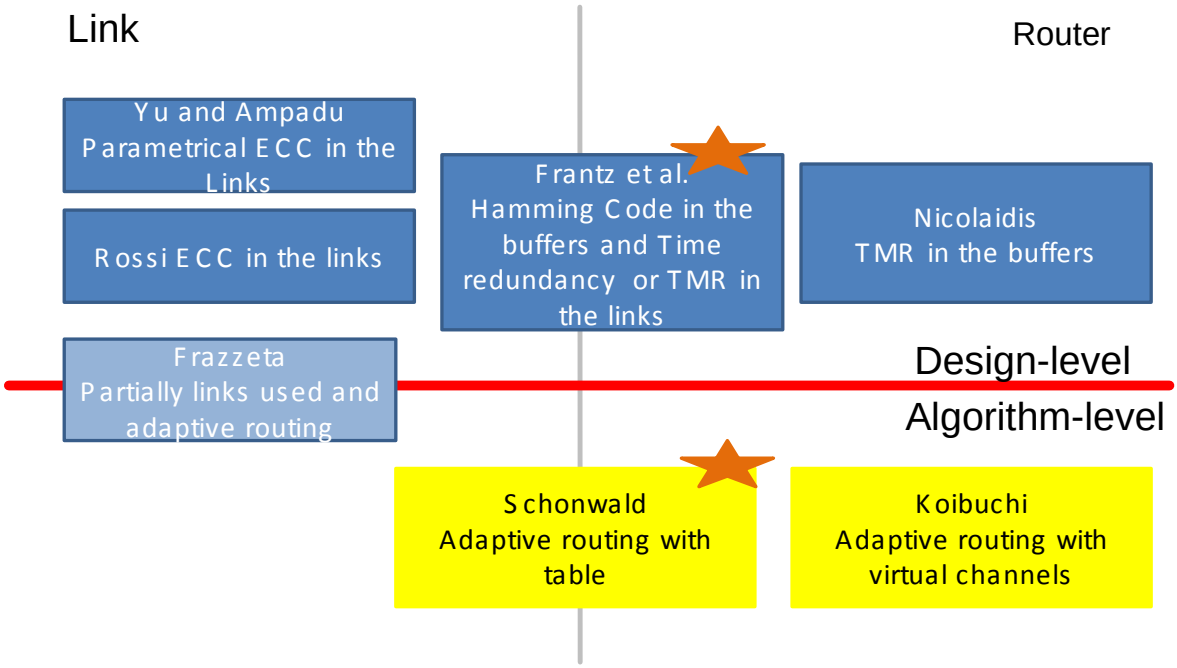


Figure 3.5: Related Works.

4 STRATEGY BASED ON DESIGN-LEVEL: ADAPTIVE ROUTER

The adaptive router proposed in this master thesis sustains performance and low power consumption. The design level strategy takes advantage of the fact that we can change the size of each buffer (number of slots) in accordance with the need of the application. From this idea, a fault-tolerant adaptive router is proposed to protect the buffers allowing further improvements in the performance of the system.

The underlying ideas of our fault-tolerant router lie in the fact that not all buffer slots have to work at the same data rate, i.e., some buffers have to endure larger traffic than others. This means that, in a NoC with the same buffer sizes for all channels, there is a set of buffers that are underutilized regarding the actual communication need.

The buffers used in the router channels are responsible for storing the incoming packets from the links. A faulty buffer slot sends wrong information to other routers. This chapter presents a method to use the buffer slots of other channels when a fault in a certain buffer slot is detected. We focus on providing a fault-tolerant adaptive router (FTAR) to dynamically adapt the utilization of buffer words in the channels in accordance with the faults detected. In addition, the router uses Hamming Code and TMR to protect the data wires and the FSM (FIFO, handshake and arbiter) of the router, respectively. One extra column and row was added to protect the crossbar of the router. With the proposed strategy it is possible to increase the yield and the system lifetime while at the same time ensuring the system performance and low power consumptions when compared to other techniques of fault tolerance. The approach used in this Thesis goes in the direction of having a minimal overhead in terms of area and power, while maintaining the performance level and improving the yield and reliability of the system. We present some adaptive mechanisms that can be used to increase reliability without power or performance penalties. The next sub-section presents some proposals about heterogeneous SoCs and also reasons to use heterogeneous routers.

4.1 Heterogeneous SoCs

System-on-Chips (SoCs) are emerging as one of the technologies providing a way to support the growing design complexity of embedded systems, since they provide processor architectures adapted to selected problem classes, allied to programming flexibility.

To ensure flexibility and performance, SoC combines several types of processor cores and data memory units of widely different size, leading to a very heterogeneous

architecture. The interconnection infrastructure must guarantee several contradictory goals, like high throughput for certain communication patterns, the possibility of a very fast partial routing update for very dynamic applications, reliability in critical applications, scalability, all with low power, low energy and low area overhead.

Azimi et al., (2007) affirm that it is necessary to find a way to keep the off-die bandwidth manageable in system architectures with trade-offs like cost, power and performance. Moreover, in a hardware context, the system must offer expressive flexibility with a scalable high-bandwidth, low-latency and power-efficiency to interconnect the cores allowing them to access memory and communicate with each other and with the rest of the system.

Manferdelli et al., (2008) assure that to guarantee the increase in performance of general purpose CPUs, one needs to use massive parallel computing. For this, more independent CPUs, bigger caches and more independent memory controllers have been used and it is possible to find many applications that use heterogeneous processors with several memory controllers to provide a large memory interface. One can find an example of this architecture on the Xbox360 (ANDREWS and BAKER, 2006). Fig.4.1 shows a system block diagram of the Xbox, a platform with several cores and each core having specific throughput and bandwidth. The Xbox 360 has one CPU with 3 cores and 1 Mbytes of L2 Cache. The CPU communicates directly with the L2 Cache at 10.8 Gbytes/s. The CPU has two memory controllers to communicate with an external memory, one BIU/IO Interface, a 3D accelerator core, a 10 Mbytes EDRAM memory and a Video output. The BIU/IO interface, in its turn, makes communication with the CPU, L2 Cache and with the IO/Chip and all these communications have different throughputs. Therefore, a router must consider the possibility of different configurations to guarantee the QoS to every core, as well as the right bandwidth and throughput, while at the same time achieving the smallest possible area overhead and power dissipation.

As another example of mixed communication behavior and requirement, there is a clear difference between cores in a SoC with out-of-order cores (OoCs) and in-order-cores (IoCs), as shown in the Fig. 4.2, (MANFERDELLI et al., 2008). OoCs are larger and have a worse power performance than IoCs. OoCs can be VLW processors which use a huge amount of dates to compute at once and the IoC can be a RISC processor which compute less data at a time and communicates more than OoCs. Besides, there is more communication among IoCs than OoCs, thus the former need to have different interconnection characteristics among them in order to guarantee a higher communication bandwidth among IoC devices. This is due to the fact that their communication with OoCs occurs in a much smaller scale.

From the SoC examples of fig. 4.1 and fig. 4.2, one can see that each NoC has a different communication pattern. In a NoC several items can vary from design to design, like buffer depth of FIFOs, router topology, switch and arbiter (VESTIAS and NETO, 2006). In this manner, decisions such as throughput, latency and bandwidth are defined as a modification of the NoC architecture, most of the time being made at design time, trying to guarantee the performance of the system. However, whenever the product needs an update or has to change its functionality, most likely a huge change in the communication pattern will be observed and hence decisions performed at design time would mean either a loss in performance, or excessive power dissipation. Today, cell-phones are a good example of this continuous process of updating to integrate new

functions. The decision made at design time cannot guarantee the same performance for these new services.

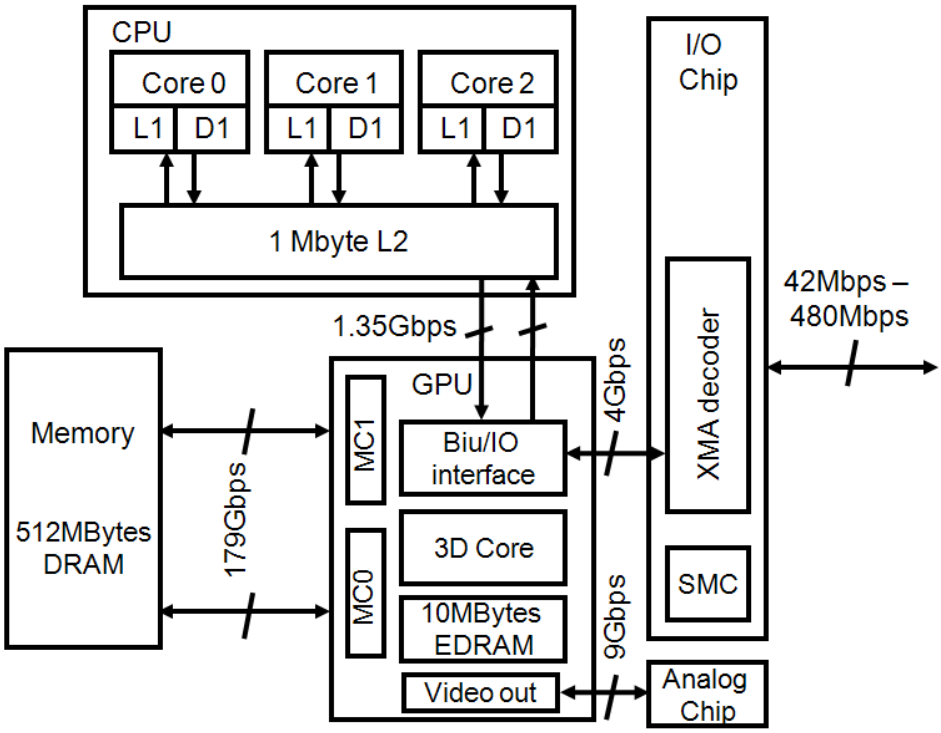


Figure 4.1: Xbox 360 block diagram (ANDREWS and BAKER, 2006).

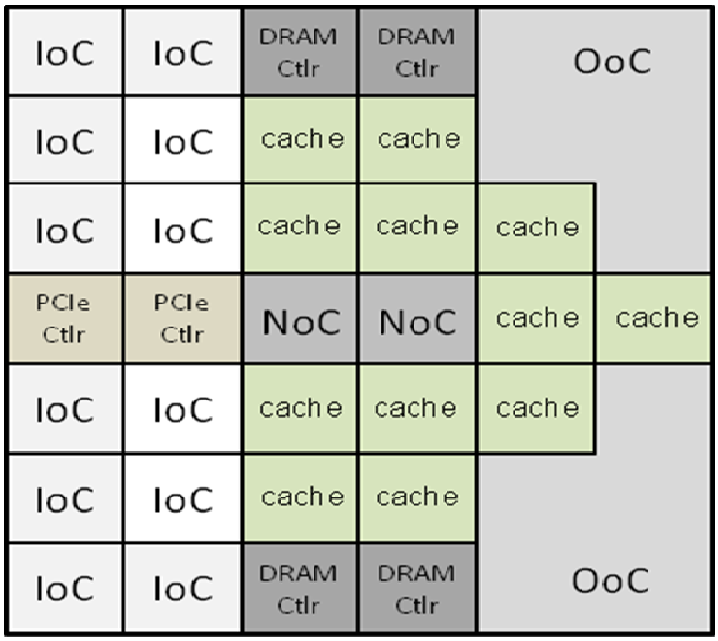


Figure 4.2: Example of different cores used in a SoC (MANFERDELLI et al., 2008).

4.2 A Quantitative Analysis of the Problem

Four examples of real applications were simulated to analyze the behavior of the router. The applications used were the MPEG4 (Motion Picture Experts Group 4), VOPD (Video Object Plane Decoder) (BERTOZZI, D. et al., 2005), MWD (Multi-Window Display) (SRINIVASAN and CHATHA, 2006) and Xbox (ANDREWS and BAKER, 2006). All have 12 cores, but with different communication patterns, as represented in each link bandwidth.

A traffic simulator in Java was utilized to evaluate the network hotspots and the average latency using the adaptive and original routers. The distribution of the cores, in the NoC, was specified in accordance with the communication needs of the cores, reflecting a design time choice based on the original application.

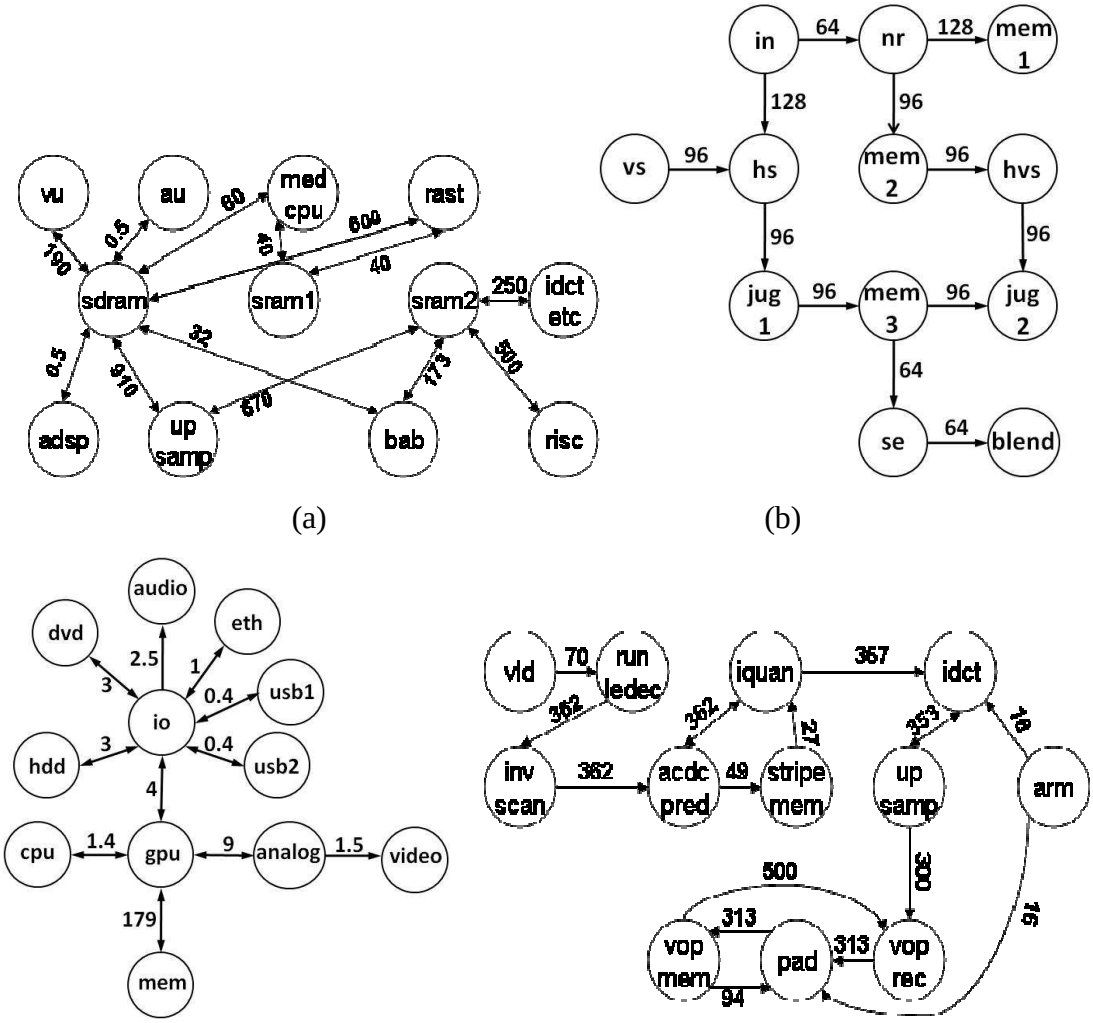


Figure 4.3: (a) MPEG4; (b) MWD; (c) Xbox; (d) VOPD task graphs.

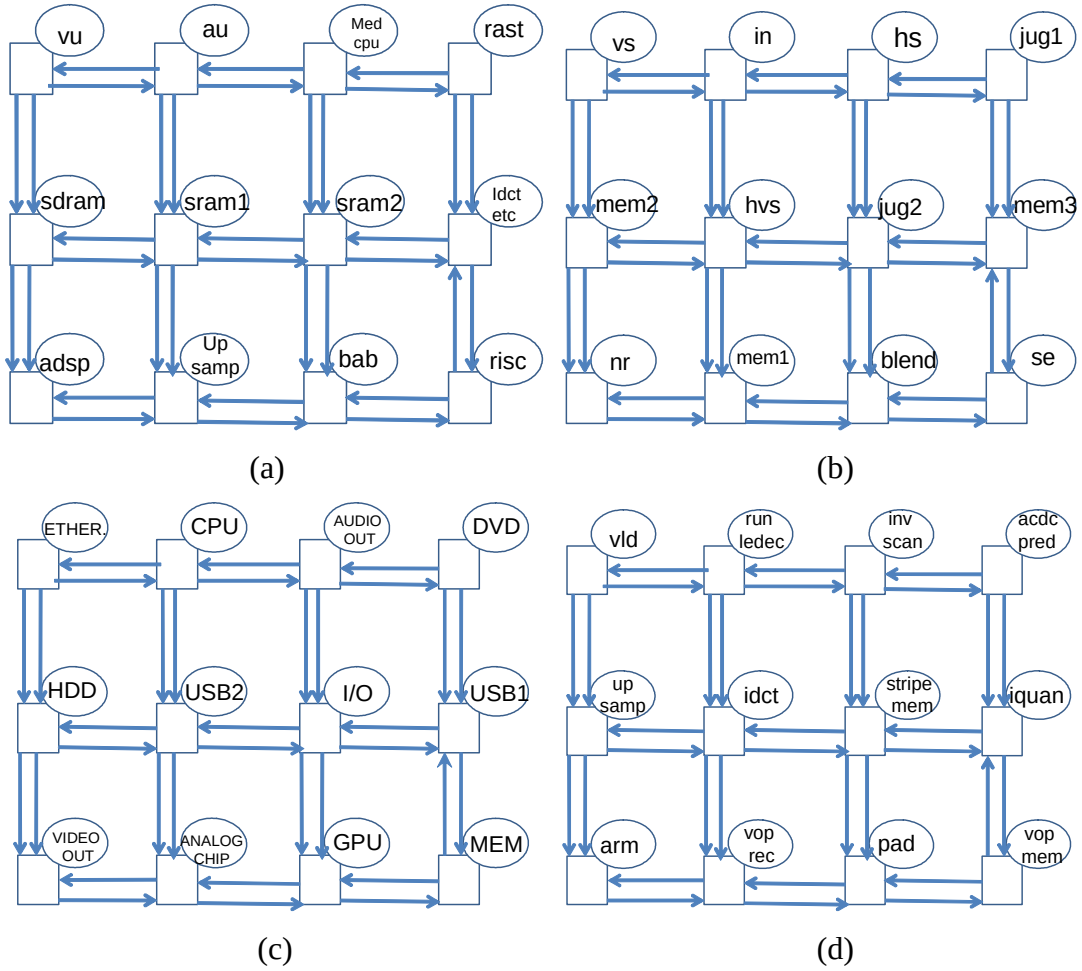


Figure 4.4:4x3 NoC for (a) MPEG4; (b) MWD; (c) Xbox; (d) VOPD.

Fig. 4.5 shows the mean efficiency of MPEG4, VOPD, MWD and Xbox to the core graph of fig. 4.3 mapped to a NoC 4x3 with homogeneous buffer size. To compute the efficiency of each router in the network, first each core of the MPEG4, VOPD, MWD and Xbox were mapped to a specific router in the NoC according to figure 4.4. This mapping is used for all experiments shown in this thesis.

In this thesis, efficiency represents how many of buffer slots are being used appropriately, in accordance with the necessity of the application. Fig. 4.5 represents the efficiency in accordance with the equation 4.1. This equation indicates the number of the buffer slots really used per number of buffers existing in each channel of the NoC. This proves that homogeneous routers use excessive buffers to some channels since not all channels present the same communication rate. In such cases, the extra buffers of the channel consume area and power unnecessarily.

$$\eta = \frac{\sum_{i=1}^{i=\#routers} \frac{\#buffers\_slots\_used\_router_i}{\#total\_buffers\_slots\_router_i}}{\#routers} \qquad (4.1)$$

The overall idea is to show how much of the buffer slot is used, assuming the size was chosen based on the maximum performance.

The buffer is sized for the highest size to guarantee that all channels in a router will have low latency, that is, each router will have the highest amount of flip-flops to provide performance for that specific link. For instance, in a *routerX* it was noticed that at design time one channel of the *routerX* needs 8 buffer slots, so all channels of the *routerX* will have 8 buffer slots to guarantee performance. On the other hand, in a *routerX+1* the maximum buffer slots needed is 2 and so all channels will have buffer slots equal to 2 in *routerX+1*. However, in this case, if the application is changed, probably the latency and the power consumption increase, since in some link there might not be enough buffers to ensure QoS. In Zeferino and Susin (2003), as the buffer depth increases the latency decreases, because the packet will be stored in the buffers and then will use less links when they are blocked. However, as the buffer depth increases the cost of the router increases too.

One can see, in fig. 4.5 that in a buffer sized to the best performance case, around 54% of the buffer slots are utilized and all the others are not. However, they are consuming power, but they are not contributing to reducing the latency or the number of hotspots in the network. These results were presented in VLSI-SoC (Matos et al.,2009) too.

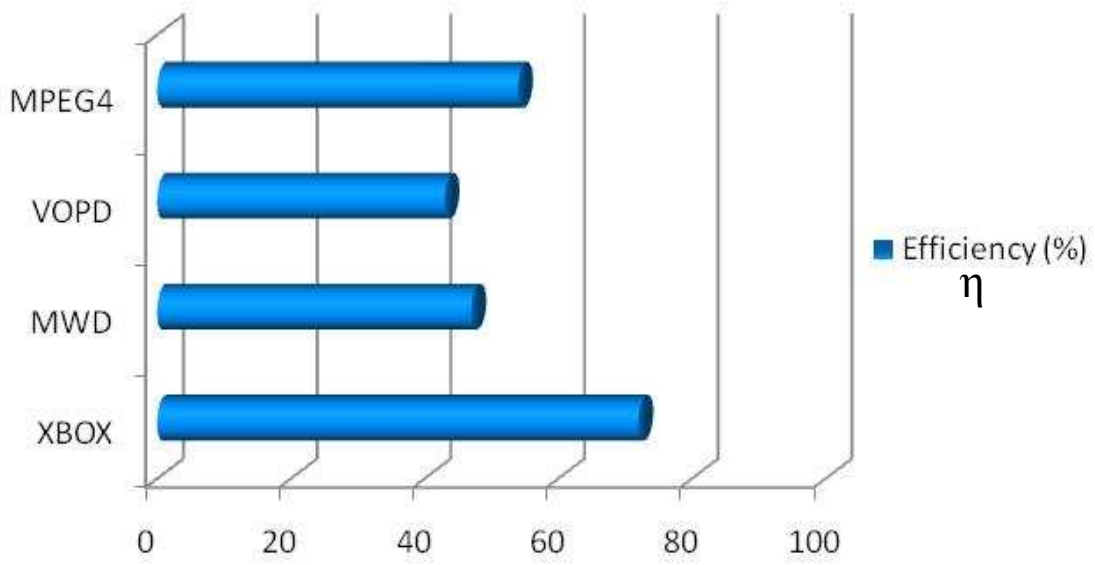


Figure 4.5: Efficiency of a homogeneous router.

As one can notice, the SoC has a dynamic nature, in this case the fault tolerance technique for SoC needs to be dynamic too. The fault tolerant technique needs to be adaptive, to adapt to changes in the systems and to adapt to failures that will arise. The following sections present the proposed techniques, where the solutions are adaptive. The solutions adapt according to the requirement of the system, or according to the fault location, without the need for oversized features or re-design to avoid the faults.

4.3 Adaptive Router Architecture

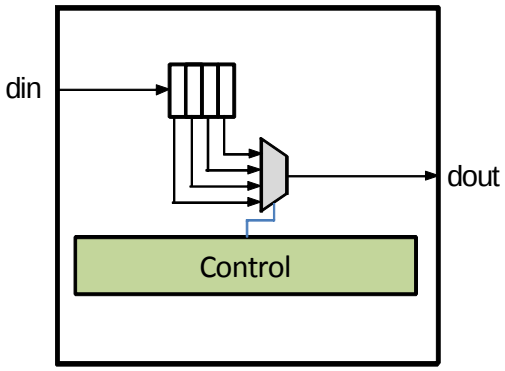
If a NoC router has a larger buffer, the throughput will be larger and the latency in the network smaller, since it will have fewer flits stagnant on the network (WU and CHI, 2005). Nevertheless, there is a limit in the increase of the buffer depth. Since each communication will have its peculiarities, sizing the buffers for the worst case communication scenario will compromise not only the routing area, but power as well. However, if the router has a small buffer depth, the latency will be larger and the QoS may be compromised. The proposed solution is to have a heterogeneous router, in which each channel can have a different buffer size. In this situation, if a channel has a communication rate smaller than its neighbor, it may lend some of its buffer units that are not being used. In a different communication pattern, the roles may be inverted or changed at runtime, without a redesign step.

In the adaptive router it is possible to dynamically reconfigure different buffer depths for each channel. A channel can lend part or the whole of its buffers in accordance with the requirements of the neighbor buffers. To reduce connection costs, each channel may only use the available buffer slots of its right and left neighbor channels. In such a case, each channel may have up to 3 times more buffer slots than its original buffer size planned at design time. When a channel fills its entire buffer it can borrow more buffer words from its neighbors. In this manner, some signals of each channel must be sent to the neighboring channels in order to control its stored flits. The information about how many units of the buffer are used for each channel is sent by an external control and this information can be dynamically altered outside the router on the fly.

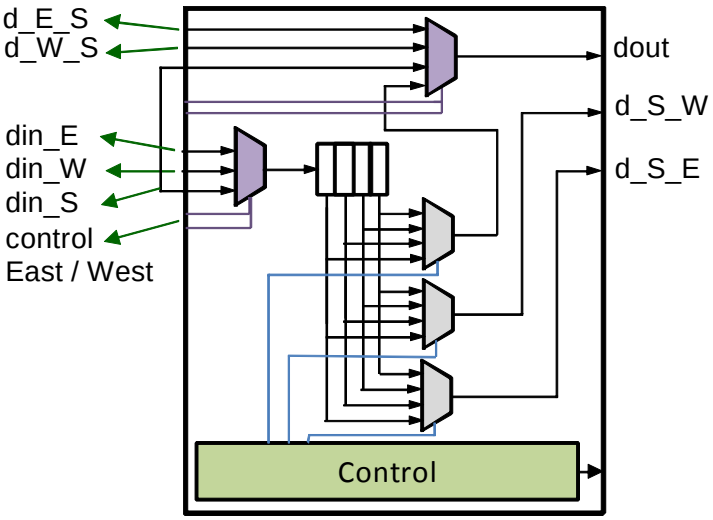
Fig. 4.6(a) shows the original and proposed input FIFO which contains 4 buffer slots. Comparing the two architectures, the new proposal uses more multiplexers to allow the reconfiguration. Fig. 4.7 is the new architecture and it shows the channel of fig. 4.6(b). Fig. 4.6 (b) presents the *South Channel* as an example. In this architecture it is possible to dynamically configure different buffer depths for channels. This architecture was present in the ARC workshop (Concatto et al., 2009).

Each channel can receive three data inputs. Let us consider the *South Channel* as an example, having the following inputs: the own input (*din_S*), the right neighbor input (*din_E*) and the left neighbor input (*din_W*). For illustration purposes, let us assume we are using a router with FIFO depth equal to 4 and there is a router that needs to be configured as follows: *South Channel* with buffer depth equal to 9, *East Channel* with buffer depth equal to 2, *West Channel* with buffer depth equal to 1 and *North Channel* with FIFO depth equal to 4. In this case, the *South Channel* needs to borrow buffer units from its neighbors. As the *East Channel* occupies 2 of 4 units, this channel can lend 2 units to its neighbor, but even then, the *South Channel* still needs 3 more buffer units. As the *West Channel* occupies only 1 unit, the 3 missing units can be lent to the *South Channel*.

When the *South Channel* has a flit stored in the *East Channel* and this flit must be sent to the output, it is passed from the *East Channel* to the *South Channel* (*d_E_S*) and so it is directly sent to the output of the *South Channel* (*dout_S)* by a multiplexer. The *South Channel* has the following outputs: the own output (*dout_S*) and two more outputs (*d_S_E* and *d_S_W*) to send the flits stored in its channel but belonging to neighbor channels.



(a)



(b)

Figure 4.6: Input FIFO (a) Original; (b) proposed.

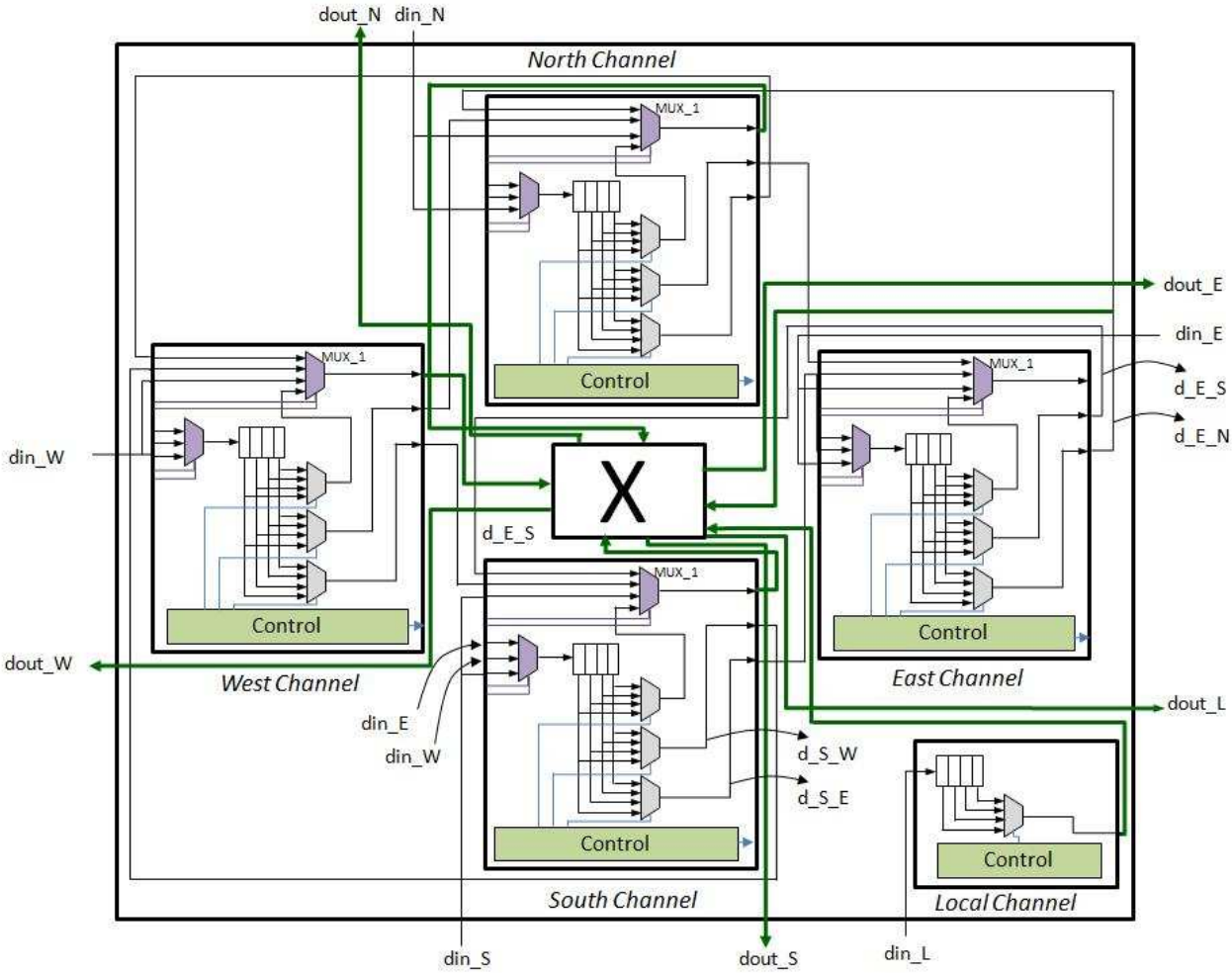


Figure 4.7: Proposed router architecture.

Some rules were defined to enable the use of buffers from one channel to other channels. To reduce connections costs, each channel may only use the available buffer words of its right and left neighbor channels. When a channel fills all its buffers it can borrow more buffer words from its neighbors. First the channel asks for buffer words to the right neighbor and if it still needs more buffers, it tries to borrow from the left neighbor.

In this manner, some signals of each channel must be sent for the neighboring channels in order to control its stored flits. The information about how many units of the buffer are used for each channel is sent by an external control (it is information received by an input pin of the router) and this information can be dynamically altered outside the router.

In the proposed router architecture each channel knows how many of its own buffers are being used in the channel and how many are being borrowed from neighbors. Each channel controls its storage flits, being those units stored on its own buffer or in the neighbor channel buffers. In this design we are not considering the possibility of the *Local Channel* using neighbor buffers, only the *South, North, West and East Channel* of a router can make use of their adjacent neighbors.

The link between the router and the core, the *Local Channel,* is not shared in the network, rather it is a dedicated link between the core and the router. Therefore it was decided to not include the loan process in the *Local channel*.

In order to reduce routing and extra multiplexers, we adopted the strategy of changing the control part of each channel. Hence, the borrow/lent process is coordinated only by simple finite state machines and pointer management for each FIFO storage unit. This work was presented in the ISVLSI Congress (Matos et al., 2009).

4.4 Performance Results of Adaptive Router Architecture

This section shows the results of performance of the adaptive router, without any fault tolerance. In this chapter we prove that the adaptive router decreases the latency of the router in almost 80%.

In order to define the buffer size to each channel in accordance with its need, we used a simple decision mechanism described by the Algorithm 4.1. This algorithm distributes buffer slots among the channels of each router. The algorithm 4.1 is very simple to implement, hence the reason it was chosen. This means that if a better algorithm were implemented the performance would be better. The algorithm 4.1 is implemented in Java by a cycle accurate simulator together with adaptive router shown in section 4.3.

The applications (MPEG4, MWD, XBox, VOPD) used to observe the performance of the NoC were emulated. According to the mapping of fig. 4.4 and the core graph of fig. 4.3, the equation 4.2 was computed.

$$num\_cycles\_per\_packet = \frac{freq}{\frac{bandwidth}{packet}} \quad (4.2)$$

By explanation, *num_cycles_per_packets* is the period that each core must send its packets, *freq* is network frequency, the *bandwidth* is given by the core graph and the *packet is* given by the application. The *num_cycles_per_packets* is given to the cycle accurate simulator to analyze the latency and throughput of the network.

The simulator analyzes each channel individually and performs the buffer words distribution according to the number of hotspots. First the algorithm verifies the hotspots of each router. We consider hotspots those channels that receive a large number of flits and in such cases, require many buffers words because of contingency reasons. This way, whenever hotspots are detected, the algorithm verifies the possibility of borrowing the neighbor's channel buffers units. When there is only one hotspot in the router, the buffer loaning process occurs between the right and left neighbors. Otherwise, if there are two hotspots in the router, the neighbor's buffer words are divided between the hotspots. For these experiments, fixed-length packets of 80 flits were assumed and the link size was defined to be 16 bits. From these decisions, with the information of bandwidth presented in fig. 4.8 and with a defined frequency, we determined the interval in number of cycles that each packet is sent to the link. The bandwidth to each link considers the need of all cores that utilize it.

When a neighbor channel is used and it is not a hotspot, the channel will leave only one buffer word to the neighbor channel and will use the remaining buffer words. When there are three hotspots in a router, no buffer slots can be lent. The constant *buffer_increase* considered in the pseudocode refers to the channel buffer depth.

Algorithm 3.1. Pseudocode to use the buffers of the router channels

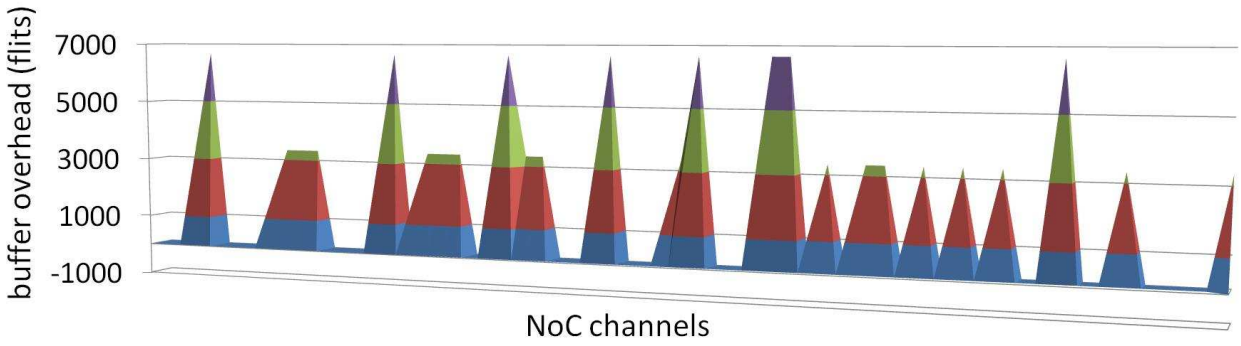
```
1  FOR router=0 to number_of_router DO
2     FOR channel=0 to 3 DO
3        IF channel is hotspots THEN
4           IF number_hotspots_router = 1 THEN
5              IF buffer_right_neighbor is not used THEN
6                 buffer_channel = buffer_channel + buffer_increase;
7                 buffer_right_neigbord = buffer_right_neigbord – buffer_increase;
8              ELSE
9                 buffer_channel = buffer_channel + (buffer_increase-1);
10                buffer_right_neigbord = buffer_right_neigbord – (buffer_increase-1);
11             END IF
12             IF buffer_left_neighbor is not used THEN
13                buffer_channel = buffer_channel + buffer_increase;
14                buffer_left_neigbord = buffer_left_neigbord – buffer_increase;
15             ELSE
16                buffer_channel = buffer_channel + (buffer_increase-1);
17                buffer_left_neigbord = buffer_left_neigbord – (buffer_increase-1);
18             END IF
19
20          ELSE IF number_hotspots_router = 2 THEN
21             IF buffer_right_neighbor is not used THEN
22                buffer_channel = buffer_channel + buffer_increase;
23                buffer_right_neigbord = buffer_right_neigbord – buffer_increase;
24             ELSE IF right_neighbor_channel is not hotspots THEN
25                buffer_channel = buffer_channel + (buffer_increase-1);
26                buffer_right_neigbord = buffer_right_neigbord – (buffer_increase-1);
27             ELSE IF buffer_left_neighbor is not used THEN
28                buffer_channel = buffer_channel + buffer_increase;
29                buffer_left_neigbord = buffer_left_neigbord – buffer_increase;
30             ELSE
31                buffer_channel = buffer_channel + (buffer_increase-1);
32                buffer_left_neigbord = buffer_left_neigbord – (buffer_increase-1);
33             END IF
34          END IF
35    END IF
36    END FOR
```

Using the algorithm 4.1 for buffer slots lending, we obtained the results of fig. 4.8, which presents the behavior of the NoC for the VOPD application using a 4x3 NoC (fig. 4.4(d)) with buffer depth equal to 4. The X axis presents the input channels of all 12 routers of the NoC and the Y axis represents the number of flits that need to wait for the availability of the buffer before being sent to the next router.

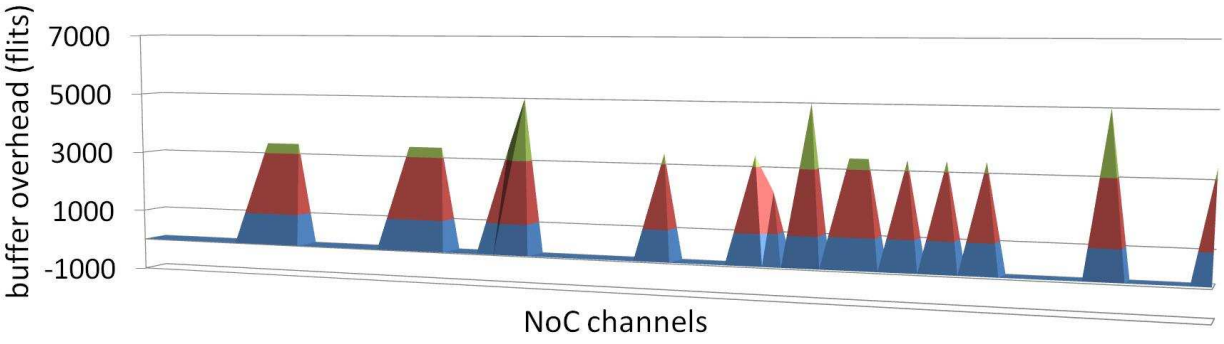
Equation 4.3 shows how the *buffer_overhead* is calculated, where *flits_to_send* refers to all flits that should be sent to a channel, according to a bandwidth required for a router with an infinitive buffer depth. The *buffer_depth* means the real buffer size of the channel.

$$buffer\_overhead = flits\_to\_send - buffer\_depth \quad (4.3)$$

Fig. 4.8(a) depicts the results obtained with homogeneous routers and fig. 4.8(b) presents the reduction of hotspots obtained with the adaptive router. One can observe that with the adaptive router the number of hotspots was drastically reduced. In fig. 4.8(a) there are 7 peaks over 3000 flits. In fig. 4.8(b) there are 3 peaks and none is higher than 5000 flits. This means that the adaptive router has 2.33 times less peaks than the original router.



(a)



(b)

Figure 4.8: Number of flits overhead per router (a) in the original architecture (b) in the new router for VOPD

We analyzed four applications (MPEG4, MWD, VOPD and XBOX) that present different traffic conditions (for the mapping presented in fig. 4.4). We fixed the buffer size of the adaptive router to 4 and sized the buffer of the original router in order to have the same latency as the adaptive ones. In this case, to reach the same average latency obtained with the adaptive router, the homogeneous router needed much larger buffers, as it can be seen in fig. 4.9.

In fig. 4.9 each application has two columns, the first column is the adaptive router with buffer depth equal to 4 and the second is the buffer size of a router fixed at design time required to reach the same latency performance. Observing the results presented in fig. 4.9, we verify that the buffer depth greatly influences the average latency. As these applications present different bandwidths in the links and a different number of connections among the cores, we can confirm that to have the same average latency obtained with the adaptive router, larger and for many cases useless buffer depths were required by the homogeneous router.

In fig. 4.9 we observed that the MWD application presented the smaller latency results to the adaptive router and thus to obtain the same latency value, the homogeneous router required a buffer with 11 positions. This can be explained due to the traffic behavior of the MWD application, since there are few connections among the cores. In this case, as there are many buffer units which are not used, they could be lent to the channels in use. This experiment proves that it is possible to use a single NoC with the adaptive router to obtain low latency results for any application.

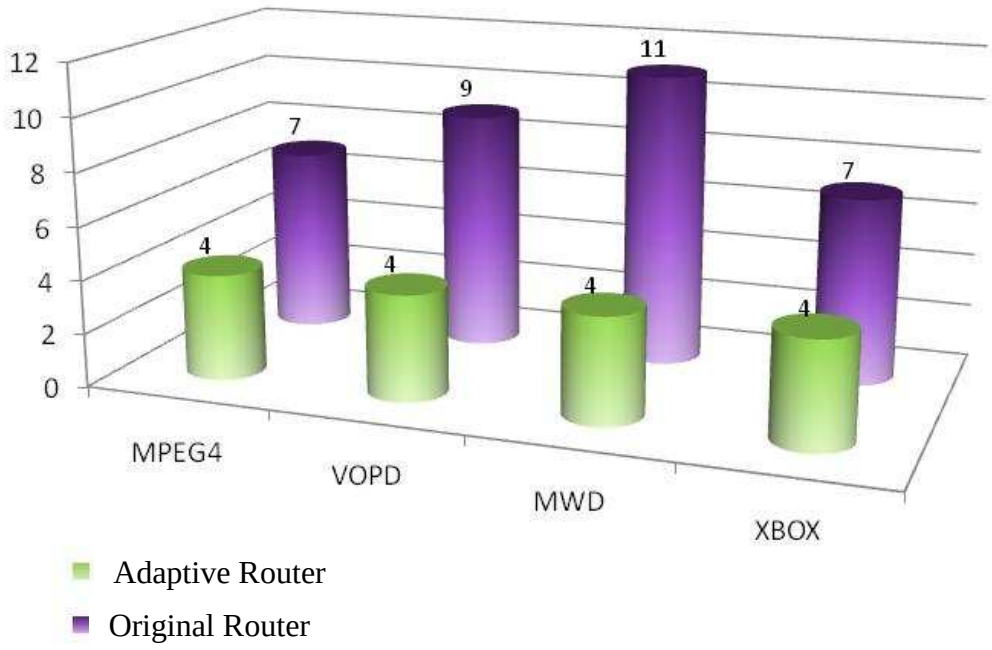
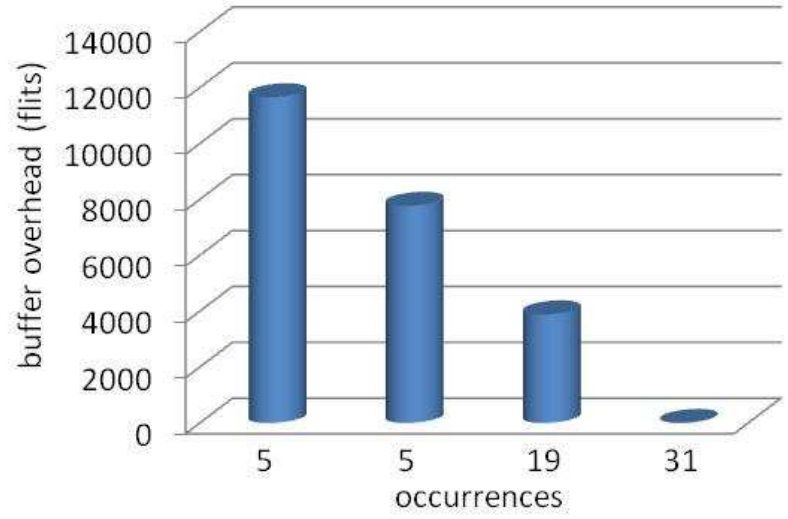


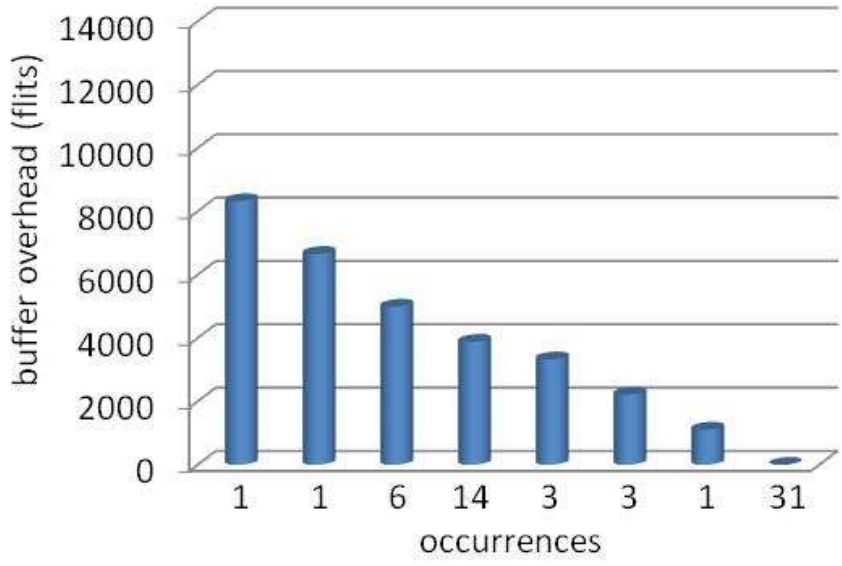
Figure 4.9: Four applications with buffer depth of the original router sized in order to obtain the same latency of an adaptive router with buffer depth equal to 4.

With the adaptive router we also verified that there was a better distribution of flits in the network. Fig. 4.10 shows the number of flits that need to wait for the availability of the buffer to be sent to the next router. We called this situation buffer overhead and the high values of buffer overhead represent the hotspots, i.e., channels that need a larger buffer depth.

In the X axis of fig. 4.10 one can see how many channels have approximately the same number of buffer overheads (flits that could not be stored in the first time). Hence, fig. 4.10 shows the number of channels that generate the network congestion, for the original router in fig 4.10(a) and for the adaptive one in fig. 4.10(b). One can observe that for the original router there are 10 channels (first two columns in fig. 4.10(a)) that potentially decrease the performance of the NOC. However, for the proposed router one has only 2 channels that decrease the performance and with a much smaller buffer overhead value, since the buffer lending/borrowing process better distributes the storage of flits in the NoC. This experiment shows that for the same buffer size, the adaptive router uses the buffer slots more efficiently and hence less power is wasted.



(a)



(b)

Figure 4.10: Occurrences of flits that need to wait the availability of the buffer to be sent to the next router to MPEG4 application for (a) original and (b) adaptive routers.

4.5 Fault Tolerant Adaptive Router (FTAR)

When a fault is detected in a set of buffer units in a FIFO, the FIFO is not totally useless. Only the buffer unit, to which the faulty flip-flops belong, must be discarded. This buffer unit is isolated and the next buffer word in the FIFO is used. Nevertheless, when all fault-free buffer units of the channel are being used by the application (due to a certain traffic pattern, for example), the faulty buffers need to be replaced. In this case, the buffers of a neighbor channel are used to substitute the defective buffer unit. As the cores connected to the NoC present different communication requirements, not all channels need to use all their buffers, in their full size, all the time. Of course, for some buffers there will be a tradeoff between performance and fault tolerance. For example, when all units of the buffer in the FIFO are faulty in a single channel, then at least one buffer slot must be borrowed from a neighbor. A channel can borrow more than one buffer unit from its right and left neighbor channel.

Fig. 4.11(c) shows the fault tolerant strategy proposed for the buffer slots to isolate a fault in a flip-flop. Fig 4.11(a) shows a buffer with all slots fault-free, but fig. 4.11(b) presents the FF3 faulty, in this case a router without a fault tolerant technique will lose this buffer and consequently the channel. Fig.4.11(c) shows the fault tolerant technique proposed for buffers. The FF3 is isolated and the fault free buffer slots continue to be used. All channels in the router have the ability to avoid the fault in their buffer slots. The *Local Channel* does not have the ability to lend or to borrow slots from the neighbors.

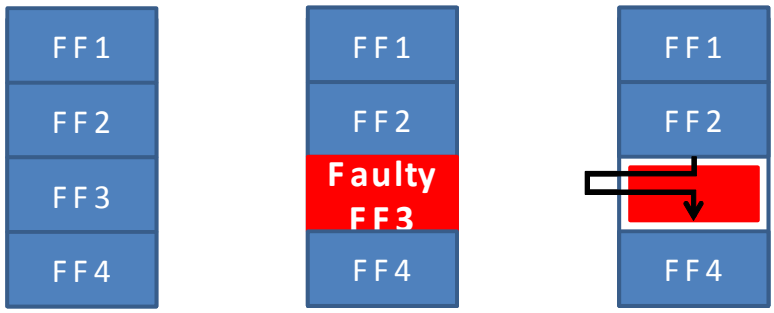
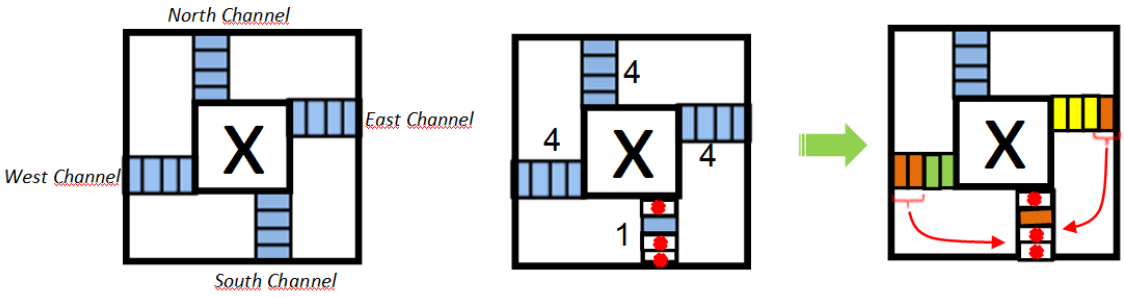


Figure 4.11: Buffer slots (a) fault free; (b) faulty; (c) isolating a fault.

Fig. 4.13 shows the adaptive (fig. 4.13(a)) and the fault tolerant adaptive buffer (fig. 4.13(b)). Fig. 4.13(b) shows that there is a *faulty_buffer* signal that informs the buffer control which is faulty. According to the *faulty_buffer* signal, the buffer slot can be used or not. If the flip-flop is faulty the control bypasses and uses the next flip-flop. Only if needed, to sustain performance, buffer slots are borrowed, following algorithm 4.1.

As in the adaptive router without fault tolerance, the FTAR must send some signals to the neighboring channels in order to control the stored flits. Each channel receives three data inputs. Let us consider the *South Channel* as an example, having the following inputs: the own input (*din_S*), the right neighbor input (*din_E*) and the left neighbor input (*din_W*), as shown in Fig. 4.13(b). For illustration purposes, let us assume we are using a router with buffer depth equal to 4 and the *South Channel* has three faulty buffer

units, as depicted in fig. 4.12(b). In such a case, the *South Channel* needs to borrow one buffer unit from its neighbor channel. Let us consider that *East Channel* can lend part of its buffer, then, as the fault in *South Channel* is signalized, the flit is sent to data input of the *East Channel* and it is stored in these buffers units. When the flit belonging to Sou*th Channel* stored in the *East Channel* is required, it must be sent to the output, passing from the *East Channel* to the *South Channel* (*d_E_S*) and so it is directly sent to the output of the *South Channel* (*dout_S*) by a multiplexer. The *South Channel* has the following outputs: its own output (*dout_S*) and two more outputs (*d_S_E and d_S_W*) to send the flits stored in its channel but belonging to neighbor channels. Firstly, in the occurrence of faults, the channel borrows a buffer from the right adjacent channel. In this design we are not considering the possibility of the *Local Channel* using neighbor buffers, but the *Local Channel* has the ability to isolate the faulty buffers.



(a) (b) (c)

Figure 4.12: (a) router designed with buffer depth 4; (b) an example of faulty buffer; (c) reconfiguration of the buffers to meet the faulty buffer demand.

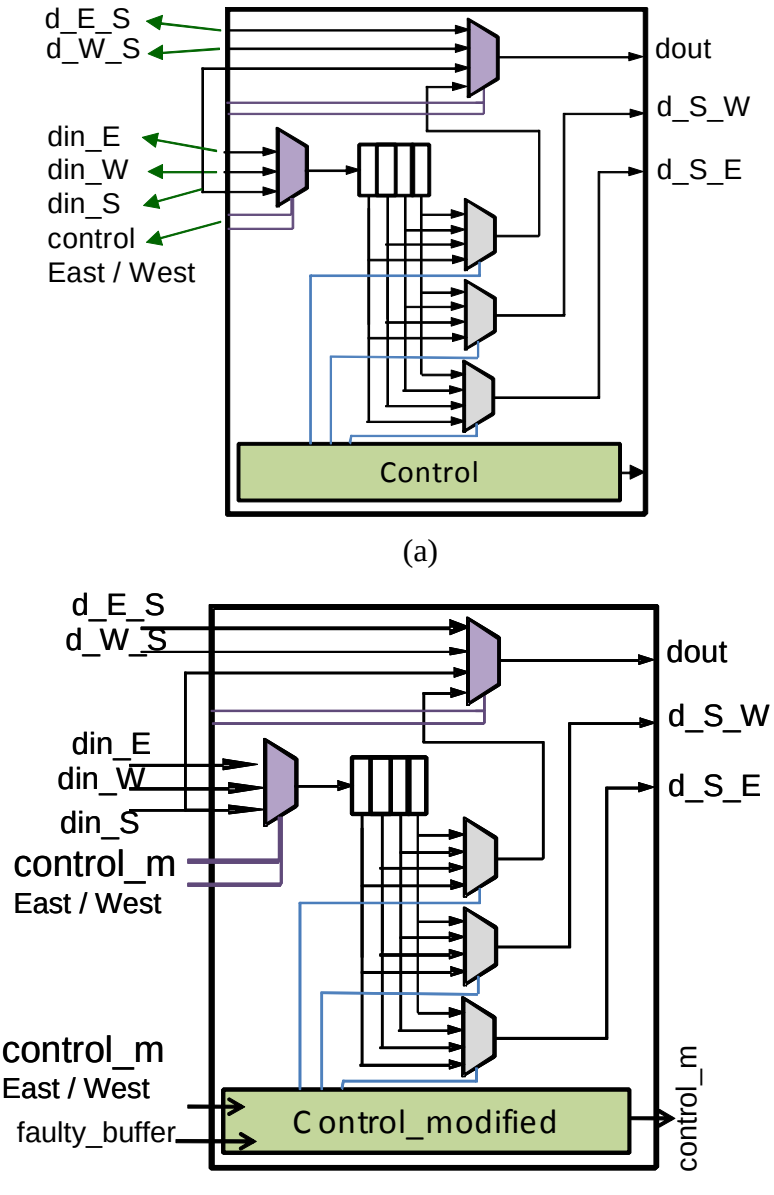


Figure 4.13: Input FIFO (a) adaptive; (b) fault tolerant adaptive.

4.6Fault Tolerance Solutions

4.6.1 Links

The idea of the strategy based on design-level is to combine four fault tolerance methods to tolerate permanent faults, each one targeting a different part of the circuit to allow the smallest possible area and performance overhead. To obtain a fault tolerant NoC, we use the adaptive router as a solution to faulty buffer units, since wasting all buffer slots, in terms of cost, is unacceptable. At the link level, any defect in the metal contacts could compromise a wire and must also be mitigated and hence hamming code is used to protect the links, as shown in fig. 4.14. Hamming encoding occurs also in the network interface and the links are decoded before entering the buffer of the FIFO. For a channel with 34 bits (32 data + 2 control), 6 bits are needed to include the hamming code.

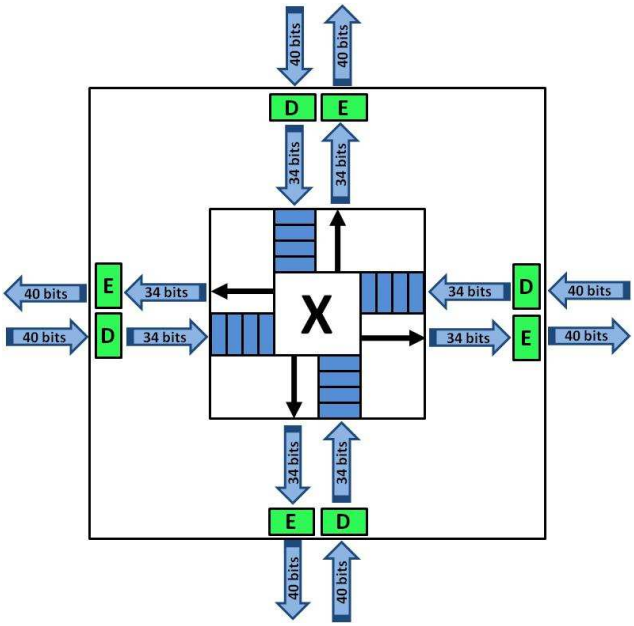


Figure 4.14: Hamming code in the NoC links. E – encoding code, D – decoding code as proposed by Frantz et al, 2006

4.6.2 Switch Matrix

One column and one row were added to protect the crossbar of the router. When a fault occurs in the crossbar it can be isolated and the extra column and/or row are used. Fig. 4.15 shows one example where the *West Channel* wants to send packets to the *North Channel*. For instance, if the row used by the West Channel in the crossbar is faulty, the multiplexer in the input of the crossbar is activate to use the extra row, then the flit is sent to the extra row, isolating the fault. As only one extra row and column were added, each switch matrix can tolerate one fault in the crossbar.

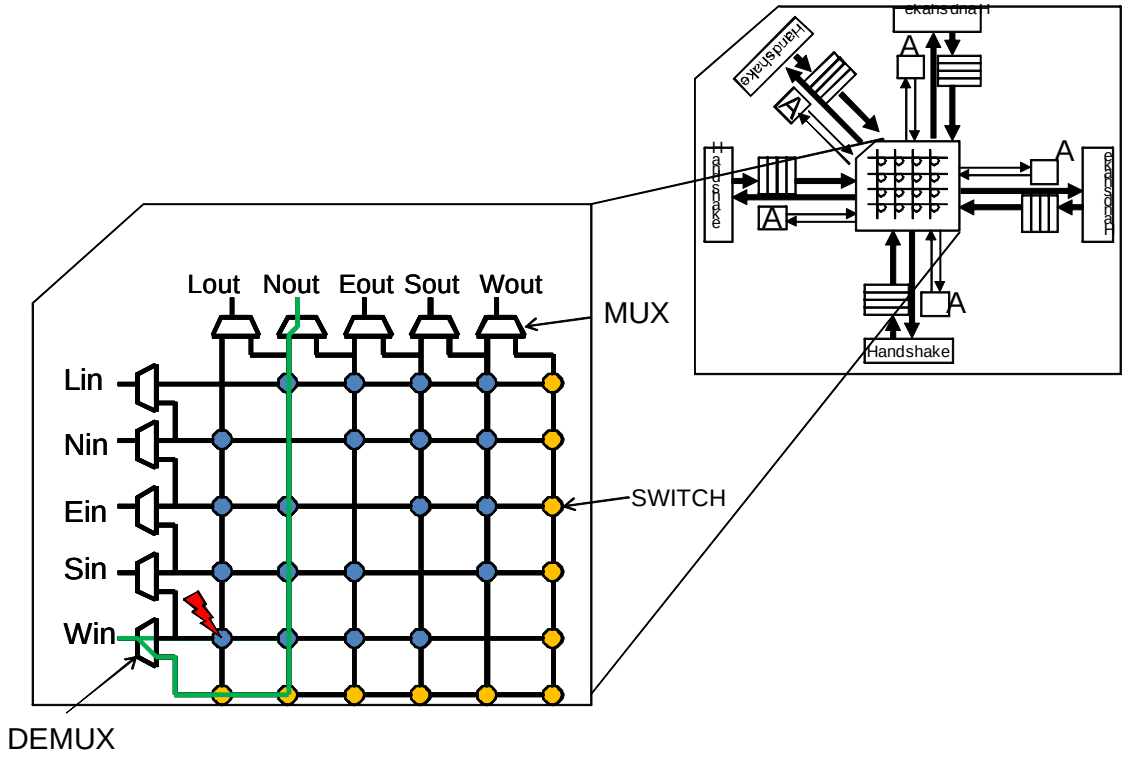


Figure 4.15: RASoC switch matrix.

4.6.3 Finite States Machines

The TMR strategy was used to tolerate fault in the FSMs of the router. There is not redundancy in the FSM or it is very difficult to find these redundancies. The easiest way to protect FSMs is TMR. For that reason were used TMR in all FSMs.

Most of the FSMs in the router architecture are simple and consequently small. For example, the FSM that controls the FIFO is very small, it has only three states. We have chosen TMR to protect the control of the FIFO, since the TMR scheme triplicates the Finite State Machine.

For the FSM in the FIFO we added 3 registers (2 bits) and 1 voter to choose the right output of FSM. All these methods have the goal of ensuring the integrity and reliability of the data that passes through the NoC in the presence of permanent faults. The TMR was also used for the handshake protocol and the arbiter, but as shown in the results, the cost is lower than for using TMR for all the router.

4.7 Performance and Synthesis Results

4.7.1 FTAR Evaluation

A traffic simulator in Java was utilized to evaluate the average latency using a homogeneous, adaptive, fault tolerant adaptive routers. The distribution of the cores in the NoC was specified in accordance with the communication needs of the cores, reflecting a design time choice, being based on the original application. For our experiments, a 4x3 NoC was used.

To simulate the FTAR we have implemented a very simple buffer distribution algorithm in Java as shown in the pseudocode of algorithm 4.1. In the presence of faulty buffers, the algorithm bypasses the faulty buffer and, in accordance with the need of the application, more buffer units can be obtained from neighbor buffers.

Firstly, the algorithm verifies which are the channels with faulty buffer words and, in accordance with *number_faulty_buffers* (which has the number of defective buffer words in the channel), decrements the depth of these buffers. The faults are diagnosed off-line and are not in the scope of this work. Subsequently the pseudocode of the algorithm 4.1 is implemented considering now only the non-defective buffers. This code verifies the total number of flits that passes over each channel. If the channel presents the number of flits above a certain threshold, it is considered one hotspot. The threshold is considered as the mean value between the two extremes of the total number of flits that cross the NoC channels. In this way, when hotspots are detected, the algorithm verifies the possibility to borrow the neighbor's channel buffers. When there is only one hotspot in the router, the loan of buffers occurs between the right and left neighbors. Otherwise, if there are two hotspots in the router, the neighbor buffers are divided between the hotspots. When a neighbor channel is used and it is not a hotspot, the channel will leave only one buffer to the neighbor channel and will use the remaining buffers. When there are three hotspots in a router, no buffer can be lent. The constant *buffer_increase* considered in the pseudocode refers to the channel buffer depth. The algorithm 4.1 does not consider the *Local Channel*.

In order to verify the use of the fault tolerant adaptive router we performed some simulations by randomly assigning faults in the buffers. Firstly we verified the possibility of the occurrence of defects in a router connected to a processor and a cache. To define the occurrence of faults, a defect density per cm² as shown in (FLYNN and HUNG, 2005) has been used. We considered a 90nm process technology that presents 0.28 defects/cm² for memory and 0.14 defects/cm² for a microprocessor. We also considered that one has, for each NoC node, a microprocessor with a small cache size for instructions and data (16K/16K), like the ARM11 MP Core processor, with total area equal to 2.54mm² (ARM - online). As the router area is very small compared with the microprocessor and memories, its value does not influence the calculation of the number of defects. With the defect density above mentioned, the average value for defects is 0.21 defects/cm². Let us imagine the worst case (and as of today, probably unlikely) scenario where the faults occurring in the node that includes processor, memory and a NoC router are concentrated only in the buffers of the NoC. The NoC faults number is given by (4.2),

$$Faults = DD\ X\ NA\ X\ TN \qquad (4.2)$$

where DD refers to defect density, NA is the area of a single node containing a router, microprocessor and memory and TN is the total number of NoC nodes. Considering a 4x3 NoC used in our experiments, the occurrence of faults will be approximately 0.065 defects for the entire NoC. However, in the near future, either CMOS (Complementary Metal Oxide Semiconductor) technology will aggressively scale or it will be replaced by nanoscale technologies. In both cases, in the same die size, the SoCs will show a great increase in complexity and more permanent faults will occur due to process parameters (DUMITRAS et al., 2003). In this context, we analyze the occurrence of a single fault in a NoC (reflecting current technologies), after this, we verify the behavior of a NoC if more faults happen (in a future technological scenario). For this later case we considered 2 faults in the whole NOC and for an extreme case, we considered the occurrence of 6 faults in the NoC, hence assuming a higher defect ratio of the current technology, aligned to what is foreseen in (HON and NAEIMI, 2008).

A cycle accurate traffic simulator in Java was utilized to evaluate the faults impact in the network (average latency) using the adaptive and original routers. The distribution of the cores, in the NoC, was specified in accordance with the communication needs of the cores, reflecting a design time choice, being based on the original application.

Using the mapping of fig. 4.4, the simulator explained in section 4.4 with the modification to be fault tolerant, we simulated 2 applications with the traffic simulator to verify the behavior of the NoC in the presence of faults in the buffers. These applications are MPEG4 and VOPD (BERTOZZI et al., 2005). First, we verified the behavior of the NoC considering a single fault, after we introduced 2 and 6 faulty buffers randomly and we analyzed the latency of the NoC for those cases. All the possibilities of faults above defined were simulated with random injection, totaling 50 simulations in the occurrence of each fault scenario (1, 2 and 6 faults in the buffers) for each application. We defined the buffer depth equal to 4 and the packet size contains 40 flits. We verified the latency results for three situations: for the original router without faults, for FTAR with the three fault possibilities (1, 2 and 6 simultaneous faults) and for the adaptive router (AR) without faults. Fig. 4.16 presents the average latency results obtained for VOPD and MEPG4 applications.

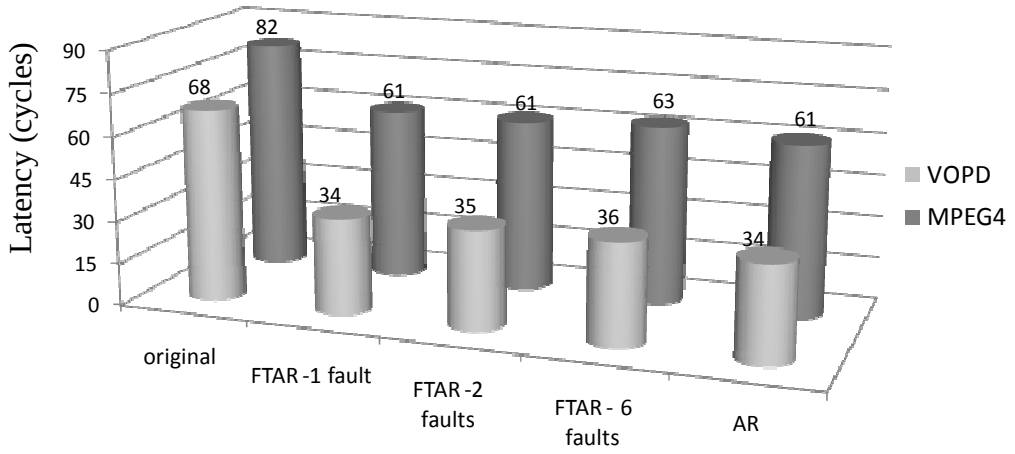


Figure 4.16: Latency results for three situations using the original, adaptive and fault-tolerant router.

In accordance with fig. 4.16, we verify that for the VOPD and MPEG4 applications, the FTAR with faulty buffers shows a minimal increment in the latency when compared with the adaptive router, but once compared with the original router, it still shows great gains in terms of performance (roughly 50% improvement), thanks to the available reconfiguration, even in the presence of random fabrication faults. These results were presented in Concatto et al. (2009) - SBCCI.

4.7.2 FTAR Area, Power and Frequency Results

The proposed fault-tolerant adaptive router was described in VHDL. ModelSim tool was used to simulate the code and Synopsys Power Compiler was used for synthesis. We analyzed the average power consumption, area and frequency results to a TCMS CMOS 0.18um process technology using the Synopsys Power Compiler tool. Table 4.1 presents results of two architectures (original and FTAR routers) with buffer depth equal to 4. We can see that the FTAR has larger power and area and smaller frequency, which are all disadvantages under this comparison. However, for the same buffer depth, the FTAR shows a lower latency (as shown in fig. 4.16) and it is tolerant to multiple simultaneous permanent faults.

Table 4.2 presents the results of synthesis to the original, AR, FTAR and FTAR + Hamming code, but now we require both NoCs (original and adaptive) to have the same latency values (same number of cycles with a 100MHz clock). Table 4.2 shows that the router with FTAR presents low power and high performance and this is explained because the proposed FTAR dynamically adapts the buffer depth in accordance with its need. In this case, no unnecessary buffers are used, as opposed to in the original router and hence the power consumption is reduced.

Table 4.1: Synthesis results to Original and FTAR router

	Original router	**FTAR**
Buffer Depth	4	4
Area (μm^2)	182	354
Power Consumption (mW@100MHz)	8.5	10.43
Frequency (MHz)	273	225

We have also compared the proposed solution with a router fully protected by TMR. The area overhead of a TMR router is 1068 μm² and the power consumption is 56.43mW. In this case, taking the numbers of Table 4.2 as a reference, the proposed solution would be 1.77 times smaller than TMR, with 2.73 times less power dissipation, with the same latency and withstanding multiple faults even in different blocks (something that could make a TMR circuit fail), in both cases the network is fully protected.

Using the combination of techniques here proposed, 100% of the NOC circuit is now protected (links, buffers, switch matrix and fsm). These results were presented in Concatto et al. (2009) -DSNoC too.

Table 4.2: Synthesis results to Original, Adaptive (RA), FTAR, FTAR(link) and FTAR(link, buffer, switch matrix, fsm) router

	Original router	**RA**	**FTAR**	**FTAR(link)**	**FTAR(link, buffer, switch matrix, fsm)**
Buffer Depth	9	4	4	4	4
Area (μm2)	356	337	354	394	603
Power Consumption (mW@100MHz)	16.81	9.34	10.43	14.07	18.47
Frequency (MHz)	225	230	225	185	185

The four composed methods (TMR, Hamming, switch matrix and FTAR) present hardware overhead, but significantly improve the reliability, the yield, reduce power and still increase the performance of the system when compared with the original router.

5 STRATEGY BASED ON ALGORITHM-LEVEL: ADAPTIVE ROUTING ALGORITHM

In this chapter we show that adaptive routing algorithm is a technique that one can use to avoid faulty links and routers, but first we review routing algorithms classification. Finally, at the end of this chapter we present a proposal of dynamic routing algorithm to avoid faulty links and routers using XY routing algorithm in a NoC Torus and results.

5.1 Introduction

The routing algorithm substantially affects the performance and the communications of the entire network. In a NoC the routing algorithm is used to determine the path of a packet traversing from the source to the destination (BENINI and DE MICHELI, 2002).

The routing algorithm, according to SCHONWALD et al., (2007), can be classified in three categories depending on the degree of adaptability:

I. Non-adaptive – which only uses one of the shortest paths between source and destination;

II. Partially adaptive – which use some of the shortest paths between the source and destination, but not all packets are allowed to use any path;

III. Fully adaptive – which allow to route packets on any path.

Partially and fully adaptive can be distinguished whether they are minimal or non-minimal. Minimal routing algorithms only use the shortest path between sender and receiver. A non-minimal routing algorithm does not use only the shortest path (SCHONWALD et al., 2007).

It is possible to find another classification of routing algorithms: deterministic or adaptive. In a deterministic routing algorithm the transmission of the packet is done using information about the source and the destination, i.e., in a XY routing algorithm, the packets first are sent in the X axis and then along the Y axis. The deterministic routing can be easily implemented and the latency of the packets is low, when there is no congestion or faults in the network. However, the throughput of the network decreases when the rate of packets injection or the rate of faults in the network increases.

In the adaptive routing algorithm, the path of the packets is chosen according to the congestion or faults in the network. When the network is congested or faulty, another path is chosen to avoid the faulty path. The adaptive algorithm can increase the

throughput and decrease the latency, nevertheless it adds overhead in terms of silicon area and energy consumption (ZHU et al., 2007). The next section presents the proposed routing algorithm.

5.2 Fault Tolerant Routing Algorithm (FTRA)

We base our analysis on a packet-switching network model called SoCIN (ZEFERINO and SUSIN, 2003), which is composed of routers named Router Architecture for Systems on Chip (RASoC). The RASoC router consists of five input and five output ports. One pair of I/O ports (called the local channel, L) is dedicated to the connection between the router and the core; the remaining four pairs (north - N, south - S, west - W and east - E) connect the RASoC router with four adjacent routers. These ports include two unidirectional opposite channels, each with data, framing (known as beginning and end of packet signals) and handshake communication signals.

The router is a VHDL soft core, with parameters in three dimensions: communication channel width, input buffer depth and routing information width. The architecture uses the wormhole switching approach and a deterministic source based routing algorithm. The wormhole approach breaks a packet into multiple flow control units called flits and its size is equal to the channel width. The first flit is a header with destination address followed by a set of payload flits and a tail flit.

Networks that use the wormhole switching approach have the target node address in the packet header, which is the first flit of the packet. By reaching a router in the path, the header allocates the path for the oncoming flits that carry the payload.

The network uses credit-based flow-control and XY routing, where a packet is first routed on the X direction and then on the Y direction before reaching its destination (ZEFERINO and SUSIN, 2003). RASoC applies the handshake protocol for link flow control and uses Round-Robin arbitration and input buffering.

Unless stated otherwise, the experiments presented are based on the RASoC configured as follows: communication channels with a four-position input buffer depth and the links with a 10-bit data width. Each communication link actually has 12 bits: eight bits for data, two extra wires for control (stored in the input buffer), called framing bits or the *bop* and *eop* bits, and two signals for the communication handshake between routers (*val* and *ack*). We assume that, in each node, a core is connected to a router through a network interface (NI) or wrapper.

Each router has in the network interface one TDG (Test Data Generator) and one TED (Test Error Detection) that sends and receives the test vector in the diagnosis phase, as depicted in fig. 5.1. The data interconnect wires consist of 8-bit wires, disregarding the four available control wires (bop, eop, ack, val).The diagnosis is made off-line, that is, when the NoC is not used. The fault location is sent to the router through a signal and this information is sent to the router to be used when the NoC is on-line.

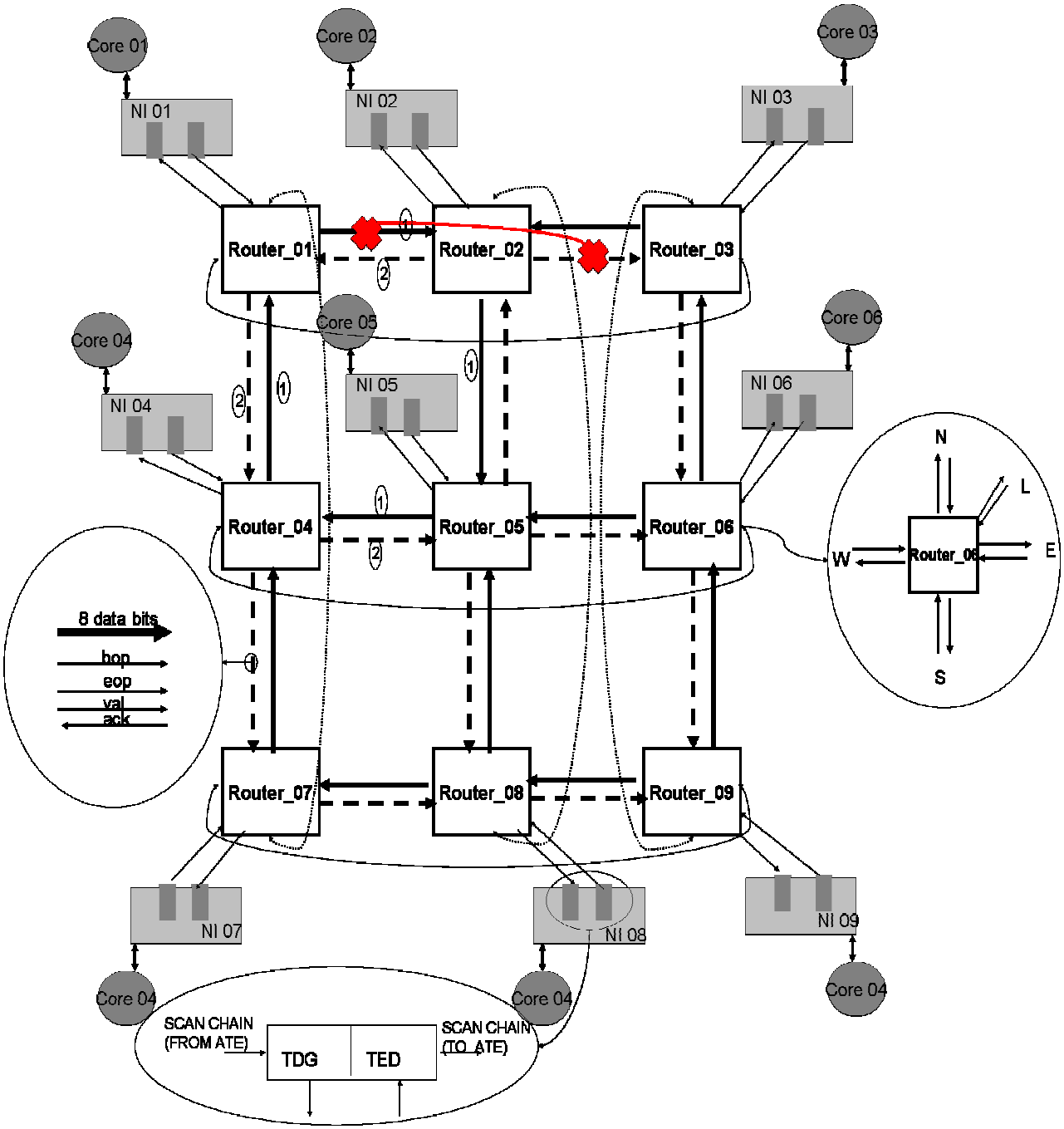


Figure 5.1: 3x3Torus-NoC with BIST in the NI.

The adaptive routing proposed for the torus NoC consists of dynamically changing the target address in a header when the original address uses a faulty channel. This change is performed by the router (based on the data that was sent to the router when the diagnosis was performed) on each header that tries to use the faulty channel.

In a torus network of size *m*x*m*, a packet has two possible routes in the same direction: it may proceed X steps one way (positive) or m – X steps the other way (negative). So, for instance, in a 3x3 NoC, a packet from router01 to route02 can be transmitted with a single step (X=1) to the East or with 2 steps (m - X = 3 – 1 = 2) to the West.

To implement the dynamic routing strategy, the router must store the fault location and the network size, so that it can calculate the new number of steps for the packet when necessary. Thus, when calculating the output port for a given packet, the routing algorithm first checks the list of faulty channels (which has been previously filled in after the diagnosis process and through the NI). If the original output channel is indicated as faulty, the alternative (longer) path replaces the original one in the packet header and the packet is routed normally. Fig. 5.2 shows the original path without faults. In fig. 5.2 and 5.3 we remove the links that are not used to send the packets between node 3 to node 5.

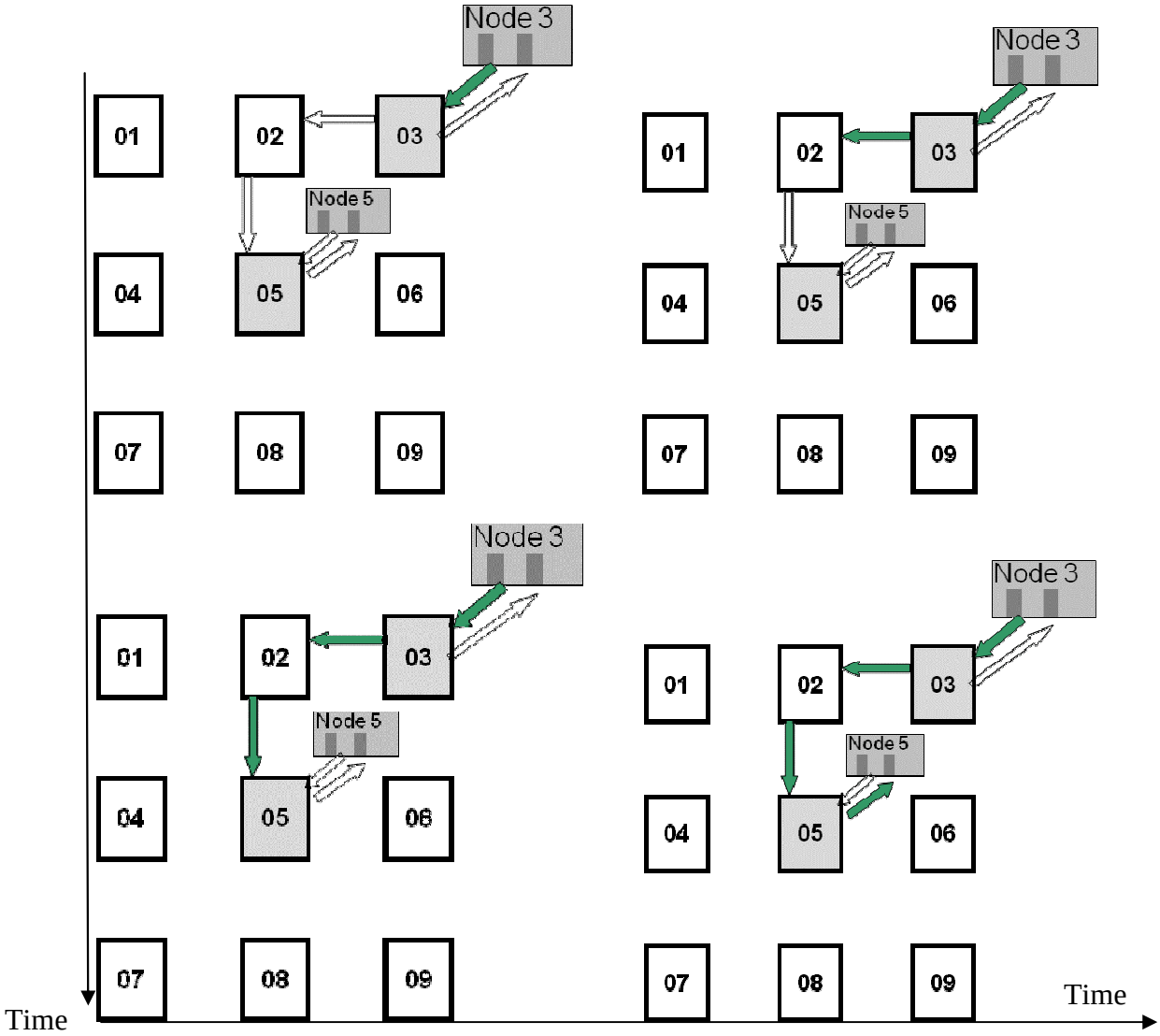


Figure 5.2: Routing Algorithm without fault.

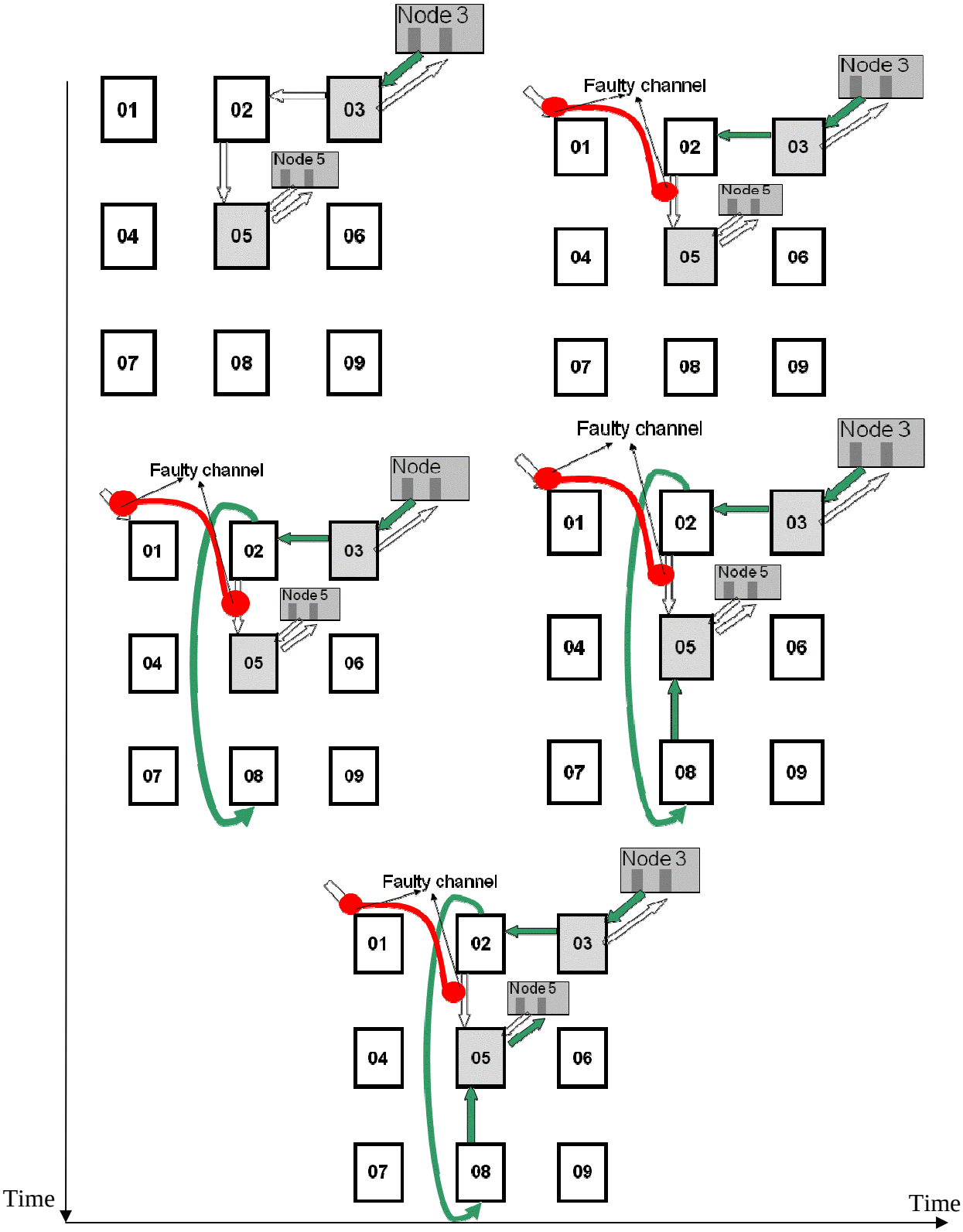


Figure 5.3: Proposed Routing Algorithm with faulty channels.

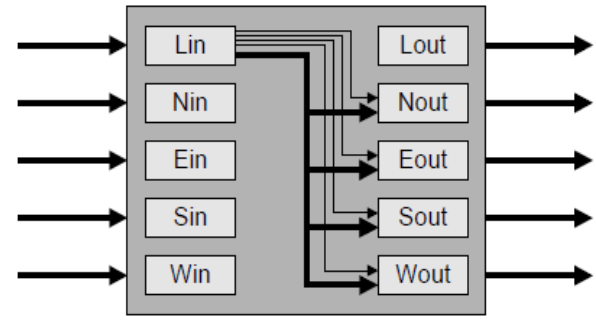
In fig. 5.3 there is a short-circuit between one wire in the channel NI01_to_router01 with a wire in the channel router02_to_router05. Thus, for the fault shown in fig. 5.3, a packet traveling from NI03 to NI05 will have its header changed in router02, due to the faulty channel in router2_to_router5. Instead of using the direct channel router02_to_router05, the header will take the feedback link router02_to_router08 and then the channel router08_to_router05 to achieve the target node.

We note that the adaptive routing can tackle the faults affecting only the channels between two routers. Links between a router and a core are not considered, because for those links there is no path redundancy. Faults affecting those links must be tackled by either using spare wires in the channel or by a new distribution of tasks among the other cores.

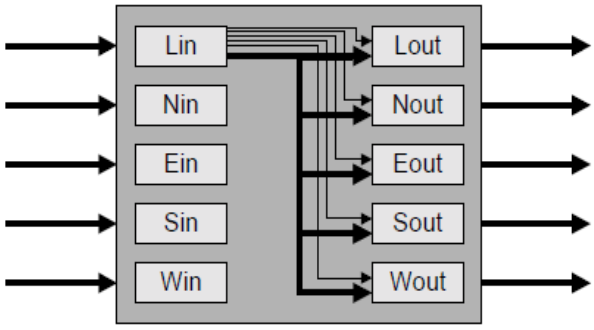
The adaptive routing algorithm can avoid the faulty routers, this can be done by modeling the faulty router as if all input channels were faulty. For instance, the neighbors of a faulty router understand the fault as if their own output channels were faulty.

For the SoCIN NoC in an *mxm* fault-free torus topology, a packet travels no more than *m*-1 steps from source to destination if *m* is odd, or *m* steps if *m* is even. Although each flit in the message will be re-routed, the impact on the overall message is at most (m+n-2) steps, because of the wormhole switching approach. Thus, we can conclude that the impact on system performance is really minimal.

In the original RaSoc's implementation, one input channel is not internally connected to the output channel with the same direction. As we see in fig. 5.4(a), for example, the input Lin is connected to four outputs, instead of five. The connection that is missing needs to exist for our purposes, since when the router notices a fault, sometimes the message is sent to the output by the same direction it arrived. So for the adaptive routing algorithm to work properly we connect all inputs to all outputs. One can see in fig. 5.4(b) that Lin is connected to Lout, the same does not happen in fig. 5.4(a) for the old architecture.



(a)



(b)

Figure 5.4: (a) Original Router and (b) new interconnections inside the router.

5.3 Performance and Synthesis Results

5.3.1 FTRA Evaluation

To evaluate the proposed technique a 3x3 Torus SoCIN NoC was implemented in VHDL together with a fault injection mechanism capable of inserting AND-shorts between all pairs of data wires in the network. The simulation was performed using ModelSim tool.

In our experimental setup a test bench simulates the ATE. Diagnosis is done by analyzing the responses of the TEDs and then comparing the erroneous flits (number is given by TED counters) against the expected contents.

Under the single fault assumption, the proposed adaptive routing guarantees that all messages will correctly arrive at the destination nodes for all diagnosed faults. The faults that cannot be tackled are:

(i) a shortcut between two outputs of the same router, in the same direction but in different ways; either north to south or east to west.

(ii) a short-cut between two inputs of the same router, in the same direction but in different ways either north to south or east to west.

In the first case, the router becomes unable to route any message in that direction, making it unusable and limiting the network's capability. In the second case, the problem is that the router cannot receive any message coming from that direction, also making it unusable. The mentioned limits are showed in fig. 5.5.

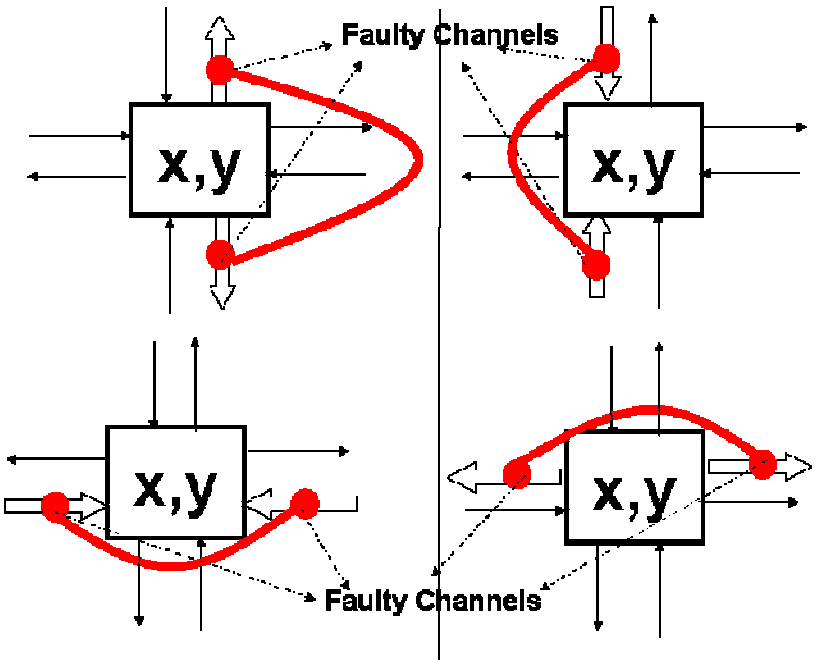


Figure 5.5: Fault limits of proposed routing algorithm.

As demonstrated in table 5.1, for our case study we have 288 (36 router to router channels x 8 data bits) data wires connecting routers. The number of data lines that can be shorted together is thus 41,328, as shown in equation 5.1. For each router, considering the exception cases described above, the proposed solution cannot deal with shorts involving 4 possible combinations of channels (either north to south or east to west. inputs and outputs). However, for each of these combinations, only shorts between wires in opposite directions will cause routing problems. For example, combining the data wires of N and S inputs we have a total of 120 possible shorts. Hence, in each combination of channels, we have 64 combinations of wires (8 wires of one channel x 8 wires of the other) where we might have limitations to tolerate the fault. With 4 combinations of channels in each router, we have 4x64=256 possible shorts that cause problems, reaching 2,304 (256x9 routers) in the entire network. This value represents 5.57% of the 41,328 wires in the NoC. These results were published at IOLTA Concatto et. al (2009) .

$$\#\ faults = C_2^{288} = \frac{288!}{2*(288-2)!} \qquad (5.1)$$

Table 5.1: Results for a 3x3 NoC Torus topology

total # of channels between routers	total # of shorts between rr_links	total # of shorts limitation for one router	#total of shorts limitation for 3x3 NoC
36	41,328	256 (64X4channels)	2,304 (256X9routers)

In the torus NoC, the alternative path is always longer compared to the original one. Nevertheless, we expect a minimum overhead, because the number of extra hops added to the communication is at most (m+n-2), where m and n are the respective paths in the X and Y dimensions of the network. Moreover the larger the NoC, the smaller is the impact on performance of the adaptive routing, since the affected channels (assuming a single fault) represent a smaller percentage of all active channels in the network.

We have chosen two communication patterns commonly used to evaluate interconnection networks (Duato et al., 1997) namely, the matrix transpose and butterfly models, to base our analysis. Such models have been largely used as benchmarks for the evaluation and comparison of network performance. In Matrix Transpose the node with binary co-ordinates an-1, an-2, …, a1and a0 communicates with the node an/2-1,…, a0, an-1,… and an/2. The Butterfly traffic is formed by swapping the most and least significant bits: the node with binary co-ordinates an-1, an-2, …, a1 and a0 communicates with the node a1, a0, …, an-1, an-2.

To estimate the performance penalty of the proposed solution, a high-level model of the SoCIN NoC was implemented in Java together with the defined communication models and a simpler fault injection mechanism. Since the granularity of the routing algorithm is the channel, only one fault per link can be considered in this analysis. Thus, instead of injecting all possible shorts among wires from rr_links, we have considered only shorts between any two rr_links and one fault related to intra-link shorts. This approach reduces the number of injected faults to the number of rr_links plus the combination of all rr_links in groups of two. For each injected fault, alternative paths are defined in the simulated network following the proposed adaptive routing algorithm.

Fig. 5.6 shows the results of the performance analysis for the two patterns of communications and three NoC sizes. One can observe that the average number of hops for the matrix transpose application increased only 8% in the 3x3 NoC. This impact was smaller for the 5x5 NoC (4%) and even smaller for 7x7 NoC (2%), as expected.

In the butterfly traffic there are more nodes sending messages than in the matrix transpose model, hence more channels are used. However, a fault affects less communications in the butterfly pattern compared to the matrix transpose model. This explains the smaller impact on the average communication cost for the same faults in the butterfly model. Thus, the matrix transpose model has a higher impact in performance compared to the butterfly one.

We note that although the average performance penalty is minimal, the designer can easily extract the exact penalty caused by each diagnosed fault in the NoC. For instance, in a 7x7 NoC the average number of hops for the transpose model increases by only 2% (considering all faults). However, for some communications, the effect of a fault can be important. For the same system, a total of 1,228 faults (out of the total 4,851 faults) lead

to a worst case scenario in terms of performance penalty on individual communications. For those faults, three communications are affected out of all 41 communications defined. The number of hops in the affected communications changes from 2, 6 and 4 respectively in the fault-free NoC to 7, 11 and 9 respectively in the faulty NoC. Thus, the performance penalty on some communications can be as high as 5 hops in this case. If, for the communications affected by the fault, the penalty imposed by the adaptive routing is not affordable, system-level techniques such as redistribution of tasks among functional units should be used to keep system performance.

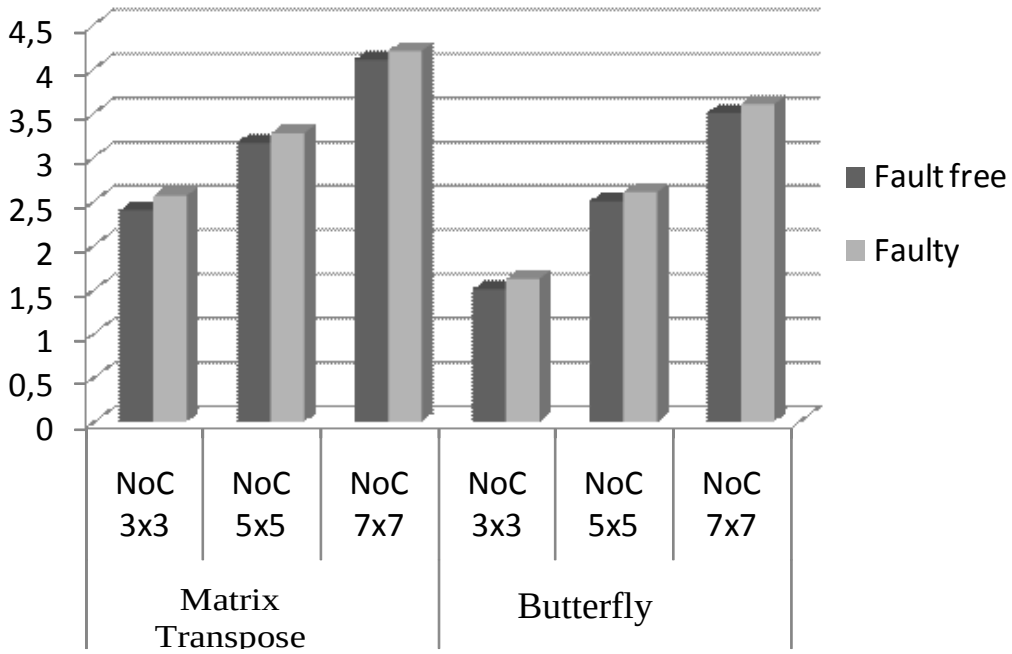


Figure 5.6: Average numbers of hops for NoC Torus.

5.3.2 FTRA Area, Power and Frequency Results

We analyzed the area and frequency results of the original and proposed router. Results were obtained for a CMOS 0.18um process technology using the Synopsys Power Compiler tool. Table 5.2 shows that the throughput is 115Mbits/s for 32-bits words and FIFO size equal to 4.

The area overhead for the adaptive routing is minimum, the reasons for this is that the routing algorithm only adds two adders to re-recalculate the new path. However as we have BIST in the network interface to execute the test and diagnose the fault location, the area overhead increases. The area overhead with BIST is 50% of the area of the RASoC router.

The adaptive routing algorithm does not use virtual channels to avoid the deadlock. As we are using the XY-routing algorithm, the deadlock does not happen and we do not need to use the routing table, because we are still working with a regular network. As we do not use the routing table and virtual channels, our area is smaller than the techniques that use routing table or virtual channels.

In terms of clock cycles, the experiments have shown that the number of cycles that the router takes to transmit the message from an input to an output port remains the same as the original RaSoC, that is, the router has three clock cycles of latency to retransmit the packet from the input to the output. However, the number of hops, with the use of adaptive routing algorithm in some cases will increase, because some new path will have more intermediate routers.

The frequency is the same with BIST or without BIST, because the BIST runs in parallel with the router. The power consumption in the router remains almost the same, because the increase of hardware was very small. The power consumption will probably increase in the links, since when a link is faulty, the packet will travel perhaps through more hops in the network.

Table 5.2: Synthesis results to Original and FTRA (Fault Tolerant Routing Algorithm) router

	Area (μm^2)	Frequency (MHz)	Power Consumption (mW @ 100 MHz)
Original	182	273	8.55
FTRA	183	270	8.58

6 COMPARING FTRA AND FTAR WITH RELATED WORKS

The FTRA and FTAR works can be confronted with Frantz et al. (2006) and Schönwald et al. (2007), because all of them protect the routers and channels. FTRA and Frantz works at design level and protect router and link. FTRA and Schönwald work at algorithm level and can protect link and router. Frantz et al. (2006) technique is a design level based technique. The buffers and the links are protected. Hamming Code protects the packets in the buffers and in the link. In Frantz the results about performance are not shown. The synthesis results show that the area overhead increases significantly, however according to Frantz, 100% of the network is protected. Reading this work, we observe that no strategy is shown to protect the switch or the FSM.

Table 6.1 shows some particularities of the compared works. The table presents: type of faults (transient, intermittent, permanent) that the techniques protect, the topology and routing needed and how faults are located. Table 6.2 is another observation about the compared works. Table 6.2 shows the area overhead, frequency penalty and area vulnerable to fault in each proposal.

Frantz has the ability to protect NoC against tree kinds of faults and it is the most generic work. In Frantz the type of technique used is independent of detection and location; however the overhead of this work is high. Table 6.2 shows that the area overhead of Frantz's work is 56%, however this result is only for channel width equal to 8. To encode 8 data bits using hamming code one needs 3 bits for encoding, resulting in a link with 11 bits. For 32 data bits one needs 6 bits for coding, resulting in a link equal to 38 bits. Concluding, the area overhead is increased when buffer depth and channel width increases. Hamming code is implemented by a cascade of XOR. For 8 bits the depth of cascade is 3 levels of XORs, however for 32 bits the depth is 5 levels of XORs, so the frequency will decrease.

FTAR (link, buffer, fsm, switch matrix) protects the NoC against permanent fault and the parts protected by hamming code and TMR are protected against transient faults too. Related to topology and routing, FTAR is also generic. FTAR (buffers and crossbar) are based on faults detection and location, because the buffers and crossbar need to know the location of faults to avoid them, but the FSM and links are independent of detection and location. Since FTAR is very generic the price is paid in area and frequency, as shown in table 6.1, where the area overhead is 80% and the frequency penalty is 34%, however these results are for 32 bits of data and a buffer depth equal to 4. We can see that the FTAR (link, buffer, fsm, switch matrix) can tolerate the faults and sustain the performance.

Frantz seems to be a good strategy, because its overhead is lower than FTAR. In Frantz the channel width is 8 bits and in FTAR it is 32 bits. FTAR has a lower area overhead and less frequency penalty when compared with Frantz, when comparing the same channel width in each. Another difference between Frantz and FTAR is that in Frantz, a technique to protect the crossbar or the FSM of the router is not shown, because of concerns about transient fault, however for permanent faults is important to protect these parts too.

Schönwald protects NoC against permanent faults and related to topology and routing it is as generic as Frantz. Schönwald works at algorithm level like FTRA. It is based on faults detection and location, as shown in table 6.1. Schönwald needs to know the locations of faults to avoid them. The location of faults is used to build the routing table, as shown in table 6.3. According to the faulty link the routing table is updated to avoid the faults. As a routing table is used, the routing algorithm is independent. The results about performance are not shown in the paper and therefore we do not have an argument to discuss. However, using another works which implement routing tables and show their results, we can conclude that the area overhead will be high, since to implement routing tables too much memory is used and, as was written in chapter 3, this routing table grows $n^{2,}$ where n is the number of cores.

Schönwald et al. (2007) works at algorithm level. To route the packets between the network Schönwald uses tables (memories) to store the number of hops that each router needs to reach the destinations router in the networks. Table 6.3 has a piece of the table from a routing table. Each router has a routing table with all routers in the network followed by the number of hops in each direction (north, south, west and east). In such a case, each router that is included in the network will be added as a destination router in the routing table of other routers. The same does not happen with FTRA, because the FTRA solution uses adaptive XY routing algorithm.

FTRA is the technique that has lower area overhead and less frequency penalty as shown in table 6.2. However as one can see, FTRA is very specific (table 6.1), it is only for torus network and XY routing. For single fault, FTRA shows that the performance overhead is less than 10% for a NoC 3x3 and even less when the network increases its size. The strategy at algorithm level has better results in performance and the area overhead is minimal because it does not use a table, nevertheless FTRA can only be applied in torus topology.

The results about fault tolerance presented in Schönwald show that it can tolerate much more faults than the XY-routing algorithm without fault tolerance. This is evident because Schönwald compares a non fault tolerant XY routing algorithm with his work, which has a fault tolerant technique. The FTRA has an adaptive XY algorithm that isolates faults. Related to Schönwald, it is a better solution if the NoC is a torus topology, because its penalty is very low, as one can see in table 6.2.

Table 6.1: Analysis of the proposals.

Technique	***Type of faults***	***NoC Topology***	***Routing Algorithm***	***How faults are located***
Frantz	transient, intermittent, Permanent.	independent	independent	do not need
FTAR (link, buffer, fsm, switch matrix)	permanent*	independent*	independent	4 register with 4 bits for each router
Schönwald	permanent	independent	independent	Table for each router
FTRA	permanent	Torus	XY	1 register 4 bits for each router

Table 6.2: Analyze of the proposals Overheads.

Technique	***Area Overhead (%)***	***Frequency penalty (%)***	***AVF***
Frantz	56*	20*	0*
FTAR (link, buffer, fsm, switch matrix)	80	34	0
Schönwald	NS	NS	0
FTRA	1	0	0*

Table 6.3: Routing Table of a router in a NoC used in Schönwald.

Port	North	East	South	West
Number of hops to destination 0	2	4	4	2
Number of hops to destination 1	1	3	5	3
Number of hops to destination 2	2	2	4	4
Number of hops to destination 3	3	5	3	1

7 CONCLUSION

The nanoscale manufacturing process is becoming less and less reliable, thus leading to lower yield. Consequently, the performance of NoCs architectures will be severely affected due to the presence of permanent faults. Therefore, it is important that future Network-on-Chip be adaptive at run-time, so that one could avoid known failures.

In this work, two techniques to increase the yield and reliability of NoC were proposed. The first strategy uses an adaptive router to protect the buffers and sustain the performance of the system, hamming code to assure reliable links and triplicates the state machine of the input buffer for tolerating permanent data errors. The fault-tolerant adaptive router can still dynamically replace a faulty buffer word of the FIFO, borrowing flip-flops from the neighbor channel.

Experimental results demonstrated that the combination of FTAR (Fault Tolerant adaptive Router), Hamming Code, TMR and switch matrix shows penalties on frequency and area, but thanks to the reconfiguration scheme, one can sustain performance and reduce power dissipation, with 100% of the router area protected when compared with the non-fault tolerant adaptive router. With the fault tolerance technique at design level, it is possible to protect the buffers FIFO, the FSM FIFO and the link.

The buffers are protected only against permanent faults, but the link and the FSM FIFO are protected against soft errors too. The link can only recover from a single fault, but the FSM FIFO with TMR and the buffers FIFO with the adaptive scheme can recover from more than one fault. However the recovery behavior from the faults has limits. The limit for the FIFO buffers is when all buffers from its neighbors and their own are faulty. The limit of TMR technique is when there are two FSM FIFOs with a wrong answer. When some of the components that are not protected are reached by a permanent fault, the scheme at design level cannot protect the NoC.

The second strategy tolerates interconnect shorts in NoC communication channels for a 2-D Torus and faults in the router. With the proposed scheme one can tolerate a fault that affects one or two channels for NoC Torus larger than 2x2 and faulty routers thus increase the yield. The current work includes the analysis of the limits of the adaptive routing algorithm. Almost 95% of the channels between routers can be re-routed. This means that 66% of total channels of NoC can have the packets re-routed. One can see that the area overhead of the adaptive routing algorithm is small, as is the power consumption. However, the power consumption of the whole system will increase, because with the adaptive algorithm the packets stay longer in the channels when the channel is faulty.

7.1Future Work

For the FTAR, as future work, we intend to change the protecting of the link, that is, replace the hamming code with another technique and make high level simulations to have performance results.

FTAR with hamming protects the wires, however hamming code corrects the link from single faults. As future work we intend to protect the wires from the link using half duplex channels, as in YING-CHERNG et al., (2009). The use of half duplex channels can increase the performance and at the same time reduce the power consumption when compared with hamming code.

For the adaptive routing algorithm, the links between the node and router do not have a redundant path, so as future work one must add a solution for these channels and accomplish some experiment results to discover the latency and the power consumption of the whole system when the adaptive algorithm is used.

With these researches we saw that there are other ways to look at NoC and search for another kind of protection without loss in performance. With this thesis we started to study more about NoC topologies, adaptive NoC and we verify that there is a huge field in the research. As there are generic microprocessors, maybe it is possible to have a generic NoC that can adapt itself at run-time to guarantee some degree of performance and adaptability.

REFERENCE

ALI M., WELZL, M., HESSLER, S., An efficient fault tolerant mechanism to deal with permanent and transient failures in a network on chip, **International Journal on High Performance Systems Architecture,** Vol. 1, No. 2, pp. 113-123, 2007.

ANDREWS, J., BAKER, N., Xbox 360 System Architecture, **IEEE Micro**, vol. 26, no. 2, 25—37, 2006.

ARM: Available at: http://www.arm.com/products/CPUs/ARM11MPCoreMultiprocessor. Accessed October 2009.

AZIMI, M., KUMAR, A., KUNDU, P., PARK, S., SCHOINAS, I. AND VAIDYA,A. ,Integration challenges and tradeoffs for tera-scale architectures, Intel Technology, pp. 173–181, Aug. 2007. [Online]. Available: http:// www.intel.com/technology/itj/2007/v11i3/1-integration/1-abstract.htm.

BENINI, L., DE MICHELI, G., Network on Chips: A new SoC Paradigm. IEEE Computer, 70-78, 2002.

Bertozzi, D., Jalabert, A., Murali, S., Tamhankar, R., Stergiou S., Benini, L., De Micheli, G., NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip, **IEEE Transactions on Parallel and Distributed Systems**, vol. 16, no. 2, pp. 113-129, Feb. 2005, doi:10.1109/TPDS.2005.22

BO FU, AMPADU, P., A multi-wire error correction scheme for reliable and energy efficient SOC links using hamming product codes, IEEE International SOC Conference, p. 59-62, 2008.

CONCATTO, C., MATOS, D. , CARRO, L., KASTENSMIDT, F., SUSIN, A., COTA, E.. Improving Reliability in NoCs with a Reconfigurable Router. In: Workshop on Diagnostic Services in Network-on-Chips, DSNOC'09, 2009.

CONCATTO, C., MATOS, D., CARRO,L., KREUTZ, M., COTA, E., SUSIN, A., KASTENSMIDT, F.. Fault Tolerant Mechanism to Improve Yield in NoCs Using a Reconfigurable Router. In: 22nd Symposium on Integrated Circuits and System Design, 2009, Article No. 26 . ISBN:978-1-60558-705-9.

CONCATTO, C., ALMEIDA, P., KASTENSMIDT, F., COTA, E., LUBASZEWSKI, M., HÉRVE,M. . Improving Yield of Torus NoCs Through Fault- Diagnosis-and-Repair

of Interconnect Faults. In: **15th IEEE International On-Line Testing Symposium**, IOLTS -2009, pp.61-66.

COTA, E. , CARRO, Luigi, LUBASZEWSKI, M., Reusing an On-Chip Network for the Test of Core-based Systems. **ACM Transactions on Design Automation of Electronic Systems**, v. 9, n. 4, p. 471-499, 2004.

COTA, E., KASTENSMIDT, F, CASSEL, M., HERVE, M., ALMEIDA, P., MEIRELLES, P., AMORY, A., LUBASZEWSKI, M. . A High-Fault-Coverage Approach for the Test of Data, Control and Handshake Interconnects in Mesh Networks-on-Chip. **IEEE Transactions on Computers**, v. 57, p. 1202-1215, 2008.

DALLY, W.J. TOWLES, B., Routes Packets, Not wires: on chip interconnection networks, Proceedings Design Automation Conference, p.684-689, 2001.

DUATO, J., YALAMANCHILI, S. and NI L., Interconnection Networks, an Engineering Approach. In: **IEEE Computers Society Press.** 1997.

DUMITRAS, T., KERNER,S., MARCULESCU R., Towards On-Chip Fault-Tolerant Communication. In: **Proc. Asia and South Pacific Design Automation Conference** (ASP-DAC), pp. 225-232, 2003.

FLYNN, M.J., HUNG P. Microprocessor Design Issues: Thoughts on the Road Ahead. **IEEE Micro**, vol.25, n 3, pp. 16-31, 2005.

FRANTZ, A., KASTENSMIDT, F., COTA, E., CARRO, L., Dependable Network-on-Chip Router Able to Simultaneously Tolerate Soft Errors and Crosstalk, **Proceedings International Test Conference (ITC)**, vol. 1, pp. 1 – 9, 2006.

FRAZZETTA, D., DIMARTINO, G., PALESI, M., KUMAR, S., CATANIA, V., Efficient Application Specific Routing Algorithms for NoC Systems utilizing Partially Faulty Links, In **11th EUROMICRO Conference on Digital System Design Architectures, Methods and Tools,** DSD '08, pp. 18-25, 2008**.**

FURBER, S. Living with Failure: Lessons from Nature? **Proceedings of the Eleventh IEEE European Test Symposium(ETS'06)**-Vol.00, p.4-8, 2006

GRECU, C., PANDE, P., IVANOV, A., SALEH, R.; BIST for network-on-chip interconnect infrastructures, **24th IEEE VLSI Test Symposium**, VTS, 2006, 6 pp.

GRECU, C., IVANOV, A., SALEH, R.; SOGOMONYAN, E.; PANDE, P., **On-line Fault Detection and Location for NoC Interconnects. In: IOLTS06**.

HERVE, M. B., COTA, E., KASTENSMIDT, F. G. L., LUBASZEWSKI, M., Diagnosis of Interconnect Shorts in Mesh NoCs. In: **ACM/IEEE International Symposium on Networks-on-Chip**, 2009, San Diego. Proceedings. Los Alamitos: IEEE Computer Society, 2009. v. 1.

Hamming Code– Available at: http://en.wikipedia.org/wiki/Hamming_code. Accessed October 2009.

HON, A.D.; NAEIMI, H., Seven Strategies for Tolerating Highly Defective Fabrication, **Proceedings of the 21st annual symposium on Integrated circuits and system design**, pp. 34-39, 2008.

HOROWITZ, M., MAI, K., HO, R., The future of wires. In: Proceeding of IEEE, 2000, Vol89, Issue 4, p490-504.

KOIBUCHI, M., HIROKI, M., AMANO, H., PINKSTON, M., A Lightweight Fault-Tolerant Mechanism for Network-on-Chip. In: **Second ACM/IEEE International Symposium on Networks-on-Chip**, pp 13-22, 2008.

MANFERDELLI, J., GOVINDARAJU, N., CRALL, C., Challenges and Opportunities in Many-Core Computing, **Proceeding of the IEEE**, 96, no. 5, 808—815, 2008.

MATOS, D., CONCATTO, CARRO, L., KREUTZ, M., SUSIN, A., KASTENSMIDT, F.. NoC Power Optimization Using a Reconfigurable Router. In: **IEEE Computer Society Annual Symposium on VLSI**, 2009, pp.235-240.

MATOS, D., CONCATTO, CARRO, L., KREUTZ, M., SUSIN, A., KASTENSMIDT, F., Highly Efficient Reconfigurable Routers in Networks-on-Chip. In: **17th IFIP/IEEE International Conference on Very Large Scale Integration**, 2009, Florianopolis, Brasil.

MATOS, D., CONCATTO, C., CARRO, L., KASTENSMIDT, F., SUSIN, A., The Need for Reconfigurable Routers in Networks-on-Chip. In: **International Workshop on Applied Reconfigurable Computing,** ARC 2009, 2009, p. 275-280

NICOLAIDIS, M., Design for soft error mitigation, **IEEE Transactions on Devices and Materials Reliability**, vol. 5, Issue 3, pp. 405-418, 2005.

PANDE, P.P, GRECU, C., IVANOV, A, SALEH, R., Design of a switch for network on chip applications, **Proceeding of International Symposium on Circuits and Systems, ISCAS'03**, Vol. 5, pp217-220, May 2003.

ROSSI, D., METRA, C., NIEUWLAND, K., KATOCH, A., Exploiting ECC Redundancy to Minimize Crosstalk Impact, **IEEE Design & Test of Computers**, vol. 222, Issue1, pp. 59-70, 2005.

SCHÖNWALD, T., ZIMMERMMAN, J., BRINGMAN, O., ROSENSTIEL, W. Fully., Adaptive Fault-Tolerant Routing Algorithm for Network-on-Chip Architecture, **10th Euromicro Conference on Digital System Design Architectures, Methods and Tools,** p.527 – 534, 2007.

SRINIVASAN, K., CHATHA, K. S., A Low Complexity Heuristic for Design of Custom Network-on-Chip Architectures, **Proceedings of Design, Automation and Test in Europe Conference** DATE´06, 2006.

VESTIAS, M.; NETO, H., Area and Performance Optimization of a Generic Network-on-Chip Architecture, **Symposium on Integrated Circuits and Systems Design**, pp. 68-73, 2006.

WU, C., CHI, H., Design of a High-Performance Switch for Circuit-Switched On-Chip Networks, **Asian Solid-State Circuits Conference**, 2005, pp. 481-484.

YING-CHERNG LAN, SHIH-HSIN LO, YUEH-CHI LIN, YU-HEN HU, SOA-JIE CHEN, BiNoC: Bidirectional NoC Architecture with Dynamic Self-Reconfigurable Channel, **In 3rd ACM/IEEE International Symposium on Networks-on-Chip**. 2009.

YU, Q., AMPADU, P., Adaptive Error Control for NoC Switch-to-Switch Links in a Variable Noise Environment, **IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems**, pp 352-360, 2008.

ZHU, H., PANDE, P.P., GRECU, C., Performance Evaluation of Adaptive Routing Algorithm for Achieving Fault Tolerance in NoC Fabrics, **IEEE International Conf. on Application -specific Systems, Architectures and Processors**, 42-47, 2007

ZEFERINO, A., SUSIN, A., SoCIN: A Parametric and Scalable Network-on-Chip. In**: 17th Symposium on Integrated Circuits and System**, pp. 169-174, 2003.