THÈSE DE DOCTORAT DE L'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité Informatique

(École Doctorale Informatique, Télécommunication et Électronique)

Présentée par ZHEN ZHANG

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ PIERRE ET MARIE CURIE

DÉTECTION DES PANNES FRANCHES ET RECONFIGURATION AUTOMATIQUE DANS UN MICRO-RÉSEAU INTÉGRÉ SUR PUCE

Soutenue le 27 juin 2011, devant le jury composé de

Mme.	LORENA ANGHEL	TIMA	Rapporteur
M.	BRUNO ROUZEYRE	LIRMM	Rapporteur
M.	MARCELLO COPPOLA	STMicroelectronics	Examinateur
M.	ERIK JAN MARINISSEN	IMEC	Examinateur
M.	PIERRE SENS	UPMC	Examinateur
M.	ALAIN GREINER	UPMC	Directeur de thèse
M.	MOUNIR BENABDENBI	TIMA	Co-directeur de thèse

PH.D. THESIS OF THE UNIVERSITY PIERRE AND MARIE CURIE

Department : COMPUTER SCIENCE AND MICRO-ELECTRONICS

Presented by : ZHEN ZHANG

Thesis submitted to obtain the degree of
DOCTOR OF THE UNIVERSITY PIERRE AND MARIE CURIE

"ON THE FIELD" DETECTION, DE-ACTIVATION & RECONFIGURATION (ODDR) MECHANISM FOR PERMANENT FAULT-TOLERANCE OF NETWORK-ON-CHIP

Defence on 27th June 2011, Committee :

Mme.	LORENA ANGHEL	TIMA	Reviewer
M.	BRUNO ROUZEYRE	LIRMM	Reviewer
M.	MARCELLO COPPOLA	STMicroelectronics	Examiner
M.	ERIK JAN MARINISSEN	IMEC	Examiner
M.	PIERRE SENS	UPMC	Examiner
M.	ALAIN GREINER	UPMC	Advisor
M.	MOUNIR BENABDENBI	TIMA	Co-Advisor

Résumé

Cette thèse présente un mécanisme de tolérance aux pannes franches pour un micro-réseau intégré sur puce (Network-on-Chip), ce dernier utilisé dans une architecture de type MP^{2}SoC (Massively-Parallel Multi-Processors System-on-Chip). Ce mécanisme se décompose en trois étapes : 1) détection des composants du NoC, 2) désactivation des composants défectueux et 3) reconfiguration de fonction de routage.

Ce mécanisme est exécuté lors de chaque redémarrage du système ou de mise sous tension de puce, pour détecter et désactiver les composants défectueux du NoC, et pour activer les composants sans défaut. Ces composants activés seront alors utilisés pour lancer une auto-reconfiguration de fonction de routage du NoC. En conclusion, grâce au mécanisme, un NoC endommagé peut structurellement s'auto-tester, partiellement s'auto-désactiver, globalement s'auto-reconfigurer et fonctionnellement s'auto-réparer, après un redémarrage du système ou une mise sous tension de puce. En outre, ce mécanisme peut être utilisé dans les architectures multiprocesseurs, se basant sur un NoC de type 2D-Mesh.

Le mécanisme mentionné a été implémenté dans un NoC de type 2D-Mesh : DSPIN. Grâce à cette implémentation, nous l'évaluons du point de vue de la couverture de fautes, du surcoût matériel, du temps d'exécution, etc.

Il convient de noter que, le mécanisme peut aussi être utilisé pour améliorer le rendement de fabrication, en évitant de jeter celle-ci à cause d'un composant défectueux.

Mots-clés : micro-réseau sur puce, MP^{2}SoC, architecture multiprocesseurs, tolérance aux fautes, tolérance aux pannes franches, NoC auto-test, NoC BIST, reconfiguration du NoC, algorithme de routage, l'infrastructure de configuration, DCCI.

Abstract

This thesis presents a complete ODDR ("On the field" Detection, De-activation and Reconfiguration) mechanism, providing a permanent fault-tolerance for a 2D-Mesh Network-on-Chip (NoC) in a shared memory, Massively-Parallel Multi-Processors System-on-Chip (MP^2SoC) architecture.

This mechanism is executed at each system reboot or chip power-on, to detect and de-activate the faulty components of NoC and to activate the fault-free components ; then it makes NoC itself to achieve a self-reconfiguration through the fault-free/activated components. In conclusion, with the help of this ODDR mechanism, a damaged 2D-Mesh NoC can structurally self-test, partially self-disable, globally self-reconfigure and functionally self-recover, after a simple system reboot or at chip power-on. Moreover, this mechanism can be used in any 2D-Mesh NoC based, shared memory, multi-cores architecture.

The ODDR mechanism has been implemented in a typical 2D-Mesh NoC : DSPIN. Thanks to this micro-network, we evaluate and analyze the mechanism, from the point of view of the stuck-at fault coverage, of the silicon area overhead, of the execution time consumed, etc.

It should be noted that, the ODDR mechanism can be used not only "on the field", but also in the manufacture, where it's helpful to improve the yield by avoiding to throw the whole chip when one single component is faulty.

Keywords : Network-on-Chip, MP^2SoC, ODDR, "on the field", fault-tolerance, NoC test, NoC BIST, NoC reconfiguration, routing algorithm, reconfigurable, deterministic, configuration infrastructure, DCCI.

English Title : "On the field" Detection, De-activation and Reconfiguration (ODDR) Mechanism for the Permanent Fault-Tolerance of Network-on-Chip.

Acknowledgements

I would like to express my deepest gratitude to my advisor, Mr. Alain Greiner, Professor at LIP6 laboratory, who gave me the freedom and the confidence to explore on my own. I am grateful to his guidance to recover when my steps faltered. His patience and support helped me to finish this dissertation. I hope that one day I would become as good an advisor to my students as Alain has been to me.

I would like to thank my co-advisor, Mr. Mounir Benabdenbi, Assistant Professor at TIMA laboratory, who was always there listening, providing advice and classifying my ideas. I am deeply grateful to him for the long discussions that helped me sort out the technical details of my work, for his careful reading and commenting on countless revisions of this manuscript.

Mr. François Pécheux, Assistant Professor at LIP6 laboratory, deserves special thanks for sparking my interest in the research at the beginning of my postgraduate study, and inviting me to join the ADAM project. François and Etienne Faure gave the numerous helpful suggestions for my publications. Thanks also go to Dimitri Refauvelet for his cooperation in the ADAM project, and for his wonderful solution (DCCI, Distributed Cooperative Configuration Infrastructure) from which my work benefited.

I am grateful to Hamed Sheibanyrad and Nicolas Pouillon for their technique supports, Joël Porquet for his assistances. I am also thankful for the discussions with Christophe Choichillon about my future work, who cooperates with me to implement a new Network-on-Chip architecture in the TSAR project.

Furthermore, I want to thank Mrs. Lorena Anghel, Professor at TIMA Laboratory and Mr. Bruno Rouzeyre, Professor at the LIRMM laboratory, for their detailed reports, suggestions and comments. Thanks also go to Dr. Marcello Coppola from STMicroelectronics Grenoble, Dr. Erik Jan Marinissen from IMEC Belgium, and Prof. Pierre Sens from LIP6 laboratory, for giving me the honor of being my doctoral thesis examiners.

I greatly appreciate the members in the office “A102” of laboratory : Emna Amouri, Ludovic Noury, Haluk Ozaktas, Mathieu Rosière, Sami TakTak, for their friendship.

I am also thankful to other members of LIP6-SoC team, specially to Mme Shahin Mahmoodian for her patience and efficiency, and Mme Anne Derieux for her kindness and her help in improving my french.

Finally, my love and gratitude is extended to my family. I thank my parents for their supports, concerns and strength during these years of my studies particularly in the last three months. I appreciate having my wife in my life. I am deeply grateful for her unwavering understanding, constant supports, encouragement and companionship during these ten years.

Table of Contents

List of Figures

List of Tables

Listings

Introduction

The Network-on-Chip (NoC) [Benini and Micheli, 2002] based, shared memory, Massively-Parallel Multi-Processors System-on-Chip (MP^2SoC) architectures, are the future for the semiconductor industry, since such architecture can integrate thousands of IP (Intellectual Property) cores in a single chip. However, for such chip, the production yield is an issue [Furber, 2006], because a huge number of permanent faulty components will appear not only in the manufacture, but also "on the field" when the chip is already mounted on the final equipment. The NoC reliability is a key issue since it is a shared, global and irreplaceable resource. Thus, the "on the field" permanent fault-tolerance for NoC, must be taken into account and defined in the MP^2SoC architecture design. To our knowledge, there is no complete and feasible solution in the world.

Facing this problem, we propose an "on the field" permanent fault-tolerant mechanism for 2D-Mesh NoC used in shared memory MP^2SoC architecture, according to the graceful degradation theory. This mechanism can be summarized by three keywords : Detection, De-activation & Reconfiguration. The general principle is the following :

> At each system reboot or chip power-on, the mechanism is operated to detect & de-activate the faulty components of the NoC, and to activate the fault-free components, then to achieve the NoC auto-configuration using the fault-free components. In other words, the mechanism aims to make NoC self-test & self-recover at each system reboot or chip power-on.

In this thesis, we present and detail this fully automatic, "On the field" Detection, De-activation & Reconfiguration (ODDR) mechanism. This mechanism is implemented in a typical 2D-Mesh NoC : DSPIN [Panades et al., 2006] (Distributed Scalable Predictable Interconnect Network). The fault coverage, silicon area overhead, time consumed, the formal proof of deadlock-free property, etc. are evaluated, discussed and analyzed.

The organisation of this thesis is described as below :

In chapter 1 : We define and discuss three main problems of "on the field" fault-tolerance issues.

1. **"NoC Test"** : the faulty components of NoC (router and channel) must be detected by an appropriate built-in self-test (BIST) mechanism.
2. **"Configuration Infrastructure"** : the NoC reconfiguration mechanism requires the definition of a configuration master and a reliable configuration bus.
3. **"Routing Algorithm"** : a deterministic, fault-tolerant & reconfigurable routing algorithm must be defined. And this routing algorithm must fit the modification of NoC topology resulted from the faulty components.

In chapter 2 : We summarize the state of the art of NoC fault-tolerant strategies. And we present the related solutions for the three main problems and analyze their limitations in the context of "on the field" fault-tolerance.

In chapter 3 : We introduce a typical 2D-Mesh NoC : DSPIN. And we present the DSPIN based, shared memory MP^{2}SoC architecture that is used in this thesis to evaluate the ODDR ("On the field" Detection, De-activation & Reconfiguration) mechanism. This mechanism uses the following DSPIN characteristics :

- 2D-Mesh topology
- Full-Crossbar Switch
- FIFO-based Flow Control Policy
- Default routing algorithm : X-First

All the Network-on-Chip architecture having the stated characteristics, can adopt and implement the proposed "on the field" fault-tolerant approach.

In chapter 4 : We present the general scenario of ODDR ("On the field" Detection, De-activation & Reconfiguration) mechanism, used in the DSPIN based, shared memory MP^{2}SoC architecture. This scenario consists of three stages :

1. Initialization stage / NoC Test
2. Pre-configuration stage / Configuration Infrastructure
3. Configuration stage / Routing Algorithm

In chapter 5 : We detail the initialization stage and a fully distributed off-line BIST dedicated to 2D-Mesh NoC in a GALS (Globally Asynchronous, Locally Synchronous) context. This BIST is implemented in NoC as an initialization procedure, and executed at each system reboot or chip power-on. With the help of this procedure, the faulty components are detected and de-activated, finally configured to behave as a "Black Hole", while the fault-free components are activated to operate the normal function.

In chapter 6 : We details the pre-configuration stage, the used configuration infrastructure (DCCI, Distributed Cooperative Configuration Infrastructure), and the software-based diagnosis/localization of faulty component. The main function of DCCI is to construct a software communication tree on top of the initialized NoC, using only the faulty-free components. Each tree node is an embedded processor core. The tree root is the configuration master. The tree itself is the configuration bus. We use this DCCI tree to locate all "Black Holes" with a software application, finally to identify the modification of NoC topology.

In chapter 7 : We detail the configuration stage and the deterministic, lightweight, reconfigurable routing algorithm. Following this algorithm, the network itself is split into two regions : normal region and faulty region. The routers of normal region are not reconfigured, but the border routers of the faulty region are configured to create a cycle-free contour, aiming to bypass the faulty/disable routers. The routing function (of cycle-free contour) is proven to be deadlock-free with any single-faulty-router topology.

Chapter 1

Problem Definition

Contents

In this chapter, we discuss the problems addressed by this thesis and list the involved questions.

1.1 Thesis motivation

To face the challenge of high-performance multi-computing issue, the semiconductor industry has begun to integrate in a single chip a large number of processor cores interconnected by a Network-on-Chip (NoC). Vivid examples of such architecture are the Intel 48-cores Single-chip Cloud Computer (SCC) [SCC, 2009], the Intel Tera-flops 80-cores chip [Vangal et al., 2008], or the Tilera 100-cores TILE-Gx Processor [TILE-Gx, 2009]. In the incoming years, NoC-based, shared memory, Massively-Parallel Multi-Processors System-on-Chip (MP^2SoC) architecture, containing more than one thousand processors [Shekhar, 2007] will be used and implemented.

However, as stated by [Furber, 2006], zero-defect manufacturing of such architecture on a chip, with 100% fault-free components, is a big challenge that is very difficult, not to say impossible to achieve. The same author also recalls that during the first year of use, numerous faults are likely to occur "on the field", in chip operation, due to wear-out effects. To solve these two major problems, not only the technique improving the chip yield at manufacture time is needed, but also the "on the field", architecture-level fault-tolerant approach must be taken into account.

According to the industrial classification, the faults occurring on chip circuit can be classed into three families : permanent, transient and intermittent. Their definitions are summarized as below :

- **Permanent** faults are irreversible physical changes, due to some pollutions in manufacture process or wear-out effect during the operation of circuit. So an error caused by permanent fault cannot be recovered by hardware reset.
- **Transient** faults are induced by external temporary environmental events, such as ions or electro-magnetic radiation. They create non physical change. So all errors caused by the transient fault can be recovered by hardware reset.
- **Intermittent** faults are induced due to unstable hardware, in some environmental conditions, such as temperature or voltage change. Then all errors caused by the intermittent fault can be treated by hardware reset in the environment adjusted.

In brief, as the permanent fault can physically change the chip structure, it will strongly decrease the chip yield and lifetime. Thus, our objective all along this thesis is to provide solutions to improve permanent fault-tolerance.

A MP^2SoC architecture will contain a larger number of replicated identical components, such as the processor cores, the embedded memory banks, the network routers and

so on. In this kind of architecture, even if some of the components are faulty, the remaining ones are able to continue to work in a gracefully degraded mode. Borrowing the ideas from the error control process : Detection, Containment & Recovery [Dally and Towles, 2004] [1], we can define a new mechanism, as an "on the field", architecture-level fault-tolerant approach. For example, for the faulty processor cores, or the faulty embedded memory banks, once their erratic behaviors have been detected (detection), they will be de-activated (containment), and the software application will be remapped on the remaining operational hardware (recovery). Unfortunately, for the NoC itself, to realize the recovery mechanism is not always straightforward.

In order to save silicon area, and to minimize the network latency, most NoCs, taking advantage of the regular micro-network topology, use dedicated routing algorithms, such as X-First routing [Dally and Seitz, 1987] for a N×M 2D-Mesh topology. Thus, in a defective NoC, once the faulty components (router or communication channel) have been detected and de-activated (became holes), the NoC regular topology has been in fact modified. This results in a new irregular topology. With the original routing algorithm, the packets might be routed towards the hole, or be routed in a deadlock or livelock way, which, in the worst case, will lead to a whole NoC block. Therefore, recovery means to reconfigure the NoC itself and the global routing function, aiming to support the new topology.

The goal of this thesis is to define, realize and evaluate a complete "On the field", Detection, De-activation & Reconfiguration (ODDR) mechanism for a 2D-Mesh NoC used in a shared memory MP^{2}SoC architecture.

1.2 Analysis of "On the field" Detection, De-activation & Reconfiguration (ODDR)

As shown in Figure 1.1, three basic components of a 2D-Mesh NoC are : router, communication channel and network interface controller (NIC).

– **Router** : is the switching units of network. Its task is to route the packets from an input channel to an output channel, respecting the routing algorithm. The routing function module and switch module (such as crossbar) constitute the router structure.

1. The error control process : Detection, Containment & Recovery, is usually used to deal with a transient fault, as a packet-transmission-level fault-tolerant approach. Detection means to check out the failure of packet, using such as ECC (Error Control Code) or CRC (Cyclic-Redundancy Check) field. Containment aims to prevent the failure packet propagation. Recovery means the packet retransmission.

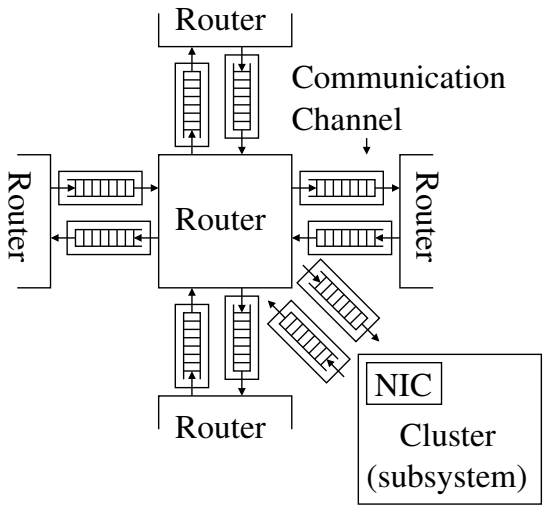


Figure 1.1 – A generic 2D-Mesh NoC structure.

- **Communication channel** : is a set of unidirectional buffering devices, connecting an output port and an input port of the neighboring router/router or router/NIC. Its main task is to flow the packets. A communication channel might travel clock boundary, when the NoC supports GALS (Globally Asynchronous, Locally Synchronous) approach.
- **Network Interface Controller (NIC)** : is a gateway between network and each local subsystem (called a cluster). It provides the services at the transport layer on the ISO-OSI reference model. Its main tasks are the protocol conversion and the packet building.

In the context of fault-tolerance, among these three basic components, the NIC is normally considered as a component of cluster, rather than a NoC component. Because a faulty NIC doesn't modify the network topology, but leads a cluster unavailable. The router and the communication channel are considered as two atomic components of NoC.

For the ODDR ("On the field" Detection, De-activation & Reconfiguration) mechanism, once a router or a communication channel is detected as faulty, it must be fully de-activated. Then, all faulty/de-activated components must be diagnosed/located to identify the modification of topology. And the routing function must be reconfigured to support this modified topology.

To realize such mechanism, we must solve three problems :

1. **"NoC Test"** : We must define an appropriate test strategy, which must be able to detect and de-activate all faulty components (routers and channels) of NoC. In order to fit "on the field" requirement, namely, to recover (permanent) failures happening when the chip is mounted in the final equipment, the test mechanism must be

embedded on chip.

2. **"Configuration Infrastructure"** : We must define a robust global configuration master. And we must make this master able to diagnose/locate the faulty/fault-free components, and able to identify the modification of topology, and finally able to calculate the configuration information. Besides, we must also define a robust global configuration bus for the master, in order to distribute the configuration information to the corresponding NoC components.
3. **"Routing Algorithm"** : We must define a deadlock-free, livelock-free and reconfigurable routing algorithm to support any modification of topology. And we must also implement this routing algorithm in NoC.

In the following section 1.2, 1.3 and 1.4, we analyze and discuss these three problems. And we conclude and list the questions at the end of each section. These questions are answered in the thesis "Conclusion".

1.3 NoC Test

In the semiconductor industry, the classical test approach aims to exactly determine which chip is faulty in **early** stage of the manufacturing process. Earlier detection consumes lower cost (for example, a typical incremental ratio of detecting a fault at the various manufacture levels is **10**, as presented in [Williams and PARKER, 1982]).

	Manufacturing Process	Cost of detecting a fault ($)
Low	Wafer	0.01-0.10
⇓	Packaged chip	0.1-1
⇓	Board	1-10
⇓	System	10-100
High	Field	100-1000

Once a chip is detected as defective, it will be thrown away as a garbage. This solution is acceptable when the die yield per wafer is high ($> 70\%$). But, for a modern high-density chip, shrinking transistor geometry and increasing transistor quantity per single die [ITRS, 2009], could result in a low yield. In order to improve this yield, the manufacture defect must be detected and located. A faulty chip can therefore be (statically) reconfigured to work in a degraded mode, and be sold at a lower cost.

However, facing a future high-permanent-fault-rate MP^{2}SoC chip, not only the manufacture defects, but also the "on the field" failures (where the chip is already mounted in the final equipment), must be handled. Thus, "test for de-activation" strategy must be used,

to isolate the failure propagation. For the NoC itself, the "test for de-activation" strategy aiming to detect and de-activate all faulty components, must be defined and realized. The related problems are analyzed and discussed in this section.

1.3.1 BIST (Built-In Self-Test)

In order to detect the faults both in the manufacture and "on the field", a BIST (built-in self-test) strategy is the unique solution, since we cannot use an external tester. Thus, the test generator/analyzer are embedded on a chip, so that the test process can be operated at any slice of chip lifetime. The BIST strategies can be split into 2 classes :

– On-Line : Testing occurs during normal functional operating conditions.

– Off-Line : Deals with testing a system when it is not in its normal mode of operation.

As a permanent fault physically and globally changes the NoC structure and topology, "off-line BIST" techniques must be adopted, where the components can be de-activated as soon as they have been detected as faulty, when the NoC is not yet running. In this thesis, we make the test process systematically operated at each system reboot or chip power-on.

To define/realize such off-line BIST for NoC, the first task is to place each embedded test generator/analyzer at reasonable position.

1.3.2 Placement of Embedded Test Generator/Analyzer

A Network-on-Chip is composed of a set of atomic components : routers and channels. These two kinds of component connect with each other through the wire-links. We must insert the multiplexers to break the original connections, as shown in Figure 1.2, in order to isolate each component test, and then to de-activate the faulty ones.

With the help of these multiplexers, we can place the BIST modules : embedded test generator/analyzer, to test each component. As shown in Figure 1.3, there are in fact two possible placements : non-interactional and interactional.

Figure 1.3.{A} presents non-interactional placement. With this placement, the router test and the channel test do not interfere. Thus, each component can be tested in parallel and in isolation without any coordination. However, this placement implies that some data-paths - the inputs of the multiplexers - can not be tested, namely, the multiplexers can not be fully tested.

Figure 1.3.{B} presents interactional placement. With this placement, all data-paths can be tested. However, the router test and the channel test mutually interfere.

In this thesis, we show more interest in the interactional placement, because it can provide a complete data-paths test. With this placement, we must develop an algorithm

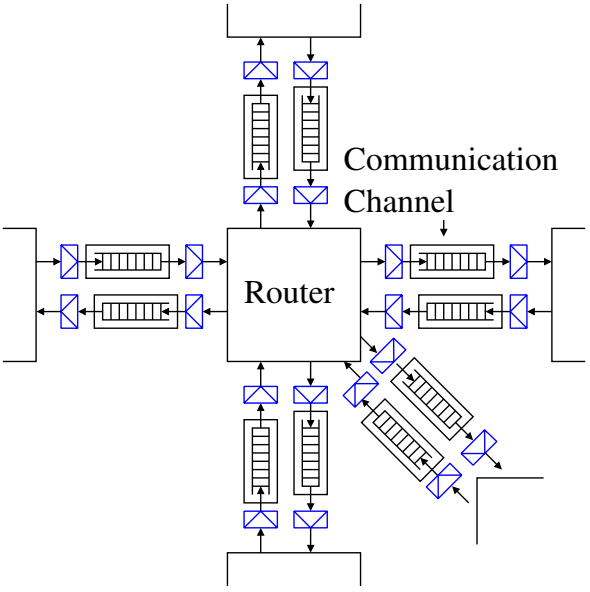


Figure 1.2 – The inserted multiplexers.

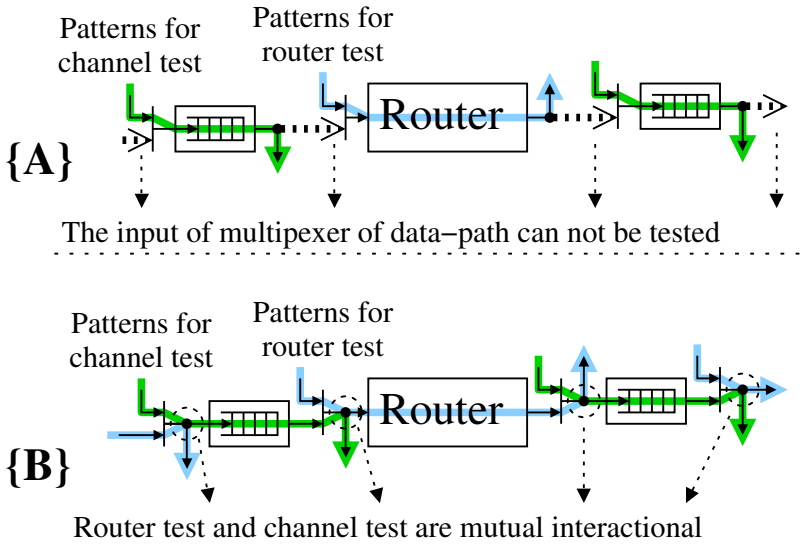


Figure 1.3 – Two placements of BIST modules.

for test process to handle the test interaction. Besides, we must solve the second problem of NoC test, that is to generate a set of efficient test patterns for router test and channel test.

1.3.3 Test Pattern, Fault Model & Fault Coverage

As shown in Figure 1.4, used by a couple of test generator/analyzer to detect a circuit, the basic method is to assign an input value on the inputs ports, at a given time period, and to observe the output value on the outputs ports. Such a group of assigning input signals,

observing output signals and a given time period, is defined as a **test pattern**.

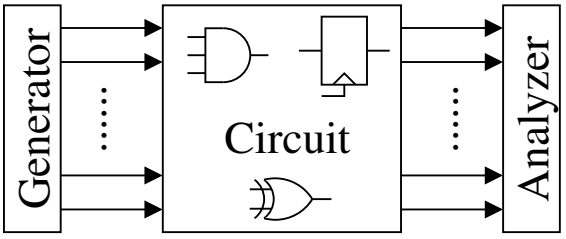


Figure 1.4 – A generic circuit is tested by a couple of test generator/analyzer.

The quality of a test pattern is evaluated from point of view of fault coverage on a fault model. In this thesis, the target fault model is SAF (stuck-at fault).

Stuck-At Fault : The faults most frequently occur due to gate oxide or metal wire short with GND or Vdd, resulting in a signal or a gate in/output port stuck at a 0 or 1 value.

The definition of the SAF coverage is :

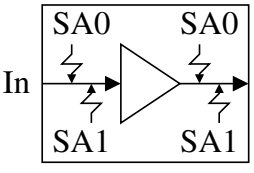
$$\text{Fault Coverage } (\%) \quad = \quad \frac{\text{Number of SAF Detected}}{\text{Total Number of SAF}}$$

In this formula :

"Total Number of SAF" is calculated from the total number of gate port of the target circuit.

"Number of SAF Detected" comes from the number of SAF detected through a given test pattern.

For example, as shown in Figure 1.5, for a circuit with an inverter gate, the total number of gate port is 2 (In and Out). Each port can be injected by SA0 or SA1, "Total Number of SAF" is thus $4 = 2 \times 2$. And we can list all of SAFs. For a given test pattern (In :0, Out :1), two SAFs can be detected, "Number of SAF Detected" is thus 2. "Fault Coverage" reached by this test pattern is $50\% = 2/4$.



SAF LIST:	A test pattern (In:0,Out:1):
In_SA0	can detect 2 SAFs
In_SA1	In_SA1
Out_SA0	Out_SA0
Out_SA1	

Figure 1.5 – A circuit with an inverse gate.

A higher fault coverage means that more SAF can be detected. To get a high fault coverage becomes the main goal at test pattern generation stage. However, to reach this goal, it normally needs a larger number of test patterns, which will lead to consume more test times and more silicon area overhead (when the test patterns are directly embedded on chip). So, to get the balance between the number of test patterns and the fault coverage is an issue.

In this thesis, we want to obtain the highest possible fault coverage, limiting the test time less than 1 second at $1Ghz$ clock and the silicon area overhead less than 200% (the overhead due to TMR : Triple Modular Redundancy) of the total NoC silicon footprint. To meet these big challenges, we must define an efficient test pattern generation.

1.3.4 Test Pattern Generation

At the beginning of this discussion, we introduce the classification of circuit : combinational and sequential.

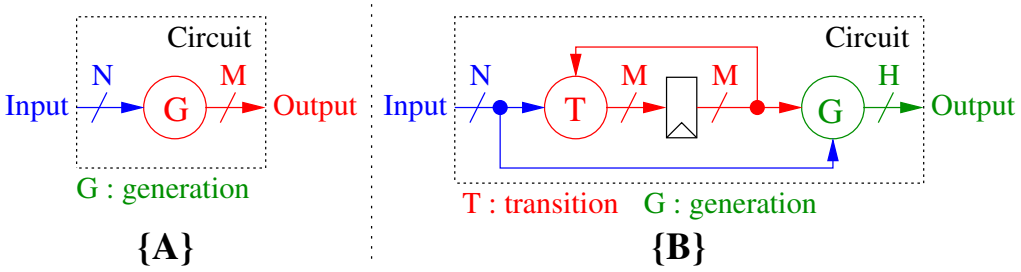


Figure 1.6 – The combinational circuit and the sequential circuit.

Combinational Circuit consists of logic gates whose outputs at any time are determined by combining the values of the applied inputs using logic operations. In a NoC, the routing function module of the router is a combinational circuit.

As shown in Figure 1.6.{A}, for a combinational circuit with N bits input, the total number of circuit states is 2^N. And each circuit state can be directly tested through assigning the input and observing the output.

Sequential Circuit consists of some combinational circuits and storage elements. The output depends not only on the presented input but also on the contents of storage elements. In a NoC, the channel and the switch module (such as crossbar) of the router are sequential circuits.

As shown in Figure 1.6.{B}, for a sequential circuit with N bits input and M bits register, the total number of circuit state is $2^{N+M} = 2^N \times 2^M$. To test each circuit state, we need at least 2 clock cycles for 2 input assignments : one clock cycle for register

initialization with first input assignment ; the other clock cycle for output observation with second input assignment. That is to say, two test patterns work together to test one circuit state.

To test a circuit using BIST strategy, three classical test pattern generations can be used : pseudo-random test pattern generation, automatic test pattern generation and ad-hoc test pattern generation.

Pseudo-random Test Pattern Generation produces a set of test pattern with an organized & random process, such as LFSR (Linear Feedback Shift Register). LFSR is a shift register whose input bit is a linear function of its previous state. For the combinational circuit shown in Figure 1.6, LFSR (with N bits register) can produce $(2^N - 1)$ continuous and unduplicated test patterns to detect the corresponding circuit state. Moreover, cooperating with a signature computation, such as MISR (Multiple Input Signature Register), they (LFSR/MISR) can directly play the role of BIST generator&analyzer.

However, when N is a big number, we can not not easily get the balance between the number of test pattern and the fault coverage with the LFSR/MISR couple. This weak point will be very obvious, when we use LFSR/MISR to handle a sequential circuit. In order to deal with this problem, we can use multi-reseeding reinitialization for the LFSR. But, this method breaks the linearity feature of LFSR, leading to high complexity of test control and high hardware overhead. Therefore, this method doesn't interests us.

Multi-reseeding reinitialization for the LFSR : To test a circuit, one seed of LFSR might lead to a good fault coverage increase, such as 5%. However, the following continuous states of LFSR might be inefficiency, the fault coverage increase might stay at 5% for a long time. So, LFSR will need many times of reinitializations to increase quickly the fault coverage. But, this solution breaks the linearity feature of LFSR, leading to high complexity of test control.

Automatic Test Pattern Generation can produce directly high fault coverage with number limited test patterns for combinational circuits. This test pattern generation is done using the structure analysis of the circuit with the help of an algorithm, suck as Roth's D-Algorithm [Paul, 1966]. Therefore, many industrial ATPG (Automatic Test Pattern Generation) tools have been launched on the market. In this thesis, we will use Synopsis TetraMAX, to generate directly the test patterns for the routing function module of router.

It should be noted that, the ATPG tool can not directly deal with a complex sequential circuit, as the controllability and observability of most nodes of circuit are very low.

– **Controllability** of an internal circuit node presents the complexity of setting a node to a 1 or 0 value through the circuit input.
– **Observability** of a particular circuit node describes the complexity of observation of the node state from the outputs of this circuit.

As shown in Figure 1.7, a complex sequential circuit means a large depth of register level, and the node **a** means an internal single node in the middle of the logic level. If it's difficult to be controlled by the input and to be observed through the output, its controllability & observability is low. For such circuit, ATPG tools get very low fault coverage.

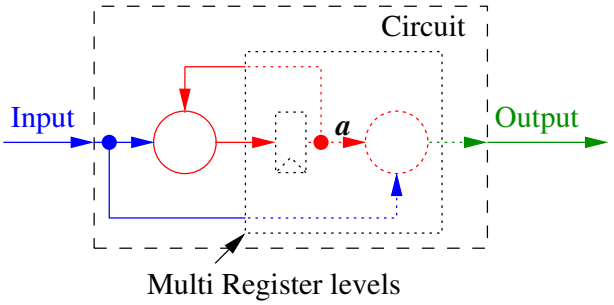


Figure 1.7 – A complex sequential circuit with multi-register-levels.

In fact, ATPG tools are useful when the techniques improving the observability and controllability are implemented, such as the scan-path approach.

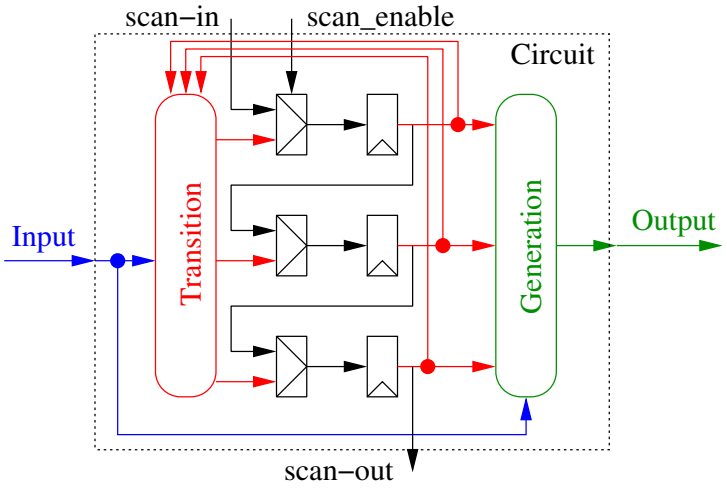


Figure 1.8 – Scan-based approach.

As shown in Figure 1.8, the registers of circuit are connected as a long-chain shift register called scan-path. The circuit can thus operate in one of two modes : the normal mode and the scan mode. When to process test, the circuit works firstly in scan mode (M clock cycles, a M bits data are shifted in the registers as the initialization state). Then the circuit works in normal mode one clock cycle, the circuit produces the register value and the output value. Finally, the circuit works again in scan mode (M clock cycles, so the M bits register value are shifted out and analyzed). In other words, the scan-path allows to split a complex sequential circuit into a set of combinational parts (for example, the transition part and the generation part as shown in Figure 1.8). Each part can be easily handled by ATPG tool.

However, using the scan-path, the time consuming will be very huge caused by shifting-in, shifting-out. Therefore, this approach is not used in this thesis.

Ad-hoc Test Pattern Generation corresponds to explicitly generate the test patterns, considering the characteristic of component function and structure. This method can get a high fault coverage, and can limit the number of test pattern. Taking account of these advantages, we propose in this thesis, an ad-hoc test pattern generation for channel test and the router switch module (crossbar) test. The detail will be presented in chapter 5.

In the last two subsections, we will analyze and discuss two constraints of "NoC test" : GALS and de-activated component behavior.

1.3.5 GALS

The GALS (Globally Asynchronous, Locally Synchronous) approach can solve the design bottleneck of clock tree distribution, by partitioning a large circuit in small synchronous parts communicating asynchronously. And most Network-on-Chip designs support the GALS on the communication channel. Thus, the channel test procedure must be able to handle the clock boundaries.

1.3.6 De-activated Component Behavior

The faulty components must be de-activated as soon as they have been detected as faulty, with the purpose of isolating error source and avoiding failure propagation. This de-activation approach must be implemented together with the NoC BIST. Moreover, the de-activated component should not disturb the normal function of fault-free components. To meet this requirement, in this thesis, the de-activation component is proposed to be configured to behave as a "Black Hole", which discards any incoming data, and produces no outgoing data.

1.3.7 List of questions

In summary, the question related to the embedded test procedure are the following :

- How to handle the interaction between the router test and the channel test ?
- How to generate efficient test patterns for channel test and router switch module (crossbar) test ?
- How to realize the de-activation approach within the BIST strategy ? And How to configure the fault component to behave as a "Black Hole" ?
- Can the proposed communication channel test handle the clock boundaries ? And How ?
- How many test patterns are generated for the NoC test ?
- What is the cost of the NoC test ?
- What is the SAF coverage respectively reached by the channel test and the router test ? What is the SAF coverage reached by the whole network (with the BIST circuit) test ?

1.4 Configuration Infrastructure

To support the "on the field" NoC reconfiguration mechanism in a MP^{2}SoC architecture, the configuration infrastructure must be implemented in the architecture. The specific responsibilities of such infrastructure are :

- To define a robust global configuration master. This master is in charge of the control of all configuration actions.
- To define a method for the master, to diagnose/locate the faulty/fault-free components detected by the BIST.
- To define a robust global configuration bus. This bus is used by the master to distribute the configuration information to the NoC components.

In the section, we analyze and discuss each responsibility.

1.4.1 Configuration Master

Considering the context of "on the field" NoC reconfiguration mechanism, the global configuration master must be embedded on the chip. Thus, there are two methods to define a master : to design an appropriative embedded component or to reuse an embedded processor core of MP^{2}SoC. In this thesis, we prefer the second method, because it improves the fault-tolerance : any core can be used as the global configuration master. However, we have to solve the problem produced by the use of this method : how to select an embed-

ded processor core to play the role of global configuration master. In fact, we can use two approaches of selection : static selection and dynamic selection.

- **Static selection** fixes a processor core to play the master role. However, this method is not "robust" when the fixed processor core is faulty.
- **Dynamic selection** elects core respecting "free competition" principle. This selection method allows to meet "the robust" requirement.

Once the global configuration master problem has be solved, the next one is to develop a mechanism of diagnosis/localization of faulty/fault-free components operating in the master.

1.4.2 Diagnosis/Localization of Faulty/Fault-free Component

One of the tasks of the global configuration master is to process the NoC test results, namely, to diagnose/locate the faulty/fault-free components, with purpose of identifying the modification of topology and calculating the configuration information. The practicable methods are : direct diagnosis/localization or indirect diagnosis/localization.

- **Direct Diagnosis/Localization** method reads or accesses directly the devices storing the NoC test results. Therefore, there are two requirements implied in this method. Firstly, the NoC test results must be locally stored in some distributed memory devices. Secondly, an appropriative TAM (test access mechanism) must be built on the chip to access these memory devices. These two requirements will result in more hardware overhead and more risk of failures. So it is not proposed in this thesis.
- **Indirect Diagnosis/Localization** method is to find out the faulty/fault-free components, through a functional test. According to the NoC test strategy, any faulty component will be de-activated and be configured to behave as a "Black Hole" (discards any incoming data, and produces no outgoing data). This strategy is actually a functional fault model, that can be located by a functional test, such as the solutions proposed in [Grecu et al., 2005, Stewart and Tragoudas, 2006, Raik et al., 2007]. It should be noted that, the functional test can be realized using a software application. It will be executed by the configuration master.

Once the faulty/fault-free components have been diagnosed/located by the master, the modification of topology has been identified, the configuration information can be calculated according to the new routing algorithm. So, we have to define a robust global configuration bus for the master, in order to distribute the information to the configuration registers, which is the next problem to be solved.

1.4.3 Configuration Bus

First of all, the proposed global configuration master is a not yet defined embedded processor core. Secondly, the NoC itself already connects each processor core and each configuration memory device. Considering these two characteristics, we propose to reuse NoC itself to play the role of configuration bus. For example, using the remaining fault-free components to propagate the messages (such as the flooding broadcast propagation proposed in [Dumitraş et al., 2003]), the NoC can implement a function configuration bus. This method is robust and it doesn't induce any hardware overhead and any risk of failure.

1.4.4 List of questions

This thesis will use an existing configuration infrastructure solution : the DCCI (Distributed Cooperative Configuration Infrastructure) [Zhang et al., 2011]. DCCI is developed by REFAUVELET Dimitri at the LIP6 laboratory. This DCCI dynamically selects an embedded processor core to play the role of the global configuration master, and it reuses the NoC itself to realize a functional global configuration bus. And we aims to develop on top of DCCI, a functional test software to diagnose/locate faulty/fault-free component.

The questions addressed in this topic are :

- Can the test software 100% diagnose/locate faulty/disabled component ?
- Can the test software 100% diagnose/locate the fault-free component ?
- What is the performance of the test software ?

1.5 Routing Algorithm

The routing algorithm is in charge of guiding packet transmission. It must route any packet to the destination from the source in a given topology without deadlock and livelock. However, when the topology is modified caused by some faulty/disabled routers or channels, the routing algorithm must be reconfigured to adapt the modification of topology. In this section, the problem of defining a fault-tolerant reconfigurable routing algorithm is discussed and analyzed.

1.5.1 Micro-network Regular Topology

As shown in Figure 1.9, the typical regular topologies are *Linear, Ring, Fat tree, Mesh, Torus, Octagon and Butterfly*, each one defines a static arrangement of the communication

channels, the routers and the clusters (a couple of go/back channels are presented by a line, a channel is presented by an arrow line, a router by a square node and a cluster by a circle).

For a given regular topology, the number of connections per router and the total number of channels are determined, and these numbers will be utilized by the fault-tolerant reconfigurable routing algorithm, to bypass the faulty components.

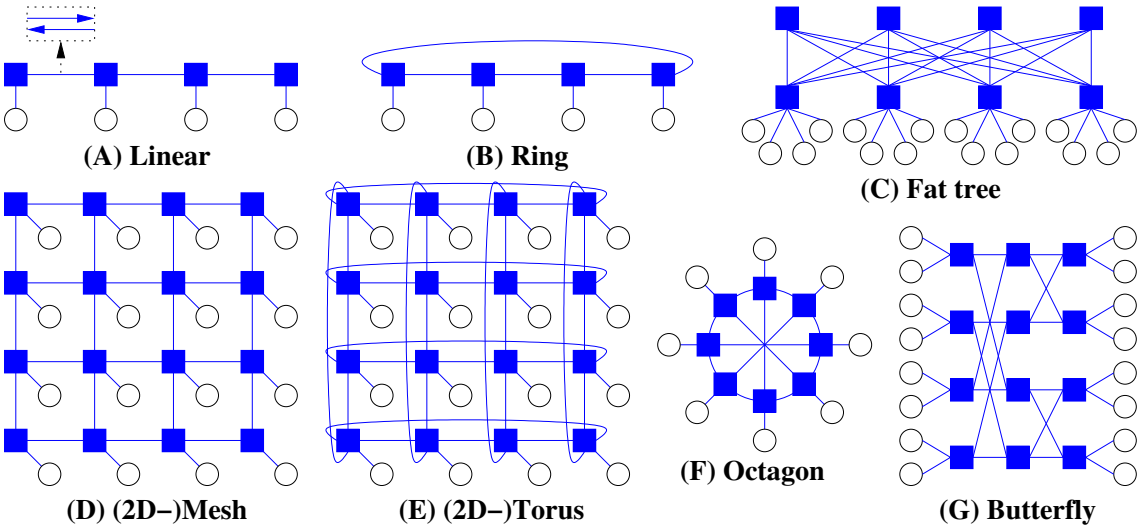


Figure 1.9 – A summary of typical regular topology.

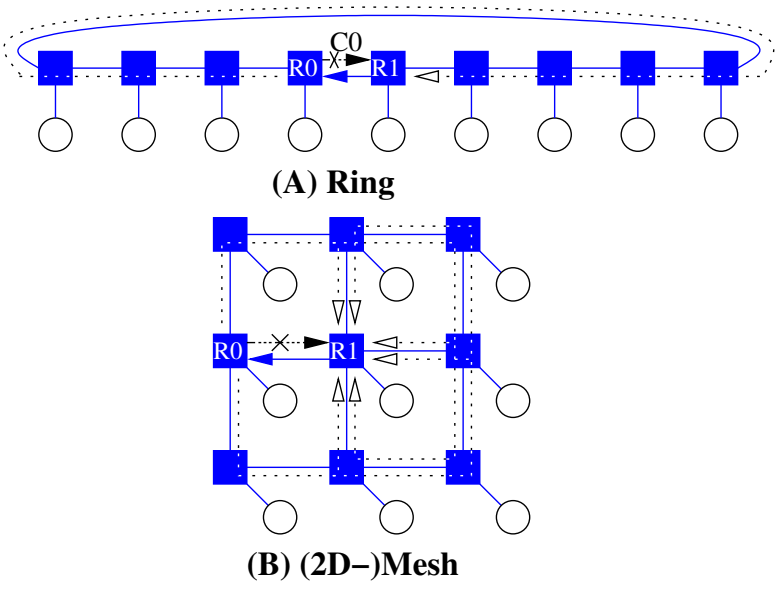


Figure 1.10 – A 9-nodes Ring topology and 3×3-nodes 2D-Mesh topology.

As shown in Figure 1.10.A, in a 9-nodes Ring topology, there are two paths from R0 to R1. Once the communication channel C0 is broken, only one path can be used to route the packet from R0 to R1 for any routing algorithm

to bypass the faulty channel. As shown in Figure 1.10.B, in a 3×3-nodes 2D-Mesh topology, there are seven paths between routers R0 and R1. Even though the channel C0 is broken, six paths can still be used by a routing algorithm to bypass the faulty channel.

In this thesis, we are focussing on 2D-Mesh micro topologies.

1.5.2 Fault Model of Modification of Topology

In the regular 2D-Mesh, the modification of topology can be classified into five fault models :

- **Single** faulty channel topology.
- **Multiple** faulty channels topology.
- **Single** faulty router topology.
- **Multiple** faulty routers topology.
- **Mixed** Single or Multiple faulty channel and router topology.

These five fault models must be handled by the fault-tolerant reconfigurable routing algorithm, without deadlock or livelock. It should be noted that, handling all single-faulty-router topologies is the minimum requirement for a fault-tolerant routing algorithm.

In the following, we will present the routing strategy and analyze and discuss deadlock/livelock problems.

1.5.3 Routing Strategies

The routing strategies can be classified in two types : deterministic and adaptive. According to a deterministic routing algorithm, all the packets of a pair of {source, destination} will follow the same fixed path. It guaranties the in-order delivery property. Oppositely, in an adaptive routing, the path of a pair of {source, destination} can be dynamically modified for various causes (router congestion, channel busy, channel failure and so on). It results in an out-of-order delivery.

For a deterministic routing, once a fixed path is broken, the reconfiguration mechanism must redefine a new one to replace the broken one. For an adaptive routing, the reconfiguration mechanism must mark which channel and/or router as failed or as unusable (not only the faulty components must be marked, but also some fault-free might need to be marked as unusable for some packets, so as to avoid the livelock or deadlock).

As the adaptive routing has more flexibility than the deterministic routing, it's used in many published solutions. However, the adaptive routing causes a higher overhead of

silicon area.[2] Moreover, the loss of the in-order delivery property introduces big difficulties in higher level communication protocols. Therefore, to design a deterministic, lightweight, deadlock-free, fault-tolerant and reconfigurable routing algorithm is the goal of our work.

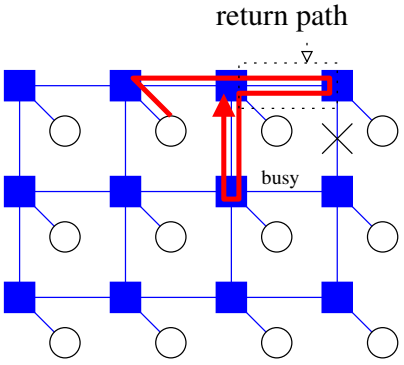
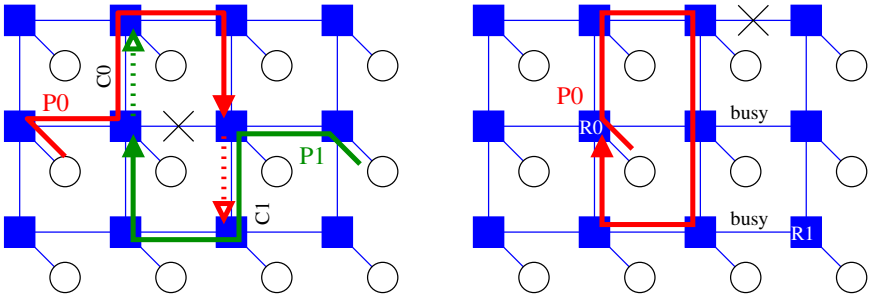


Figure 1.11 – A return-back path in a 3×4-nodes 2D-Mesh topology. It induces a higher deadlock or livelock risk.

1.5.4 Deadlock & Livelock

For any routing algorithm, both deadlock and livelock must be avoided. They are occasioned by incompatible allocation/deallocation conditions, presented as :



(A) Deadlock (B) Livelock

Figure 1.12 – A deadlock and a livelock.

– **Deadlock** is caused by two or more packets waiting for each other to release a network resource, such as a communication channel. As shown in Figure 1.12.A, in a 2D-Mesh topology, the packets P0 and P1, waiting for each other to release the channels C1 and C0, induces a deadlock situation. To handle this problem, two methods can be used : deadlock prevention and deadlock recovery.

2. Firstly, the adaptive routing requires a reordering approach at the destination. Secondly, to treat some irregular topologies, to avoid to route a packet in a dead end, the adaptive routing permits the return-back path (as shown in Figure 1.11) in NoC, that induces a higher deadlock or livelock risk. Finally, some adaptive routing implementations depend on the timeout & retransmission approach.

- **Deadlock prevention** avoids deadlock by source restriction of the routing function. For example, the turn model [Glass and Ni, 1994] forbids some turns on a 2D-Mesh to break the cycle in the channel dependency graph.
- **Deadlock recovery** introduces some special resources to break the deadlock situation, when deadlock appears. For example, in Figure 1.12.A, we can suspend the packet P0, until the end of transmission of P1, then release P0, so to recover deadlock.

– **Livelock** occurs only in the adaptive routing strategy. Unlike deadlock, livelocked packets continue to move through the network, but never reach their destination. For example, one packet enters a cycle situation, as shown in Figure 1.12.B, in a 2D-Mesh topology, if some channels are very busy due to some applications, the packet P0 (from R0 to R1) might keep moving, entering an infinite periodic loop.

A reconfigurable routing algorithm must be formally proven to be deadlock-free and livelock-free, since the modifications of topology are irregular and numerous. According to the minimum requirement, the reconfigurable routing algorithm must be formally proven to be deadlock-free and livelock-free with all single-faulty-router topologies [3].

In the following subsections, we will analyze and discuss two issues of fault-tolerant reconfigurable routing algorithm : transmission mode and virtual channels.

1.5.5 Transmission Mode

[Bogdan et al., 2007, Song et al., 2009, Pirretti et al., 2004] proposed a fault-tolerant approach for NoC : changing NoC transmission mode from unicast to multicast, broadcasting a packet to increase the number of packet copies, and then making these copies travel all possible route paths, finally guarantying that at least one copy of packet can reach at the destination. This method provides a shortest or fastest route path and good tolerance against faults. But, it will induce an extensive power consumption, a low transmission efficiency, a high network latency and a complex network structure. In this thesis, we keep the NoC original transmission mode (unicast).

1.5.6 Virtual Channels

As presented in the published techniques [Duato, 1993, Chalasani and Boppana, 1995a, Park et al., 2000], virtual channels can be used to increase the degree of fault-tolerance of NoC, or to realize deadlock recovery for the routing algorithm. But, the virtual channels

3. We can not prove deadlock-free and livelock-free properties using some cases of topology modifications, counter-example might exist, such as the history happened between the articles [Chen and Chiu, 1998] and [Holsmark and Kumar, 2007].

causes silicon area overhead and they are not supported by all NoCs. In this thesis, we keep the NoC original channel property (non virtual channel).

In the following subsections, we will analyze and discuss the problem addressing the routing algorithm implementation.

1.5.7 Routing Function

The final goal of a fault-tolerant reconfigurable routing algorithm design, is to implement the routing strategy in the NoC. Thus, in each router, the routing function must be defined. It determines an output for a packet depending on some parameters, respecting the routing algorithm.

$$INDEX_{\text{output}} \quad = \quad Routing\_Function \quad (parameters)$$

The typical parameters and informations are :

- $INDEX_{\text{source}}$ is the absolute coordinate of packet source in the given topology.
- $INDEX_{\text{destination}}$ is the absolute coordinates of packet destination in the given topology.
- $INDEX_{\text{router}}$ is the absolute coordinates of the router routing the packet.
- $INDEX_{\text{input}}$ is the ID of the router input routing the packet.
- $PRIORITY_{\text{message}}$ describes the priority level of the packet.
- $CONFIG_{\text{router}}$ describes the informations of configuration of current router routing the packet.

The number of parameters normally determines the implementation cost of the routing function : less parameter, less cost (area).

1.5.8 Routing Function Implementation Method

The methods to implement a routing function can be classified in two types :

- **Table-based** : The routing function hardware implementation is a table. The index of table is a group of parameters, such as {$INDEX_{\text{source}}$, $INDEX_{\text{destination}}$, ..., $CONFIG_{\text{router}}$ and so on}. The configuration memory is just the routing table itself. To be reconfigurable, this method requires to implement the routing table as a read/write memory.
- **Logic-block-based** : The routing function hardware implementation is synthesized directly from the routing function HDL description model. To be reconfigurable, this method needs only an additional configuration register.

For a fault-tolerant reconfigurable routing algorithm, the hardware cost due to logic-block-based method is very smaller than the table-based method. Thus, we will use the logic-block-based method in this thesis.

1.5.9 Routing Evaluation

A fault-tolerant reconfigurable routing algorithm must be evaluated from at least two points :

- **Performance** evaluation aims to find out the impact or penalty on the network transmission performance.
- **Silicon area overhead** evaluation shows the cost of routing function hardware implementation.

1.5.10 List of questions

The questions addressed in this topic are :

- Which fault model of modification of topology can be handled by the proposed routing algorithm ? And can all of single-faulty-router topologies be handled ?
- Can we formally prove that the proposed fault-tolerant reconfigurable routing algorithm is deadlock-free and livelock-free for all of single-faulty-router topologies ?
- What is the impact or penalty on the network transmission performance ?
- What is the cost on the silicon area (comparing with X-First routing algorithm implementation) ?
- Is it possible to generalize the proposed fault-tolerant reconfigurable routing algorithm to handle other topologies than the 2D-Mesh ?

Chapter 2

State of the Art

Contents

In this chapter, we present the state of the art of the Network-on-Chip fault-tolerant strategies dealing with the permanent fault family. These strategies are defined at different levels : link-level, router-level and network-level. And we discuss and analyze the researches related to ODDR ("On the field" Detection, Deactivation & Reconfiguration) mechanism. This discussion & analysis cover three topics : the NoC test strategy, the configuration infrastructure and the routing algorithm.

2.1 A Summary of NoC Fault-Tolerant Strategies Handling the Permanent Fault Family

A Network-on-Chip, is an embedded interconnection system. It is an inheritance design of the networks used in the high-end supercomputers or in the telecom switches. As shown in Figure 2.1, a typical NoC structure, is a set of monolithic centralized routers connected each other through the wire-link channels. Researchers have defined and published fault-tolerant strategies at link-level, at router-level and at network-level, to handle

the permanent fault family. In this section, we summarize these strategies.

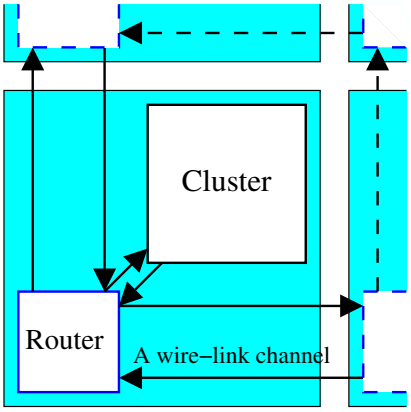


Figure 2.1 – A generic network consists of the routers and the wire-link channels.

2.1.1 Link-level

At link-level, the main idea of fault-tolerant strategy is to de-activate the faulty wire-links in a channel, and to transmit the packet through the remaining links. Two solutions are presented in the book [Dally and Towles, 2004] : **Spare Bit** or **Graceful Degradation**.

Spare Bit

This method relies on additional extra wire-links, to provide a hardware redundancy. As shown in Figure 2.2, one spare bit is added into a N bits channel, resulting in a N+1 bits channel, that is used to transmit the N bits packet. When a bit is detected as faulty, the remaining N bits are reconfigured to repair the packet transmission. However, this method can not handle the case where the number of faulty bits exceeds the number of spare bits.

Graceful Degradation

This method doesn't add any extra wire-link, but uses the remaining links to recover the packet transmission. For a N bits channel, when M ($M < N$) links are detected as faulty, the remaining (N-M) bits (or less) will be used to transmit the N bits packet, with the help of serial repackaging technique [Pasca et al., 2010]. However, the bandwidth of the faulty channel will be reduced, which decreases the channel transmission efficiency and increases the network latency.

The wire-link channel test is very straightforward. With the help of some effective test patterns, such as the patterns described in [Lien and Breuer, 1991, Jarwala and Yau, 1989, Shi and Fuchs, 1995, Park, 1996], the test generator at the head of channel, cooperating with the test analyzer at the end of chanel, can detect and diagnose the faulty wire-links.

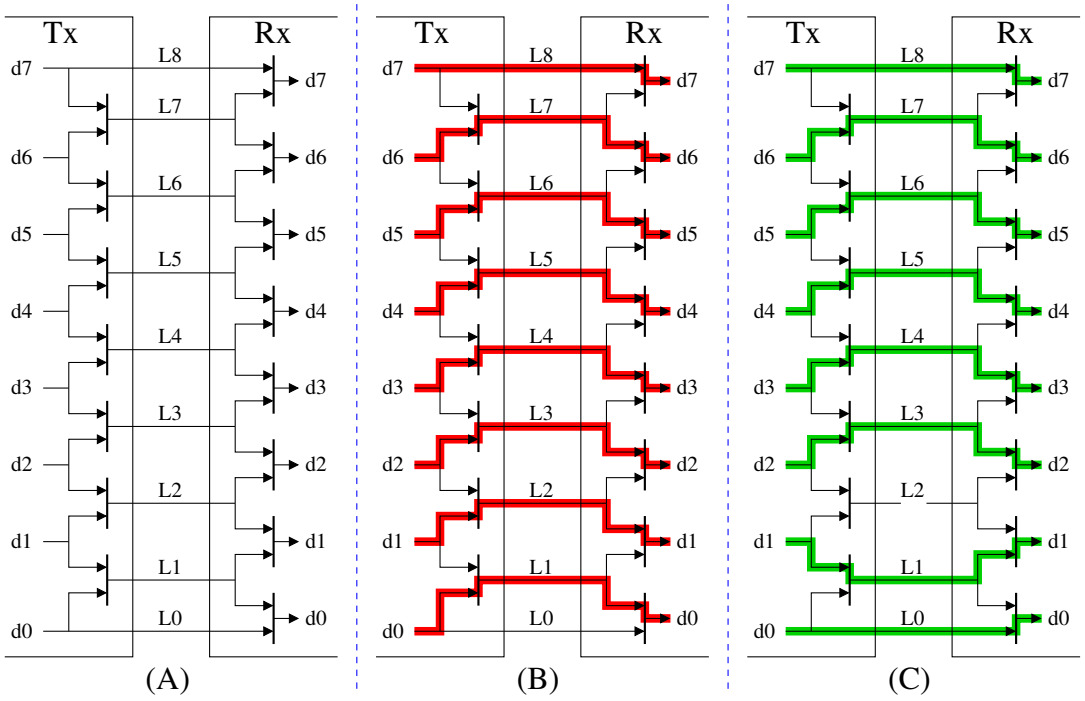


Figure 2.2 – FIG.A presents a wire-link channel containing 8 bits + one spare bit. This channel is used to transmit 8 bits data. FIG.B presents a configuration for the no-faulty link case. FIG.C presents a configuration for an single-faulty link (L2) case.

2.1.2 Router-level

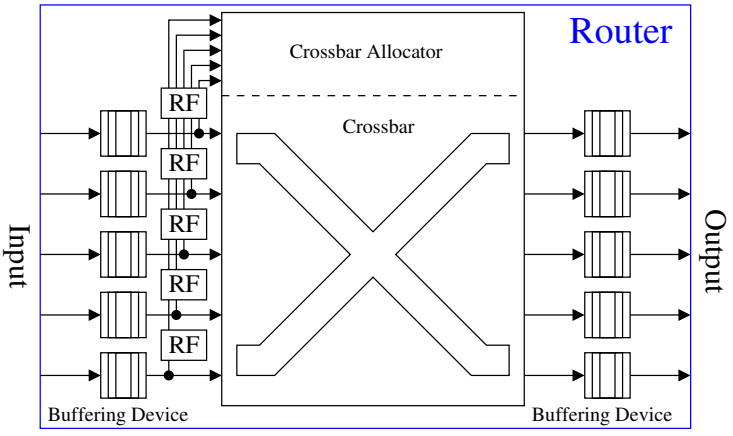


Figure 2.3 – A generic router of 2D-Mesh NoC.

As shown in Figure 2.3, a generic router consists of three separate modules : routing function (RF), crossbar and input/output buffering device. The fault-tolerant strategies defined at router-level, are presented in this section. The related solutions are **N-modular Redundancy**, **Crossbar Bypass Bus** and **CRC (Cyclic Redundancy Codes) or ECC (Error Correction Codes)**.

NMR (N-Modular Redundancy)

Following a traditional hardware redundancy idea, [Constantinides et al., 2006] proposed a NMR (N-Modular Redundancy) fault-tolerant solution for a router. This solution adopts N-copies of router modules to generate N outputs in parallel, then to vote one output as the final one, so as to guaranty the reliability. But, this solution is very expensive in terms of area overhead.

Crossbar Bypass Bus

[Fick et al., 2009b] suggested to provide a parallel bus beside the crossbar, as shown in Figure 2.4, to transmit the packet bypassing the faulty crossbar, in order to rebuild the transmission from the inputs to the outputs. This solution uses a lightweight hardware redundancy, but introduces a strong reduction of the bandwidth as well as a limitation on clock frequency.

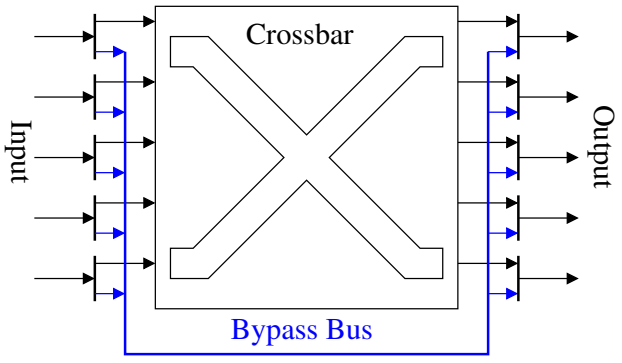


Figure 2.4 – A crossbar and a bypass bus.

CRC (Cyclic Redundancy Codes) or ECC (Error Correction Codes)

[Kohler and Radetzki, 2009, Fick et al., 2009b] proposed to adopt CRC or ECC to handle the failures in the data-paths of the crossbar or the buffering devices.

- **CRC** empower each packet to self-check the bit-error.
- **ECC** empower each packet to self-check, self-diagnose and self-correct a limited number of bit-error. However, this limited number depends on the code used. For example, the Hamming code can detect and correct a single-bit error. In order to handle more bit-errors, complex codes must be used, but this leads to a very large hardware cost.

As shown in Figure 2.5, the CRC/ECC decoding modules are built at each output port of the crossbar or the buffering device, to check each packet. Once one of the packets can

not pass the check, the corresponding component is considered as faulty.

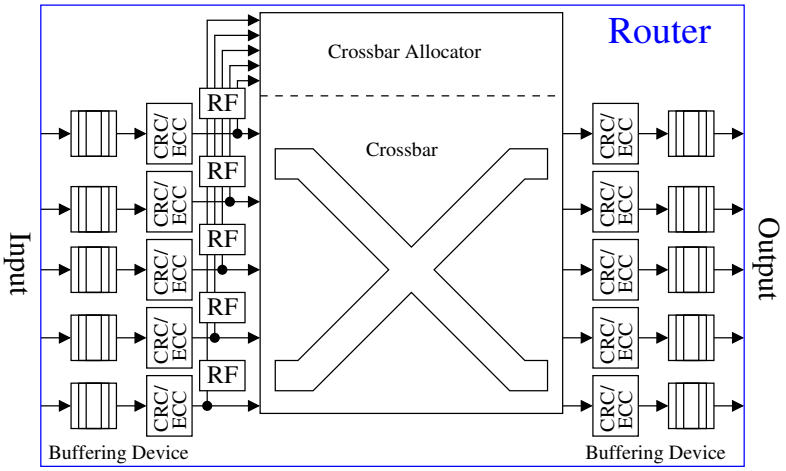


Figure 2.5 – A router with CRC/ECC modules.

In fact, CRC/ECC can only handle the packet-level errors, rather than other reliability failures. For example, the crossbar blocking, caused by a corrupted flow control, is a very dangerous failure, resulting in a global network blocking. Thus, the dedicated BIST must be used to deal with these failures.

Dedicated BIST Design

In [Fick et al., 2009b], besides using CRC/ECC to test the data-paths of the crossbar or the buffering devices, the dedicated BIST is proposed to detect the controller of the crossbar and the buffering devices. However, this BIST doesn't support the GALS approach that is very useful in modern NoC design. Moreover, without isolation or deactivation mechanism, the failure propagation can not be stopped. Thus, this BIST can not fit the "on the field" fault-tolerance.

[Lin et al., 2009] proposed an off-line, "on the field", dedicated BIST to test, diagnose and isolate the faulty buffering devices & the faulty multiplexers of crossbar. However, this BIST can test/diagnose/isolate neither the whole crossbar nor the whole channel. It is not a complete BIST solution for NoC itself.

2.1.3 Network-level

At link or router-level, the permanent fault-tolerant strategies aim to recover the function of each component (link or router), rather than to disable the faulty components. However, there will be two possible results : success or fail. Once the result is negative,

the corresponding wire-link channel or router must be de-activated. Thus, the network is modified and the recovery action enters the network-level.

At the network-level, three proposed fault-tolerant approaches can be used : **Default Backup Paths (DBP)**, **Spare Routers & Wire-link Channels** and **Fault-tolerant & Reconfigurable Routing Algorithm**.

Default Backup Paths (DBP)

[Koibuchi et al., 2008] proposed to add some lightweight redundant hardware "paths", like "Crossbar Bypass Bus", beside each component, in order to bypass the faulty components. However, this method is still an expensive solution, and these simple paths only support low throughput transmission.

Spare Routers & Wire-link Channels

[Bruck et al., 1993] suggested to add certain extra routers and wire-link channels, as redundant components. As shown in Figure 2.6, when some components are faulty and disabled, the spare components will be activated to rebuild the topology. However, this method can not handle the case where the number of faulty components exceeds the number of spare components. Moreover, this method requires very long wires that will reduce network performance.

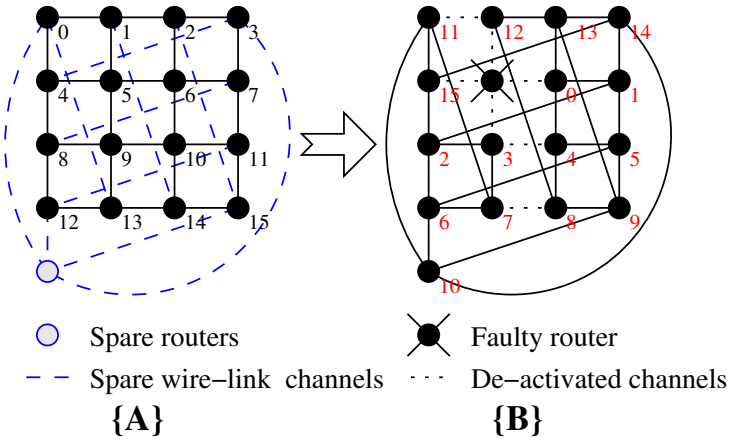


Figure 2.6 – FIG.A presents a network with one spare router and ten spare links. FIG.B presents a configuration to repair the network topology.

Fault-tolerant & Reconfigurable Routing Algorithm

In the high-end supercomputers scope, the network fault-tolerance researches, such as [Glass and Ni, 1993, Cunningham and Avresky, 1995, Chalasani and Boppana, 1995b,

Chalasani and Boppana, 1996, Duato, 1997], provided "on the field graceful degradation" theory : using the fault-free components to route the packet to bypass the faulty components with the help of fault-tolerant & reconfigurable routing algorithms. These published routing algorithms can be used in NoC to handle the modification of topology due to the permanent fault. Moreover, many researchers continue the same orientation to define and develop more fault-tolerant & reconfigurable routing algorithms for NoC, such as [Mejia et al., 2006, Nunez-Yanez et al., 2008, Fick et al., 2009a, Rodrigo et al., 2010].

However, the natural condition of the high-end supercomputers, where each faulty node (processor, network router, wire likes, etc.) can be manually, independently and physically tested, disabled, removed, replaced, reconfigured and recovered, is very different with NoC based system. The "on the field graceful degradation" of NoC means to define a practicable, complete and automatic detection, deactivation and reconfiguration mechanism, rather than only a fault-tolerant & reconfigurable routing algorithm.

In this thesis, we devote ourself to define and realize a practicable, complete and automatic ODDR ("On the field" Detection, Deactivation and Reconfiguration) mechanism for a 2D-Mesh NoC in a MP^{2}SoC architecture. The related works are presented and discussed in the next section.

2.2 Works Related to ODDR ("On the field" Detection, Deactivation and Reconfiguration) Mechanism

To our knowledge, there is no complete solution related to the ODDR mechanism. Thus, we separately present and discuss the works related to each of three problems : **NoC test**, **Configuration Infrastructure** and **Routing Algorithm**.

2.2.1 NoC Test

In the past ten years, many authors have proposed NoC test strategies. A non exhaustive survey of the major contributions to this area is given in the following summary organized by theme.

Manufacturing Test

As discussed in [Vermeulen et al., 2003], the initial goal of Network-on-Chip test is the manufacturing test : the faulty chips are thrown away. Whatever the test technique used, the goal is only to distinguish the faulty NoC circuit, rather than to tolerate faults. Aiming to this goal, some solutions are proposed : [Ubar and Raik, 2003,

Vermeulen et al., 2003, Pande et al., 2005, Grecu et al., 2005, Hosseinabady et al., 2006, Raik et al., 2006, Hosseinabady et al., 2007, Cota et al., 2007, Cota et al., 2008].

Yield Improving Test

In a modern high-density chip, the yield will be low. In order to improve this yield, the manufacture defect must be located. A faulty chip can then be reconfigured and sold at a lower cost. To reach this goal, some solutions [Amory et al., 2005, Raik et al., 2007, Petersén and Öberg, 2007, Herve et al., 2009, Concatto et al., 2009, Tran et al., 2006] are proposed, to achieve "test for diagnosis" strategy. The NoC test results are analyzed to diagnose the faulty components. However, without embedded auto-deactivation or auto-isolation of faulty components, this strategy isn't yet sufficient to fit the "on the field" fault-tolerance.

On The Field Test

Facing a future high-permanent-fault-rate MP^{2}SoC chip, not only the manufacture defects must be handled, but the failures of the chip that is already mounted in the final equipment, must be also handled. To fit such "on the field" fault-tolerance requirement, a "test for de-activation" strategy must be defined and implemented in NoC. Therefore, we need an off-line BIST with de-activation mechanism, where the components can be de-activated as soon as they have been detected as faulty, when the NoC is not yet running. As far as we know, there is not yet such BIST, and we must solve this important problem.

2.2.2 Configuration Infrastructure

In order to achieve "on the field" NoC reconfiguration mechanism, we must define and implement a configuration infrastructure in the MP^{2}SoC architecture, determine a configuration master, implement a configuration bus, and diagnose/locate the faulty/de-activated components in order to identify the modification of topology. To our knowledge, there are three related solutions : FDAR (Fault-Diagnosis-And-Repair), SDSC (Self-Diagnosis and Self-Configuration) and DCCI (Distributed Cooperative Configuration Infrastructure).

FDAR (Fault-Diagnosis-And-Repair) system

In [Kariniemi and Nurmi, 2006], the authors proposed to use each local processor to monitor, detect & diagnosis (with CRC) and repair (lock the faulty channels) each corresponding local router. This approach provides a one-to-one relationship between a couple (processor/router). A processor is a local configuration master, and the interconnection

between a couple (processor/router) is a local configuration bus. However, when a router or a processor/router couple is faulty, the network topology is globally modified, which can not be handled by FDAR, since the global configuration master and the global configuration bus are not defined.

SDSC (Self-Diagnosis and Self-Configuration) method

[Kolonis et al., 2009] proposed a mechanism for discovering in a MP^{2}SoC the fault-free paths between a specific I/O port and the fault-free processor cores. The result of the discovering process is a fault-free communication map, that is used to find out the fault-free components of NoC, and to configure them. This method is a centralized solution, as the discovering process is driven by the static configuration master, the smart I/O port. However, such smart I/O port controller will contain a very complex hardware structure, with a high failure risk. Thus, this method induces a weak point into the chip, as the I/O port is the critical resource.

DCCI (Distributed Cooperative Configuration Infrastructure)

The DCCI [Zhang et al., 2011] has been defined by REFAUVELET Dimitri at LIP6 within his thesis scope. The key idea is to have one embedded Configuration Firmware (CF) in each cluster [1]. With DCCI, a CF is able to communicate and exchange information with its 4 direct neighbor CFs.

According to the NoC detection & de-activation mechanism, DCCI makes the assumption that, the faulty components are disabled, the fault-free ones are enabled. The enabled routers use the default X-First routing algorithm. Thus, the NoC it-self enters the pre-configuration stage. The role of the DCCI distributed firmware is to progressively build (only relying on local communications between neighbor clusters) a trusted tree of operational clusters, where each operational cluster contains one operational (local master) processor running the CF. This tree is built in a bottom-up way, starting with operational clusters as leaves. This communication tree uses a limited part of the routing capabilities of the NoC. It uses only the local communication channels between neighbor clusters, to achieve software-based communication through dedicated mailboxes. This software-based communication tree can thus be seen as a slow and temporary communication infrastructure, dynamically constructed, at the beginning of pre-configuration stage, using the fault-free/activated NoC resources.

1. A cluster is a subsystem, containing several process cores, an embedded RAM, NoC router and so on. An example of a cluster is presented in chapter 3.

The root of tree is a cluster determined by a distributed election algorithm. The most important criterion in this election process is the capability of this root cluster to access an external mass storage where more exhaustive test programs and the final operating system itself are available, and can be loaded in the embedded RAM of the root cluster. The local master processor of the tree root can use this communication tree to make any processor in the tree execute any specific software task (debug, fine-grain test), to propagate any configuration command to any child tree node, or read any status information. Thus, it can be used to complete the configuration of the NoC, and is actually the global configuration master.

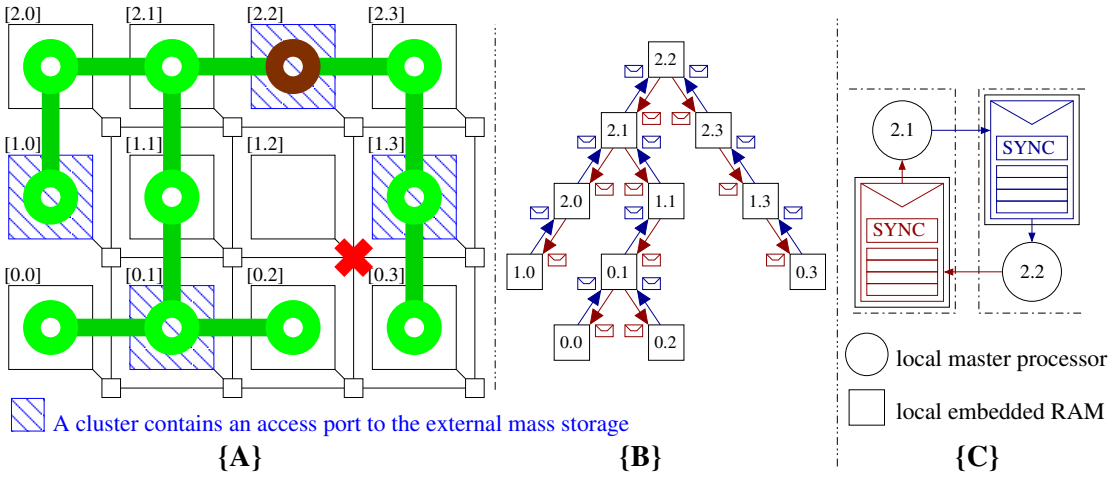


Figure 2.7 – On a damaged NoC, the router at coordinates [1.2] is faulty and disabled. A spanning DCCI communication tree is built as a result of each local CF task communicating only with its neighboring clusters. The tree root is at coordinates [2,2]. The local master processor of each tree node is presented as a circle.

Figure 2.7.{A} shows an example of a DCCI tree on a damaged NoC (the router at coordinates [1.2] is faulty and disabled). As stated before, the tree covers all reachable clusters. The local master processor of tree root, playing the role of the configuration master, becomes the “chip leader”.

Figure 2.7.{B} presents a global representation of the final DCCI tree. To be connected, two neighbor clusters (such as clusters [2.1] and [2.2]) must be able to communicate through two directed edges, for full duplex communication. Each edge corresponds to a software mailbox, as shown in Figure 2.7.{C}. We make the assumption that each cluster contains an embedded memory bank. A mailbox is a data structure in the local embedded RAM of the receiving node, and is shared by exactly one sender and one receiver. Consequently, two mailboxes are used for bidirectional communications. In the DCCI, a round-trip mail, traveling two neighboring mailboxes, is used to check the bidi-

rectional communications between two neighboring clusters. The fault-free clusters will be used for tree construction and the fault-free bidirectional communication channels can potentially become edges of the DCCI tree.

This tree can be considered as a trusted launching pad for exploration, configuration and operation. By using end-to-end protocol or flooding protocol over the tree, we can send command, application task, application results to one or to all nodes. And we can configure each router through this tree. Therefore, the DCCI tree is the global NoC configuration bus.

Compared with two above solutions (FDAR, SDSC), DCCI is more reliable. The configuration master is not a unique and critical resource, and is naturally competent for configuration task. In addition, reusing the fault-free components of NoC as the configuration bus is a low-cost solution. However, DCCI doesn't allow the diagnosis/localization of the faulty/fault-free components. Thus, the modification of topology can not be identified. So, this problem must be solved.

2.2.3 Routing Algorithm

As we want to define a deterministic, lightweight, deadlock-free and reconfigurable routing algorithm for a 2D-Mesh NoC, the adaptive routing algorithms are not discussed in this section. The published deterministic solutions for 2D-Mesh topology are [Wu, 2003, Boppana and Chalasani, 1994, Mejia et al., 2006]. In this section, we present and analyze these related works.

In fact, what ever the routing strategy is, the main idea is to route the packet to bypass the faulty routers or faulty links, but, with different approaches of deadlock prevention [2]. As proposed by Dally in [Dally and Seitz, 1987], the main idea of deadlock prevention is to remove the cycles, in the channel dependency graph (CDG). Thus, we discuss also the method of cycle removing, in each related work presentation.

As shown in Figure 2.8, in a Channel Dependency Graph (CDG), a node (red circle) means a channel, a directed edge (blue arrow) from a channel to another, means a possible path defined by the routing algorithm. According to Dally, the routing algorithm is deadlock-free, if the CDG doesn't contain any cycle.

---

2. The deterministic routing doesn't have to handle the livelock-ness problem, as it's always livelock-free.

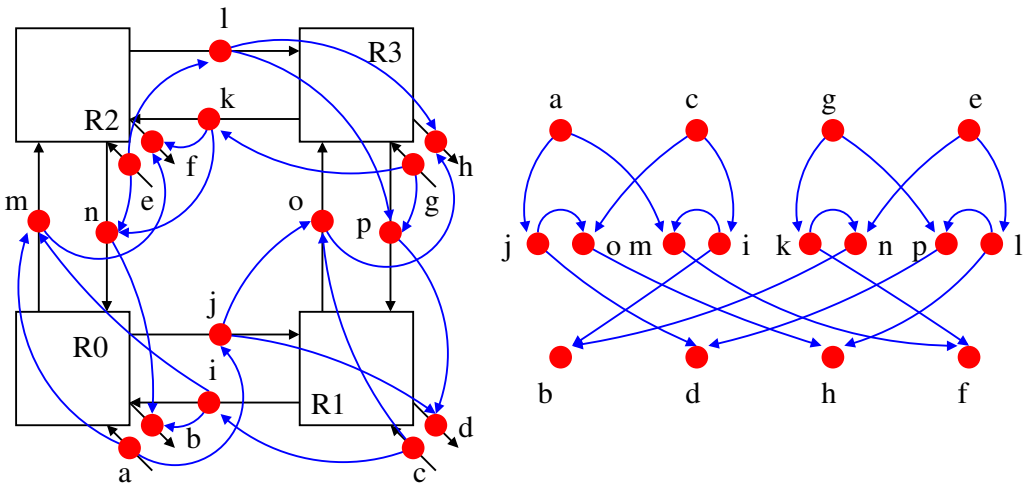


Figure 2.8 – The channel dependency graph of X-First routing algorithm in a 2×2 2D-Mesh NoC.

Turn Model

As shown in Figure 2.9.{A}, in a 2D-Mesh topology, a router can support various types of "turns".

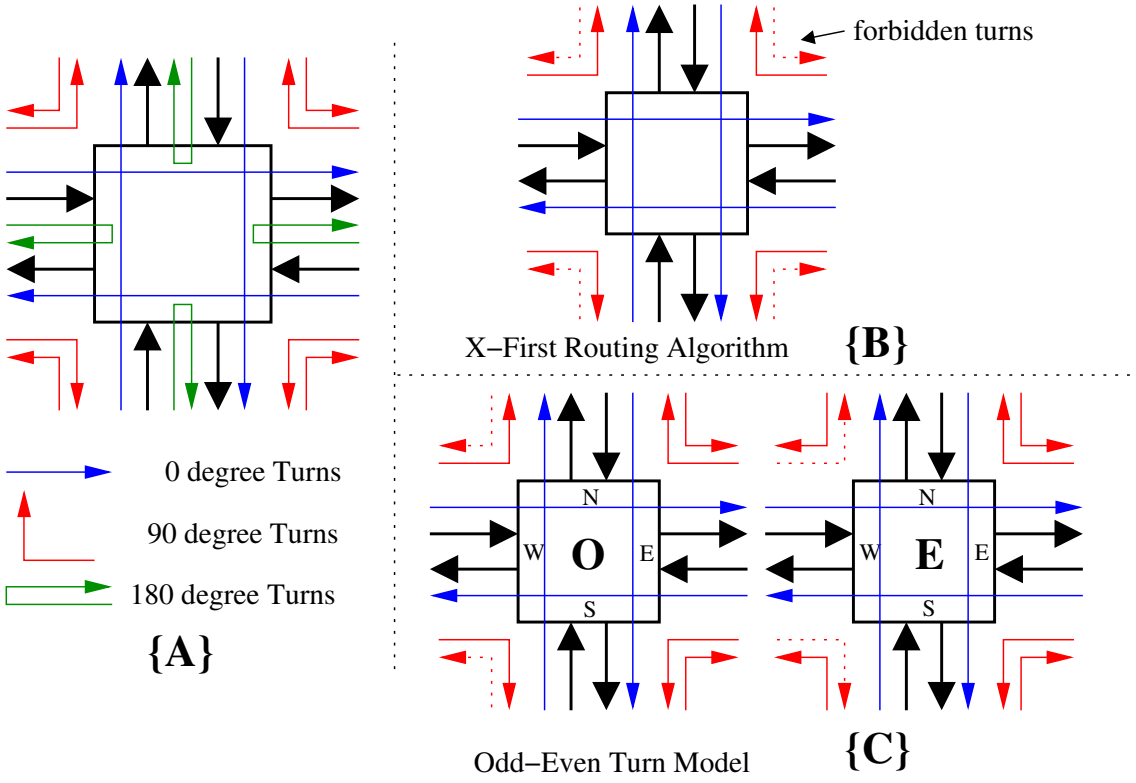
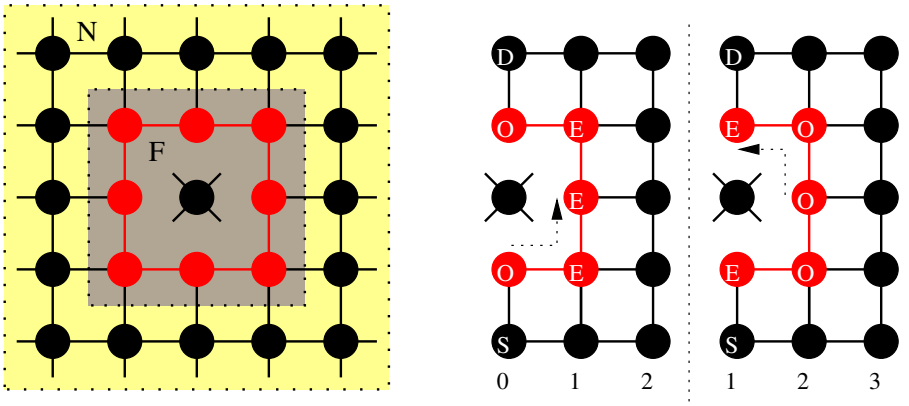


Figure 2.9 – FIG.A presents all turns possible in a 2D-Mesh router. FIG.B presents the turns restriction of the X-First routing algorithm. FIG.C presents the turns restriction of the odd-even turn model routing algorithm.

According to [Glass and Ni, 1994], the Turn Model proposed in, forbidding some turns in the routing function, can guaranty cycle-free in the CDG, so as to avoid deadlock. For example, the X-First and the odd-even turn model [Chiu, 2002] are two deadlock-free routing algorithms, based on Turn Model.

- **X-First :** As shown in Figure 2.9.{B}, the X-First forbids the 180 degree turns and forbids the turns from the Y dimension to X dimension.
- **Odd-Even turn model :** As shown in Figure 2.9.{C}, the odd-even turn model routing algorithm forbids the 180 degree turns and forbids the turns SW, NW in each router of odd column, and forbids the turns WS, WN in each router of even column.

To define a fault-tolerant routing algorithm, the authors of [Wu, 2003] proposed the extended X-First strategy, based on the X-First and Odd-Even turn model. According to this routing algorithm, a faulty 2D-Mesh topology is split into the normal region and the faulty region. As shown in Figure 2.10.{A}, in the normal region, the router uses the X-First routing algorithm. And in the faulty region, the border router uses the odd-even turn model routing algorithm. However, as shown in Figure 2.10.{B}, this strategy can not handle some single-faulty-router topologies (for example, when the faulty router is located at the West border of the Mesh). Thus, it can not fit the minimum fault-tolerant requirement.



N: Normal Region, X–Fisrt
F: Faulty Region, Odd–Even

{A}

The packet from S to D can not be routed in the faulty region, by Odd–Even.

{B}

Figure 2.10 – FIG.A shows that the router in the normal region uses X-First routing algorithm, the border router in the faulty region uses Odd-Even routing algorithm. FIG.B shows that what ever the first column is odd or even, the packet from S to D can not be routed in the faulty region, caused by the forbidden turns of Odd-Even turn model.

Segment-based Routing Algorithm

In [Mejia et al., 2006], the authors proposed a segment-based routing algorithm, using a local turn restriction strategy. As shown in Figure 2.11.{A}, with the help of a segment configuration mechanism, a faulty topology is split into several segments, each segment is a subnetwork that consists of a set of routers and links. In each segment, a local turn restriction must be placed, such as the 2.11.{B}, to forbid some bidirectional turns. As segments are independent, each CDG of each segments is cycle-free. So, the combination of all CDGs is also cycle-free, as shown in Figure 2.11.{C}, namely, the routing algorithm is deadlock-free.

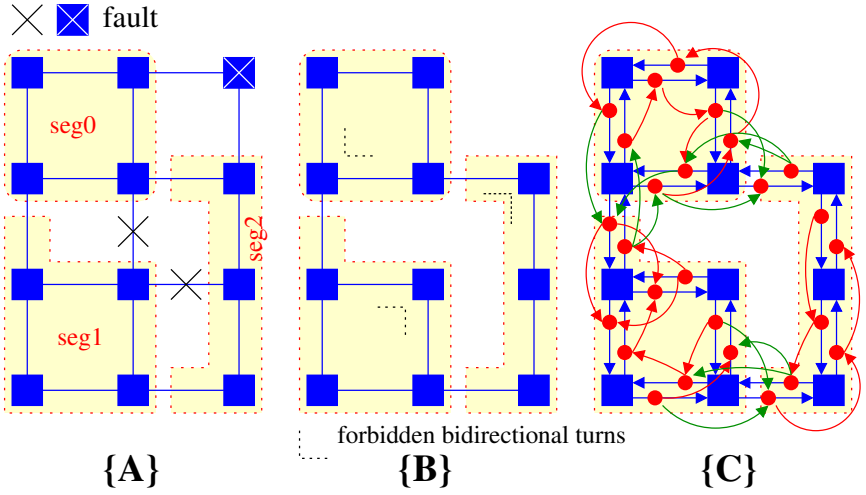


Figure 2.11 – An example of segment-based routing algorithm.

The segment-based routing algorithm can handle a complex faulty topology. But, it can only be implemented by a table-based routing function in each router. Thus, a reconfigurable routing table per router is mandatory, and this will induce a great area overhead for a NoC with large 2D-Mesh topology.

Virtual Channel

In [Boppana and Chalasani, 1994], the authors proposed to add virtual channels to avoid deadlocks. The main idea is to place different turn restriction on each virtual network, so as to guaranty each level to be deadlock-free, while all 0 and 90 degree turns are permitted on whole network.

For example, in Figure 2.12, on a network with two virtual channels (level C0 and level C1), different turn restrictions are placed to create the bypassing paths. Thus, using the level C0, the network can route a packet to bypass a faulty region from West to East or from East to West, without deadlock.

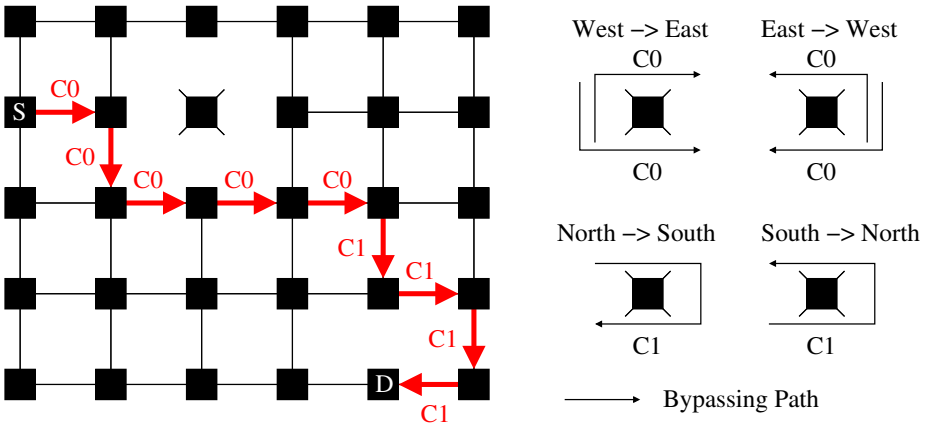


Figure 2.12 – An example of virtual channel based routing algorithm.

The virtual channel based routing algorithm can handle all single-faulty-router topologies, but, it's very expensive. A network with two virtual channels takes nearly twice the silicon area of one channel network.

It should be noted that, a fault-tolerant routing algorithm must be formally proven to be deadlock-free with all single-faulty-router topologies, otherwise, they are not be worthy of confidence. However, almost all published solutions miss this important step.

As far as we analyzed, the published deterministic routing algorithms either are expensive, or can not fit the minimum fault-tolerant requirement. Moreover, they haven't been formally proven to be deadlock-free. Therefore, we have to define a new fault-tolerant routing algorithm.

2.3 Conclusion

In this chapter, we analyzed the state of the art of the researches related to 2D-Mesh NoC routing algorithm reconfiguration strategy. And three remaining main problems are identified, on three topics. These problems must be solved in this thesis.

1. **NoC Test** : In order to support "on the field" NoC reconfiguration strategy, an off-line, "test for de-activation" BIST mechanism is necessary. As far as we know, there is not yet such BIST, and we must solve this important problem.
2. **Configuration Infrastructure** : The DCCI will be used in this thesis as the configuration infrastructure. However, DCCI doesn't allow the diagnosis/localization of the faulty/fault-free components. Thus, the modification of topology can not be identified. So, this problem must be solved.
3. **Routing Algorithm** : As far as we analyze, the published deterministic routing algorithms either are expensive, or can not handle some single-faulty-router topolo-

gies. Moreover, they haven't been formally proven to be deadlock-free. Therefore, we have to define a new fault-tolerant routing algorithm.

Chapter 3

DSPIN & MP^{2}SoC

Contents

In this chapter, we present a typical 2D-Mesh Network-on-Chip used in this thesis, DSPIN (Distributed Scalable Predictable Interconnect Network). And we introduce the general structure of DSPIN-based, shared memory, MP^{2}SoC architecture.

3.1 A Summary of DSPIN & MP^{2}SoC

The DSPIN micro-network (Distributed Scalable Predictable Interconnect Network) was designed by the LIP6 laboratory and physically implemented by ST Microelectronics. It's a typical 2D-Mesh NoC, supporting MP^{2}SoC architectures, and the GALS (Globally Asynchronous Locally Synchronous) approach.

As shown in Figure 3.1.{A}, a DSPIN based, shared memory, MP2SoC architecture is typically composed of a set of tiles called clusters, each cluster is a synchronous domain.

As shown in Figure 3.1.{B}, a cluster may contain one or several programmable processors, a local interconnect, an embedded RAM, an embedded ROM (for configuration firmware) and two routers : in order to avoid deadlocks in command/response traffic, each cluster contains two independent routers implementing two separated sub-networks for commands and responses. In addition, some special clusters contain I/O ports controllers, used to access external mass storage devices. In order to support "on the field" reconfiguration mechanism, we suppose that each programmable processor contains a timeout mechanism : when it executes a memory load/store operation, the timeout mechanism is

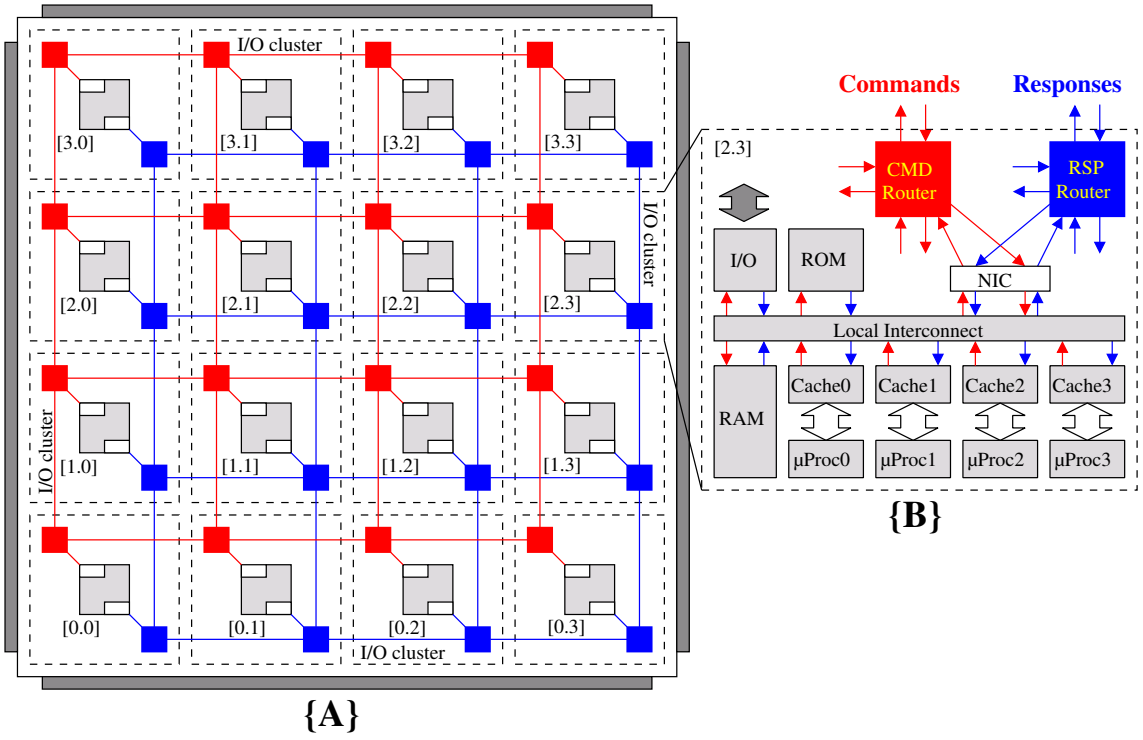


Figure 3.1 – A 4×4, DSPIN based, shared memory, MP2SoC architecture. In such architecture, the memory is physically distributed, but logically shared : All processors can directly address all memory banks.

triggered. If the transaction fails, the timeout triggers an interrupt and the processor enters its exception mode.

3.2 DSPIN Structure

The DSPIN router micro-architecture is distributed : the basic units of a router, such as routing function, crossbar and input/output buffering devices, are not centralized to result in an integrated component, but distributed into five separated and identical modules : North, South, East, West and Local, as shown in Figure 3.2.{A}. Moreover, the North, South, East and West modules are physically placed on the corresponding clusters borders (the Local module can be placed everywhere, in the cluster). This feature, combined with the mesh topology allows us to classify the network wires in two classes :

Intra-cluster Wires

Intra-cluster wires, connecting modules in the same cluster (for example : West module connects to North, South, East and Local modules), are the long wires. But, the wire length is bounded by the physical area of a given cluster, a synchronous domain. With the

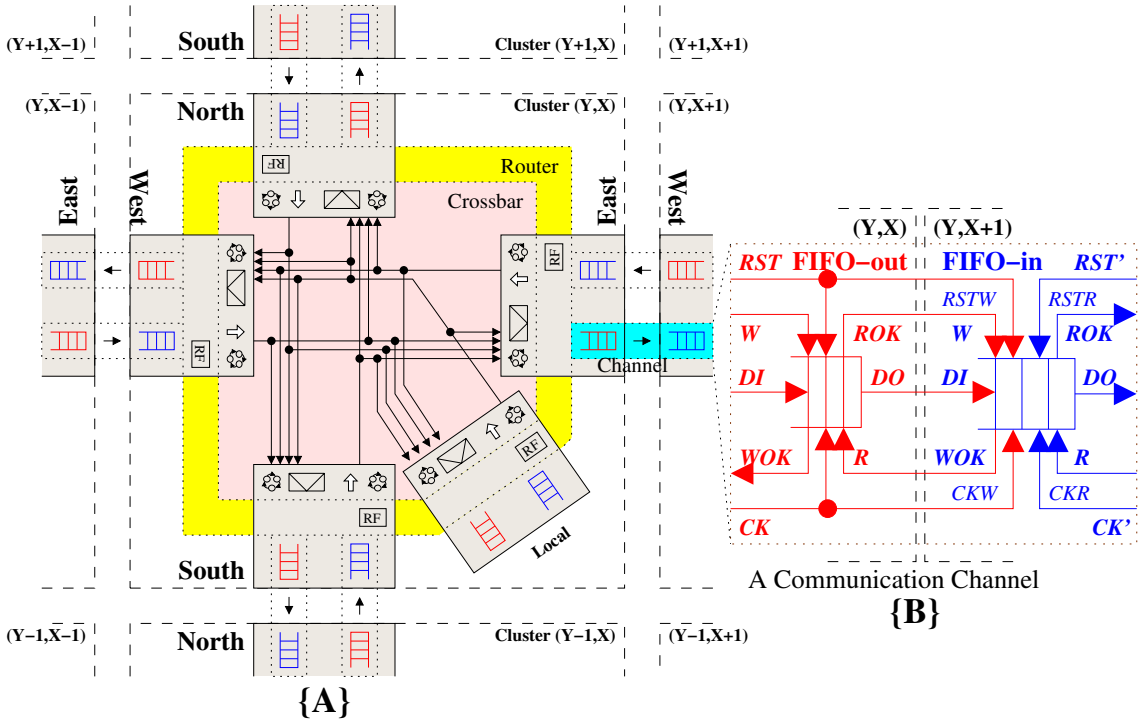


Figure 3.2 – A generic DSPIN router architecture.

help of intra-cluster wires, all modules are connected as a full crossbar as shown in Figure 3.2.{A}, in order to support the fault-tolerant and reconfigurable routing algorithm.

Inter-cluster Wires

Inter-cluster wires, connecting modules in adjacent clusters (for example : the North module of cluster (Y,X) is connected to South module of cluster (Y+1,X)), are very short wires, because those components are very close from/to each other. With the help of inter-cluster wires, the adjacent output/input buffering devices of two neighboring routers are seamlessly connected. It should be noted that, in order to support the GALS (Globally Asynchronous Locally Synchronous) approach, the input buffering device is a bi-synchronous FIFO (First-In, First-Out, depth is 4) [Miro Panades and Greiner, 2007], and the output buffering device is a synchronous FIFO (depth is 4), as shown in Figure 3.2.{B}. Thus, a packet can be transmitted from a clock domain (CK) to another clock domain (CK').

In this thesis, we redefine the boundaries of router and channel, as shown in Figure 3.2 : the router consists of the routing function (RF) and the crossbar, the channel is composed of a couple of output/input FIFOs.

3.3 DSPIN Switching and Flow-Control Policy

Like most NoC designed for shared memory multiprocessors architectures, DSPIN is a packet-switching network. As shown in Figure 3.3, a packet is divided into smallest flow control units called flits, according to the wormhole routing conception (a review of wormhole routing conception can be found in [Ni and McKinley, 1993]). The first flit is the header flit that includes the destination address defined in absolute coordinates Y and X. The second and third flits contain the payload protocol informations. The remaining flits are payload data flits. The trailer flit contains the end of packet (EOP) mark. In a DSPIN router, the header flit of a packet is analyzed by the routing function (RF) logic, and an output port is selected. Then the crossbar builds a path from the input port to the selected output port, and the whole packet is transmitted to the target.

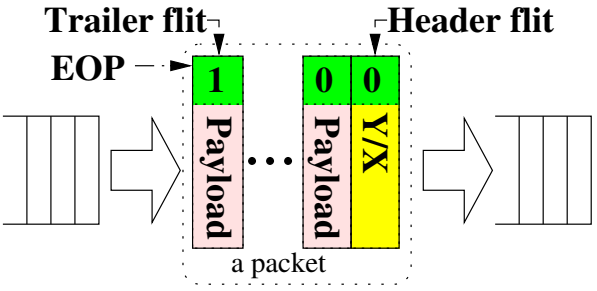


Figure 3.3 – A generic DSPIN packet.

3.4 DSPIN Deadlock-Free Analysis

The default routing algorithm implemented in the DSPIN micro-network is X-First. With this routing algorithm, the packets are firstly routed on the X direction, and then on the Y direction. As shown in Figure 2.8, using the CDG (Channel Dependency Graph), on the both command/response subnetworks of DSPIN, the deadlock-free feature is proven.

3.5 Conclusion

DSPIN is a typical 2D-Mesh Network-on-Chip, supporting the shared memory, MP2SoC architectures and the GALS approach. The ODDR ("On the field" Detection, De-activation & Reconfiguration) mechanism proposed in this thesis, will use the following DSPIN characteristics :

- 2D-Mesh topology
- Full-Crossbar Switch

– FIFO-based Flow Control Policy

– Default routing algorithm : X-First

All the Network-on-Chip micro-architectures having these stated characteristics, used in a shared memory, multi-cores or MP^{2}SoC architecture, can adopt and implement the general ODDR scenario presented in chapter 4.

Chapter 4

General ODDR Scenario

Contents

In this chapter, we introduce the ODDR ("On the field" Detection, De-activation & Reconfiguration) mechanism. This mechanism provides a "graceful degradation" based (permanent) fault-tolerance, for a 2D-Mesh Network-on-Chip used in a shared memory, MP^{2}SoC architecture. The general ODDR scenario contains three stages with three major roles :

1 Initialization stage/NoC Test
2 Pre-configuration stage/Configuration Infrastructure
3 Configuration stage/Routing Algorithm

4.1 Initialization Stage/NoC Test

In this stage, a fully distributed off-line BIST is proposed, according to the "test for de-activation" strategy. As shown in Figure 4.1.{a}, this BIST is executed as an initialization procedure, at each system reboot or chip power-on. It aims to avoid evil failure propagation and clean NoC malfunctions, by detecting and de-activating the faulty components.

As shown in Figure 4.1.{b}, with the help of additional multiplexers, we can insert, at each interface between a router and a channel, the router/channel test generator & analyzer. Thus, we distribute the corresponding test module into each router and each channel, to test each component in parallel and in isolation. In order to avoid the mutual interference (interfering), each router should be tested before each channel. In sum, there are two

possible results :

- If a router is detected as faulty, all input and output channels are directly considered as faulty without test. Thus, the router and all channels are de-activated.
- If a router is fault-free, and a channel is detected as faulty, this channel is de-activated, and the router kept activated.

Each de-activated channel is configured to behave as a “Black Hole”, that discards any incoming data, and produces no outgoing data. With this configuration, when a packet is routed to a faulty channel or a faulty router, the packet is discarded, which will trigger a timeout exception at emitter side (a processor core). These exceptions will be used after building the DCCI (Distributed Cooperative Configuration Infrastructure), to diagnose/locate the faulty routers and channels. This deactivation mechanism is detailed in chapter 5 section 5.4.

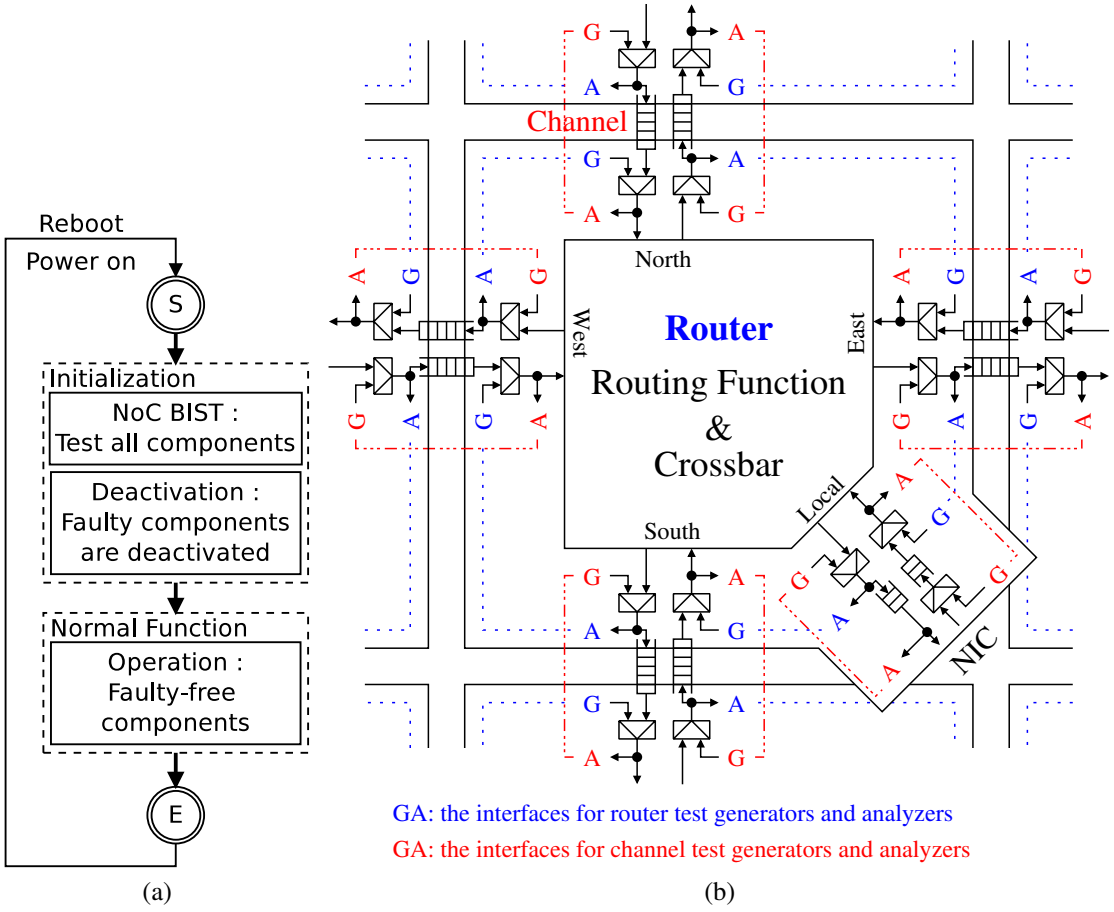


Figure 4.1 – {a} : A dedicated off-line NoC BIST is executed at each system reboot or chip power-on, as an initialization procedure. {b} : The interfaces between a router and a channel are broken by multiplexers. The embedded test generators & analyzers are reasonably placed, to realize a fully distributed BIST.

This NoC BIST with de-activation mechanism was published in [Zhang et al., 2010].

It is detailed in chapter 5.

4.2 Pre-configuration Stage/Configuration Infrastructure

After the NoC initialization procedure (BIST) is done, the faulty components have been disabled, the fault-free components are activated to work in a functional mode. The activated routers are configured to use the default routing algorithm, X-First. The network enters the pre-configuration stage.

As shown in Figure 2.7 and presented in chapter 2 section 2.2.2, at the beginning of the NoC pre-configuration stage, a DCCI communication tree is created. Each tree node is an embedded processor core. The tree root is the global configuration master, and the tree itself is the global configuration bus. It should be noted that, the configuration master must be able to identify the modification of topology (versus the ideal 2D-Mesh). Concerning this problem, we propose a software-based diagnosis/localization of faulty/fault-free components. The diagnosis/localization procedure will be distributed into each tree node, to test all paths defined by the default routing algorithm - X-First.

In Figure 4.2, we briefly describe this diagnosis/localization procedure by an example.

►1◄ Figure 4.2.{A} presents a 3×3 2D-Mesh NoC containing one faulty component : the router in cluster 3. And this router and all associated channels are de-activated and configured as “Black Hole”, at the end of test stage. Then, the whole network enters the pre-configuration stage.

►2◄ As shown in Figure 4.2.{B}, the DCCI communication tree is constructed, at the beginning of the network pre-configuration stage. The tree root is the cluster P5, it plays the role of the general global reconfiguration master.

►3◄ Figures 4.2.{C-E} : the tree root makes each node execute the local diagnosis/localization procedure. Each node (such as P0) tests each X-First path (sourced from P0). If a path contains a “Black Hole”, it’s faulty. Otherwise, it’s fault-free (blue path). For each fault-free path, the routers and the channels are marked (brown) in the source node (P0).

►4◄ Finally, as shown in Figure 4.2.{F}, when all nodes have finished their local diagnosis/localization procedures, the tree root will centralize all local marked fault-free components. Thus, the no-marked components are considered as faulty, and the modification of topology is identified.

This software-based diagnosis/localization of faulty/fault-free components on top of DCCI, was published in [Zhang et al., 2011]. It is detailed in chapter 6.

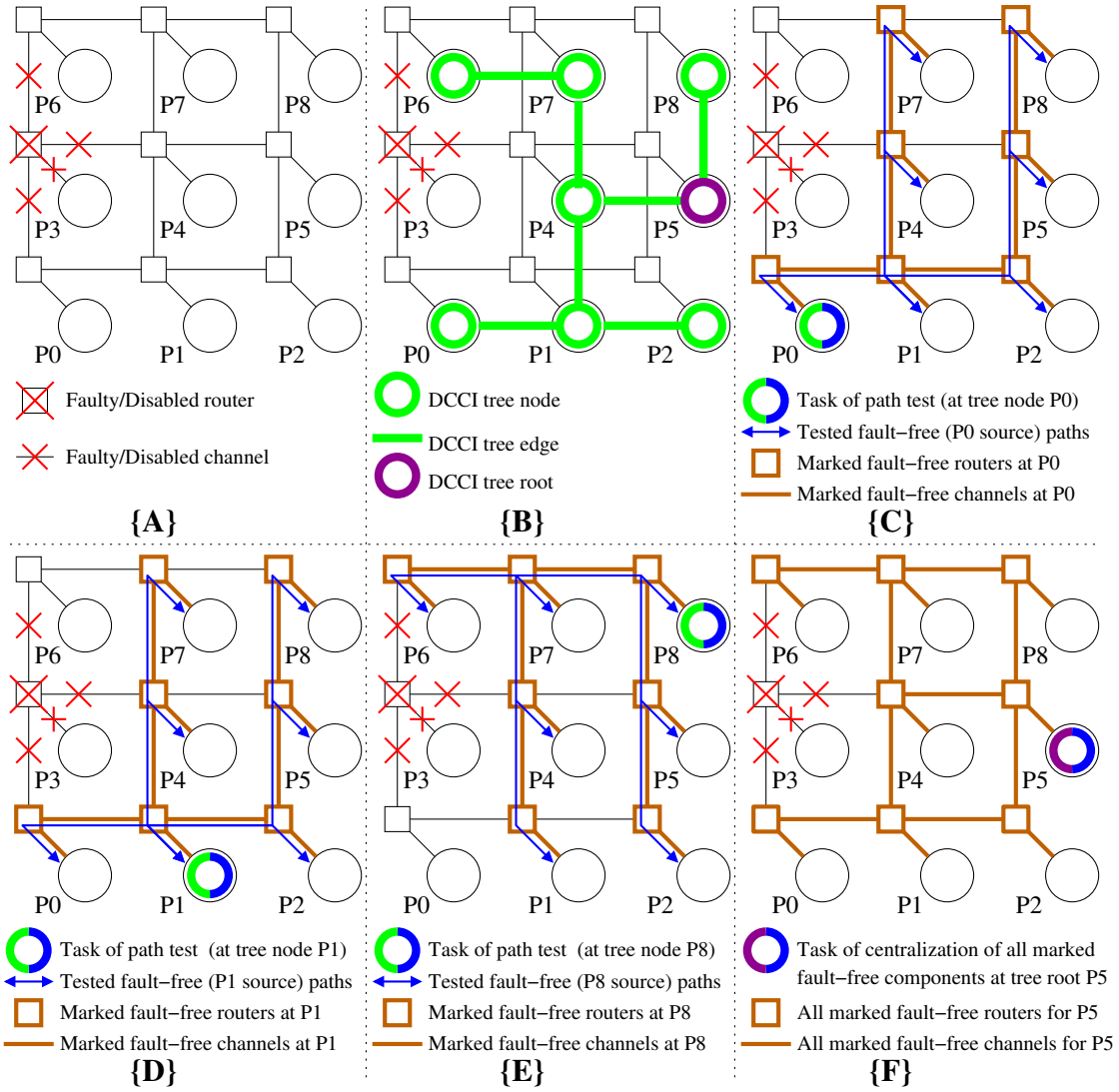


Figure 4.2 – An example of software-based diagnosis/localization procedure.

4.3 Configuration Stage/Routing Algorithm

As discussed in chapter 1 section 1.5.2, a 2D-Mesh topology involves five fault models :

- **Single** faulty channel topology.
- **Multiple** faulty channels topology.
- **Single** faulty router topology.
- **Multiple** faulty routers topology.
- **Mixed** Single or Multi faulty channel and router topology.

All these five fault models must be handled by the fault-tolerant reconfigurable routing algorithm. In this thesis, we suggest that, when any input channel of a router is detected as faulty, the entire router will be considered as faulty. Thus, these five fault models

are simplified into two types : **Single**-faulty-router topology and **Multiple**-faulty-routers topology. These two fault models can be handled by our proposed reconfigurable routing algorithm.

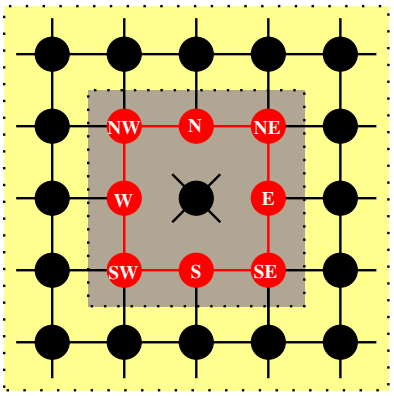


Figure 4.3 – The reconfigurable routing algorithm routes packets to bypass the faulty region through cycle-free contour.

As shown in Figure 4.3, with the proposed routing algorithm, a faulty topology is split into two regions : the normal region and the faulty region. In the normal region (yellow), the router doesn't need to be reconfigured, it uses the default routing algorithm : X-First. In the faulty region (gray), the border routers are configured to create a contour, in order to bypass the faulty router. The main challenge is to avoid that the new paths introduce a dead-lock. This routing algorithm has been formally proven to be deadlock-free for all single-faulty-router topologies. It should be noted that, supporting this reconfigurable routing algorithm, in each router, only 4 bits register for storing the configuration information is needed.

This deterministic, lightweight and reconfigurable routing algorithm was published in [Zhang et al., 2008]. It is detailed in chapter 7.

4.4 Conclusion

In this chapter, we summarize the general ODDR ("On the field" Detection, Deactivation & Reconfiguration) scenario for the 2D-Mesh Network-on-Chip, used in the MP^{2}SoC architecture. This scenario contains three stages with three major objectives :

1. Initialization Stage/NoC Test : At each system reboot or chip power-on, each component (router or channel) is tested, in parallel and in isolation, by a set of embedded, distributed BIST modules. The faulty components are detected and deactivated, finally configured to behave as a "Black Hole". The fault-free ones are activated to work. The technique detail is presented in chapter 5.

2. Pre-configuration Stage/Configuration Infrastructure : On top of an initialized NoC, a DCCI tree is built. Each node is an embedded processor core. The root is the global configuration master. The tree itself is the configuration bus. Through this tree, the root makes each node to execute local software-based diagnosis/localization procedure, and then it centralizes each local result to identify the faulty/fault-free components and the modification of topology. The technique detail is presented in chapter 6.
3. Configuration Stage/Routing Algorithm : According to the identification result of modification of topology, the network itself is split into two regions : normal region and faulty region. The routers of normal region are not reconfigured, but the border routers of the faulty region are configured to create a (deadlock-free) contour, so as to bypass the faulty/disable routers. The technique detail is presented in chapter 7.

With this scenario, a damaged 2D-Mesh NoC can be structurally tested, reconfigured and functionally recovered, in a shared memory MP^2SoC architecture.

Chapter 5

NoC Test

Contents

In this chapter, we detail a distributed off-line BIST architecture dedicated to 2D-Mesh NoC. This architecture has been implemented in the DSPIN micro-network, and both SAF (Stuck-at Fault) coverage and silicon area overhead have been evaluated.

5.1 Test Strategy

Aiming to support the ODDR ("On the field" Detection, De-activation & Reconfiguration) mechanism, a fully distributed off-line BIST is proposed, respecting the "test for de-activation" strategy. This BIST will be systematically operated, at each system reboot or chip power-on, to detect and de-activate the faulty components. And the (BIST) test process, shown in Figure 4.1.{a}, is integrated in NoC as the initialization procedure. With the help of this initialization procedure, the NoC itself can prevent the evil failure propagations and clean the malfunctions, so that it reaches a clean state, ready for self-reconfiguration.

In the following subsection, we analyze the DSPIN malfunctions due to a faulty communication channel. The analysis explains the importance and necessity of the initialization procedure.

DSPIN malfunctions due to a faulty channel

The DSPIN malfunctions due to a faulty channel are analyzed and shown in Table 5.1. In this analysis, the point-to-point channel is described as two FIFOs (Figure 5.1). The discussed faults are the SAFs (Stuck-At Faults) injected on the channel flow-control signals (W,WOK,R,ROK) as well as on the channel data signals (DI,DO).

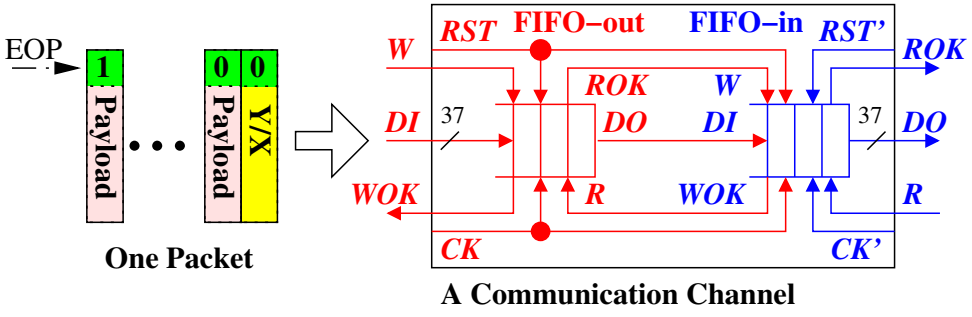


Figure 5.1 – A generic DSPIN channel.

From this analysis, we conclude that most NoC malfunctions due to a faulty channel are very dangerous for NoC self-reconfiguration. In order to avoid these destructive behaviors, the faulty channel must be de-activated as soon as it has been detected. This test and de-activation mechanism must be totally distributed, and must be done locally/separately for each router and each communication channel, when the NoC is not yet running.

Fault	Behavior
Flow Control Signals : W,WOK,R,ROK	
W/SA0	The channel ignores any packet arriving, namely, those packets are lost.
W/SA1	When the channel isn't full, it stores any DI as a packet flit, even if it is not valid. That leads to the self-generating fake packets. These fake packets will quickly fill the channel, eventually block the whole NoC.
WOK/SA0	The channel is always considered to be full, which leads to blocking some other channels, eventually blocking the whole NoC.
WOK/SA1	The channel is never considered to be full, so the packets are always accepted, even if it's full. Some of the packets routed to this channel will be lost or corrupted.
R/SA0	The channel can store the packets, but these packets cannot be consumed. The packets will quickly fill this channel, eventually block the whole NoC.
R/SA1	The packets are consumed, even if the router doesn't allocate an output port for it. These packets are lost.
ROK/SA0	The channel is always considered to be empty, even if it is not. Consequently the packets will never be consumed and will quickly fill this channel, eventually block the whole NoC.
ROK/SA1	The channel is always considered to be non empty, even if it is empty. That leads to self-generating fake packets (like W/SA1).
Data Signals : DI, DO	
EOP/SA0	If the End of Packet mark is lost, all packets become a unique packet, and the output port is never de-allocated, eventually blocking the whole NoC.
EOP/SA1	If the End of Packet mark is always set, the channel transforms every packet arriving to a set of one-flit fake packets. These packets will block the whole NoC, when they can not be consumed. Or they will destroy the function of the receiver, when they are routed to wrong destination.
YX/SA01	If the destination coordinates of a packet is changed, this packet is routed to wrong destination. At worst, it will destroy the function of the receiver.
Payload/SA01	The packets will be changed. At worst, it will destroy the function of the receiver.

Table 5.1 – Analysis of stuck-at fault impact on the NoC functions.

5.2 Test Process

To locally/separately test each component, we insert a set of multiplexers at each interface of router and channel, to break their original connections. And then, we insert the embedded test modules, and place them according to interactional placement as discussed in chapter 1 section 1.3.2.

As shown in Figure 5.2, to drive the router test, we insert one test controller (ATC : Auto-Test Controller) per router. To drive each channel test, we insert one test generator (ATG : Auto-Test Generator) per output channel, and insert one test analyser (ATA : Auto-Test Analyzer) per input channel.

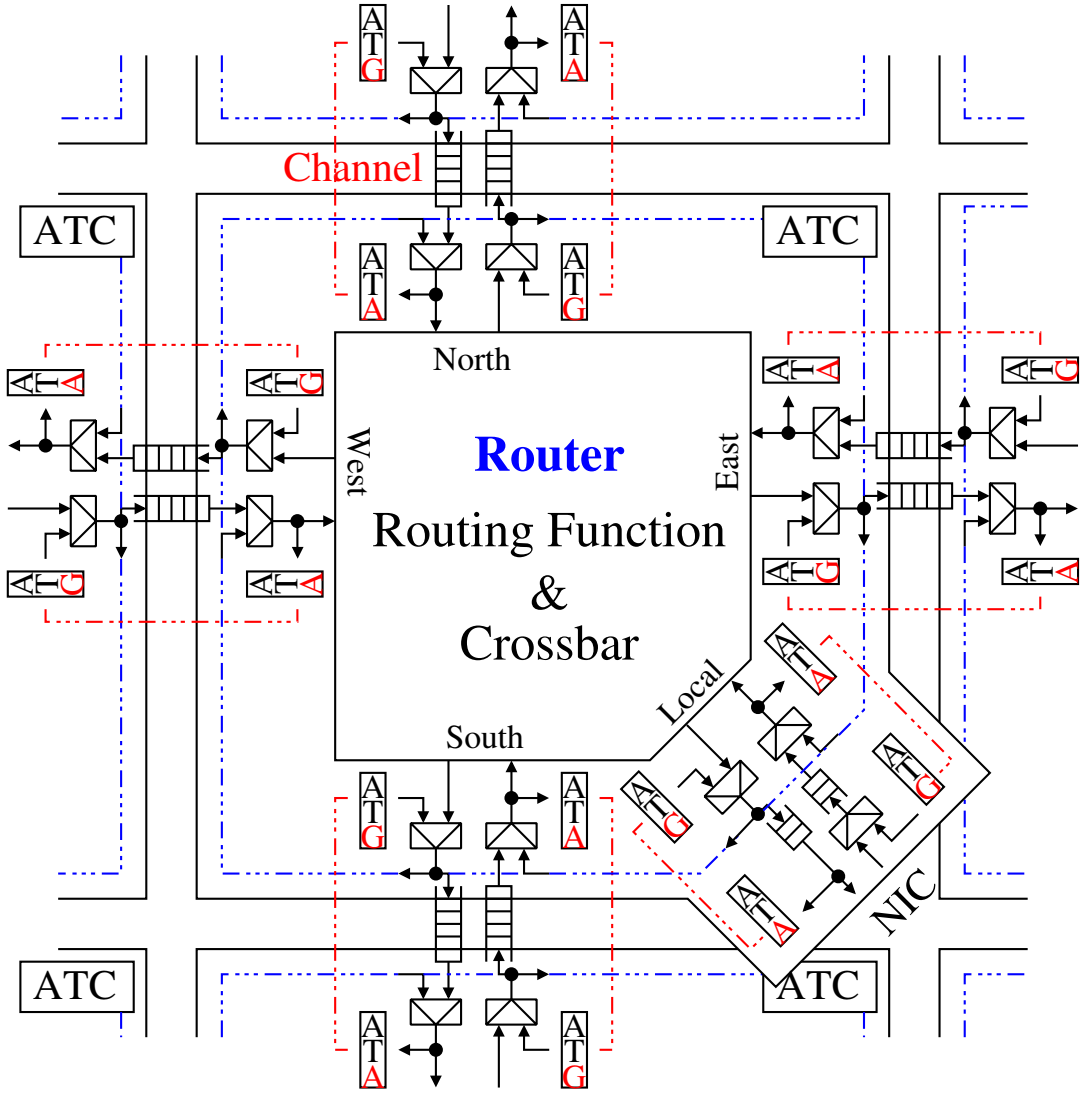


Figure 5.2 – The embedded test modules : one ATC per router, one ATG per output channel and one ATA per input channel.

Avoiding the mutual interference between router test and channel test, we propose an algorithm of test process, distributed in each router, to achieve two levels of parallelism as shown in Figure 5.3.

In this algorithm :

- At each system reboot or chip power-on, the test of all routers is started in parallel. Each router test is driven by a corresponding local ATC.

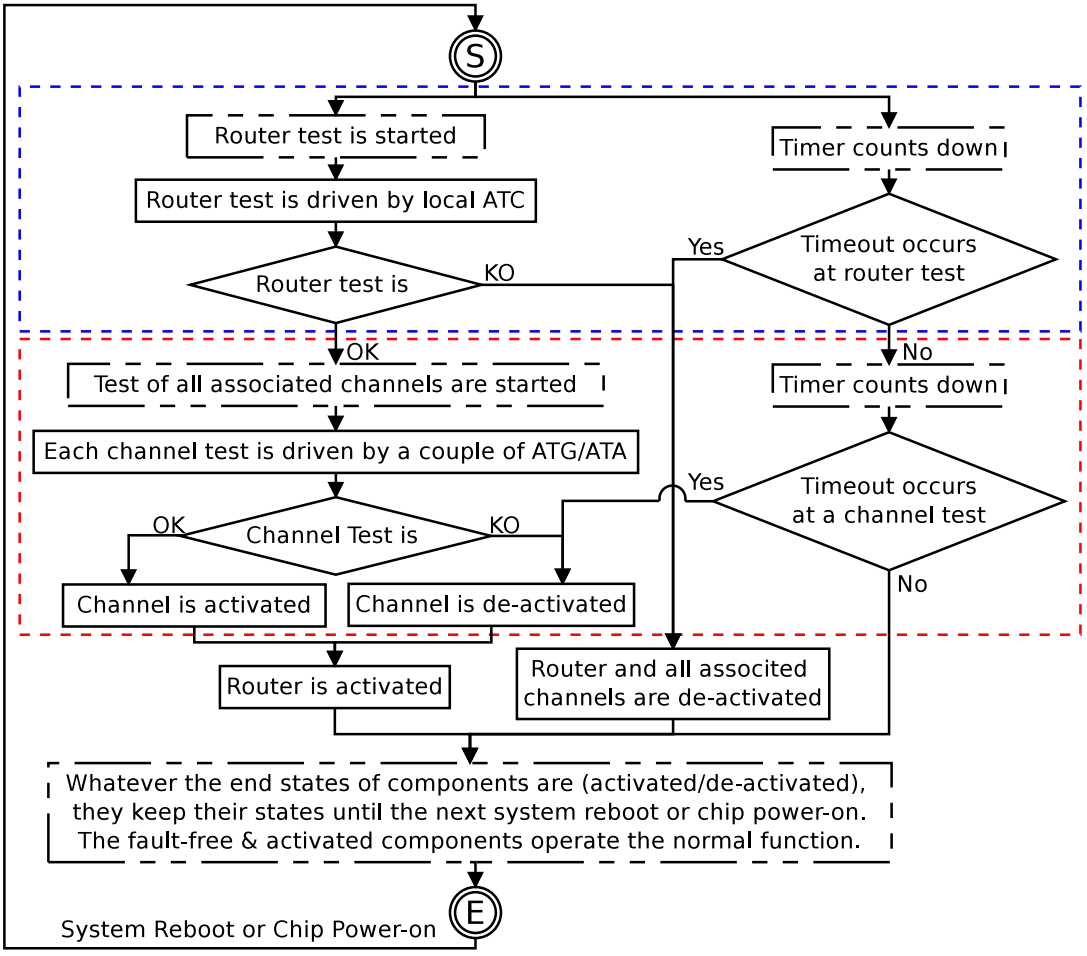


Figure 5.3 – The algorithm of test process distributed in each router.

- If a router test is KO, the router is considered as faulty. The router itself and all associated input/output channels are de-activated. Otherwise, the test of all associated communication channels is started in parallel.
- Each channel test is driven by a couple of ATG/ATA. If a channel test is KO, the channel is considered as faulty, and then de-activated. Otherwise, the channel is activated.
- For a router, when the test of all associated channel ends, the router is activated.
- When a local test doesn't go through, a timeout mechanism implemented in each router, forces the completion of the router/channel test, and the corresponding router/channel is de-activated.

Thanks to this algorithm, the router test and the channel test can be coordinated without mutual interference. And the test process can be operated before the normal function as the initialization procedure. Thus, the faulty components can be de-activated and the fault-free components can be activated, when NoC is not yet running.

5.3 BIST Structure

As shown in Figure 5.4, we detail the BIST structure implemented in a router.

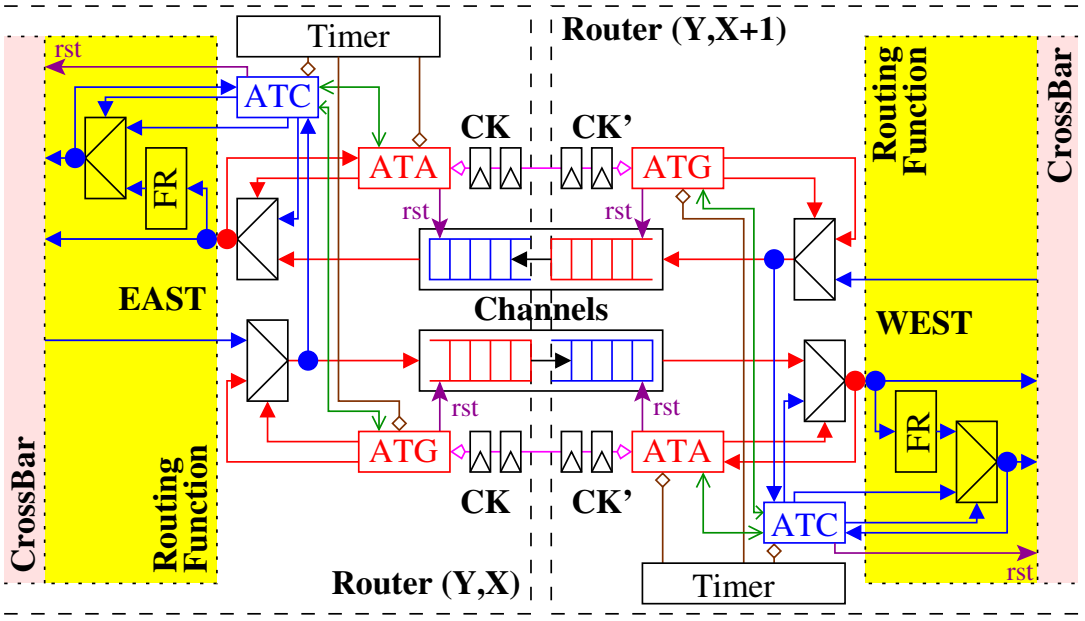


Figure 5.4 – Test Structure.

In this structure :

- The basic test modules : ATC (Auto-Test Controller), ATG (Auto-Test Generator) and ATA (Auto-Test Analyzer) are realized by FSM (Finite State Machine). Thus, there are one master FSM and ten slave FSMs in a generic router with five ports.
- The master FSM : ATC, is in charge of two tests in router : the routing function test and the crossbar test. It de-activates/activates the crossbar through RESET (rst) signal.
- The slave FSMs : ATGs and ATAs, are in charge of the corresponding channel test. They de-activate/activate the (FIFO) channel through RESET (rst) signal.
- In each router, the master FSM communicates with the slave FSMs through some command/acknowledge handshake signals, to operate the test process.
- The test of each bi-synchronous channel is driven by a couple of ATG/ATA. As ATG/ATA belong to different routers and different clock domains, they communicate asynchronously through a limited number of handshaking signals, thanks to re-synchronization flip-flops.
- The multiplexers controlled by ATGs and ATAs offer 2 functions : isolate router test and channel test, and configure the behavior of the faulty channel. This behavior configuration is detailed in the next section.
- A timer is inserted to provide the timeout mechanism.

This test structure is symmetrical and identical for each generic router. We can therefore modularize this structure and reuse it in any 2D-Mesh NoC (which has the characteristics presented in chapter 3 section 3.5).

5.4 Channel De-activation Mechanism

As shown in Figure 5.5, a faulty channel is de-activated by a couple of ATG/ATA, through the RESET signals (RST, RST'). And the de-activated channel is behaviorally configured by ATG/ATA and ATC, through enforcing two flow-control signals (WOK=1 and ROK=0). Thus, the channel is always empty for consumer, is always no-full for producer. In other words, the faulty/de-activated channel becomes a "Black Hole", that discards any incoming data, and produces no outgoing data.

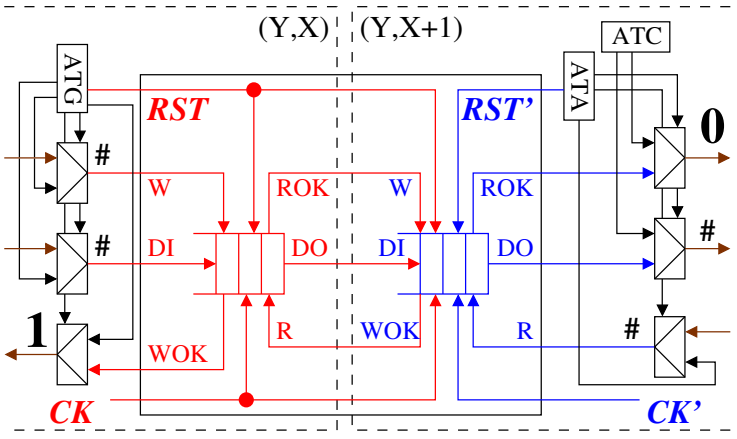


Figure 5.5 – De-activation and behavior configuration of a faulty channel.

It should be noted that, if the router is fault-free, only the associated faulty channel is de-activated and configured to become a "Black Hole". If the router is faulty, all associated channels are de-activated and configured to be "Black Holes". This auto-test phase results in a set of inner & invisible modifications, involving NoC's topology, structure and function. Therefore, each "Black Hole" must be diagnosed and located, in the pre-configuration stage.

In fact, the feature of the "Black Hole" makes itself easier to be tested by a packet, since the packet is discarded and a "transaction timeout" failure occurs at emitter. Thus, we propose a software application, that produces the packets and observes the "transaction timeout" failures, to diagnose and locate each "Black Hole". This software application is detailed in the next chapter.

So, we don't need any more hardware mechanism to extract each local test result. This advantage makes our proposed BIST be more robust, reliable and lightweight.

5.5 ATC, ATG and ATA FSMs Design

In this section, we present the details of ATC, ATG and ATA FSMs Design. It should be noted that, from the point of view of ATC, the ATG and ATA FSMs are very similar. They are thus described by one FSM (AT#FSM) as shown in Figure 5.6.

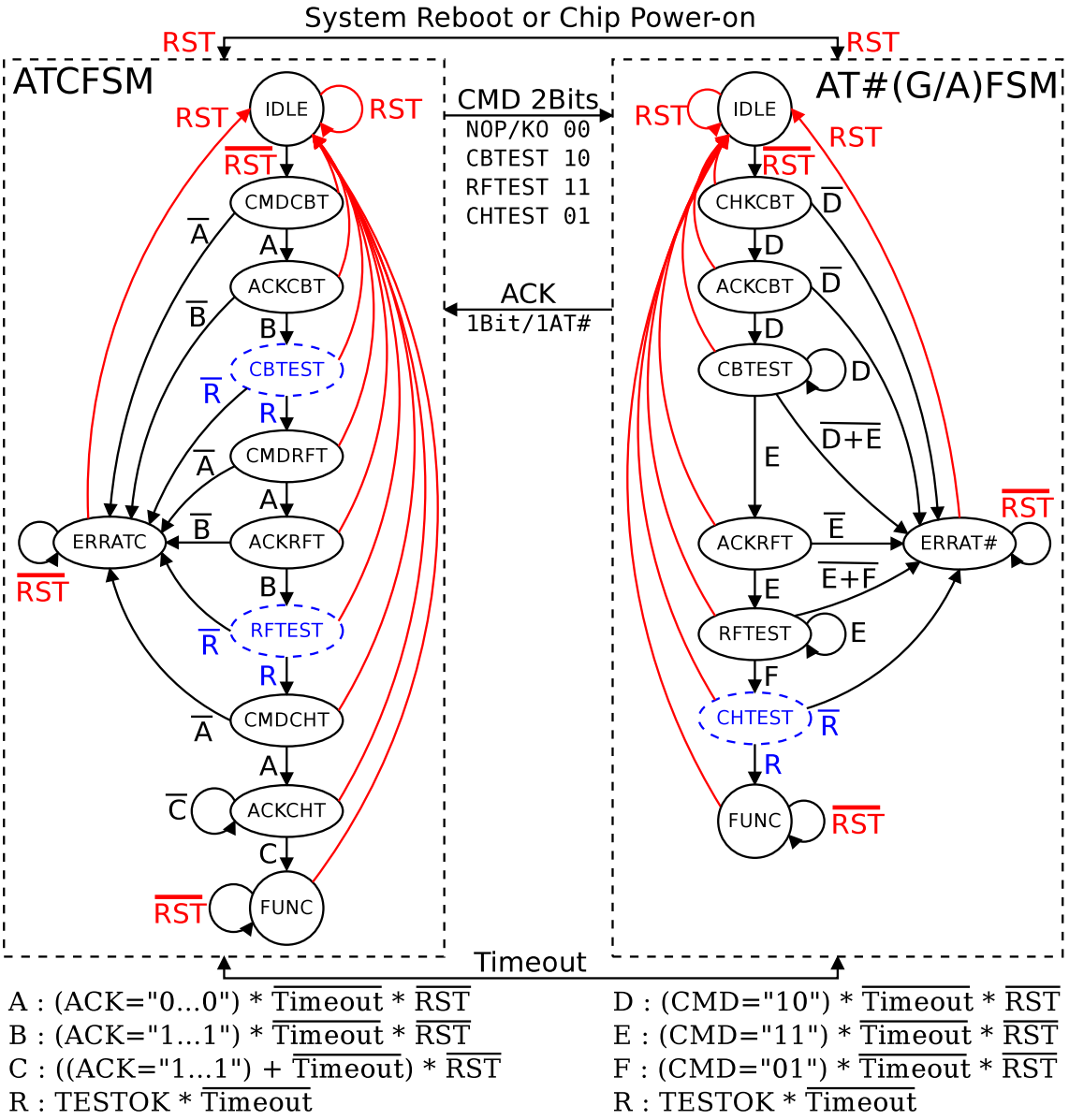


Figure 5.6 – ATC and AT# (G/A) FSMs.

►1◄ At each system reboot or chip power-on, ATC and all AT#s are reinitialized by the global RESET signal, and they enter in the idle state.

►2◄ When the RESET signal is disabled, ATC and all AT#s do the first command/acknowledge handshake through signals CMD&ACK. This handshake is realized by two states.

- ATC uses “CMDCBT” to send the command “CBTEST” (crossbar test) to all AT#s, and it uses “ACKCBT” to check the acknowledge.
- AT#s use “CHKCBT” to check the command, and use “ACKCBT” to return the acknowledge.

This first handshake has two goals :

- Check the command/acknowledge signals between ATC and all AT#s.
- Make AT#s coordinate the multiplexer selection, to fit the crossbar test.

If this first handshake is broken, due to a fault (such as the signal ACK stuck at one : SA1), ATC and all AT#s enter the end state “ERRATC” or “ERRAT#”. The router test ends with KO. The whole router is considered as faulty. The router and all associated communication channels are de-activated. Otherwise, ATC uses a set of states “CBTEST” to test the crossbar. And AT#s enter the state “CBTEST” to wait for the test result.

This table presents the value of CMD&ACK signals in the first handshake.

Signals	ATC			AT#s		
	CMDCBT	ACKCBT	ERRATC	CHKCBT	ACKCBT	ERRAT#
	Values					
CMD	10	10	00			
ACK				0	1	1

►3◄ A set of states “CBTEST” generates a set of crossbar test patterns, that are detailed in section 5.7.

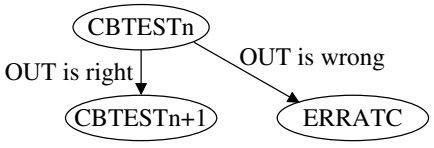


Figure 5.7 – ATC tests the crossbar with a set of states “CBTEST”.

As shown in Figure 5.7, in the state “CBTESTn”, ATC assigns the input of crossbar and observes the output of crossbar at the same time. If the output value is right, ATC enters the next state “CBTESTn+1”. Otherwise, ATC enters the end state “ERRATC”, and sends the command “NOP/KO” to all AT#s. With this message, all AT#s also enter the end state “ERRAT#”. The router test ends with KO.

►4◄ If the crossbar has passed all test patterns, it’s thus considered as fault-free. ATC and all AT#s achieve the second handshake to coordinate the multiplexer selection for the routing function test.

- ATC uses “CMDRFT” to send the command “RFTEST” (routing function test) and it uses “ACKRFT” to check the acknowledge.

- AT#s use “CBTEST” to check the command, and use “ACKRFT” to return the acknowledge.

This table presents the value of CMD&ACK signals in the second handshake.

Signals	ATC			AT#s		
	CMDRFT	ACKRFT	ERRATC	CBTEST	ACKRFT	ERRAT#
	Values					
CMD	11	11	00			
ACK				0	1	1

If the second handshake is broken, the router test will end with KO. Otherwise, ATC uses a set of states “RFTEST” to test the routing function. And AT#s enter the state “RFTEST” to wait for the test result.

►5◄ A set of states “RFTEST” generates a set of routing function test patterns, that are detailed in section 5.6.

►6◄ If the routing function has passed all test patterns, it’s considered as fault-free. The router test ends with OK. ATC and all AT#s achieve the third handshake to coordinate the multiplexer selection for the channel test. And each AT# starts to test the corresponding channel.

- ATC uses “CMDCHT” to send the command “CHTEST” (channel test) and it uses “ACKCHT” to check the acknowledge of the end of all associated channels tests.
- AT#s use “RFTEST” to check the command, and then each AT# uses a set of states “CHTEST” to test the corresponding channel. The detail of channel test is presented in section 5.8. It should be noted that, what ever the end of a channel test is, the corresponding AT# uses one end state “FUNC” or “ERRATC”, to return to ATC the “end” acknowledge.

This table presents the value of CMD&ACK signals in the third handshake.

Signals	ATC		AT#s			
	CMDCHT	ACKCHT	RFTEST	CHTEST	FUNC	ERRAT#
	Values					
CMD	01	01				
ACK			0	0	1	1

After the third handshake has successfully go through, ATC uses the state “FUNC” to activate the router, and AT#s activate the fault-free channels and de-activate the faulty channels. Thus, the whole test process is terminated, and the activated components will operate the normal function.

►7◄ A timeout mechanism implemented in each router, forces the completion of the router test and the channel test. This will lead to two possible results.

- If a timeout has occurred before the third handshake, the timeout mechanism will enforce ATC with the state “ERRATC”, and enforce AT#s with the “ERRAT#”. The router test finally ends with KO.
- If a timeout has occurred during the third handshake, the timeout mechanism will enforce ATC with the state “FUNC”, and enforce no-end AT#s with the “ERRAT#”. The corresponding channel test finally ends with KO.

►8◄ What ever the end state is, ATC and all AT#s keep their end states until the next system reboot or chip power-on.

In the following sections, we will detail test pattern generation for : routing function test, crossbar test and channel test.

5.6 Test Pattern Generation for Routing Function Test

The routing function module is a combinational circuit. Its test patterns are directly generated by Synopsis TetraMAX, an ATPG tool.

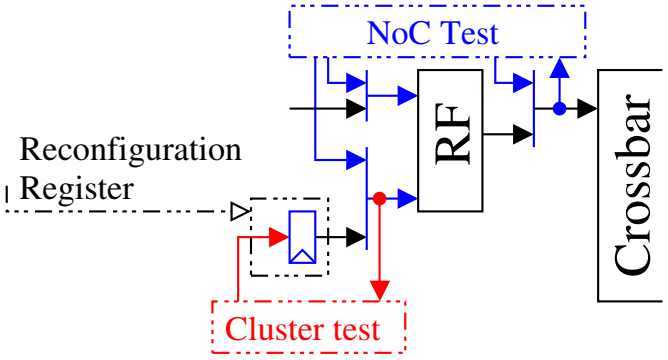


Figure 5.8 – The full test of a reconfigurable routing function.

As shown in Figure 5.8, the full test of a reconfigurable routing function is presented. This full test contains the test for the routing function (RF) and the test for the reconfiguration register. The NoC test doesn’t address the (write&read) test of reconfiguration register, which will be achieved in the cluster test.

As shown in Figure 5.9, with the help of enabling some options (blue rectangles) in TetraMAX, we can generate a set of test patterns, dedicated to SAF (Stuck-at Fault) model, to reach 100% fault coverage. Moreover, we can limit the number of patterns (red rectangle).

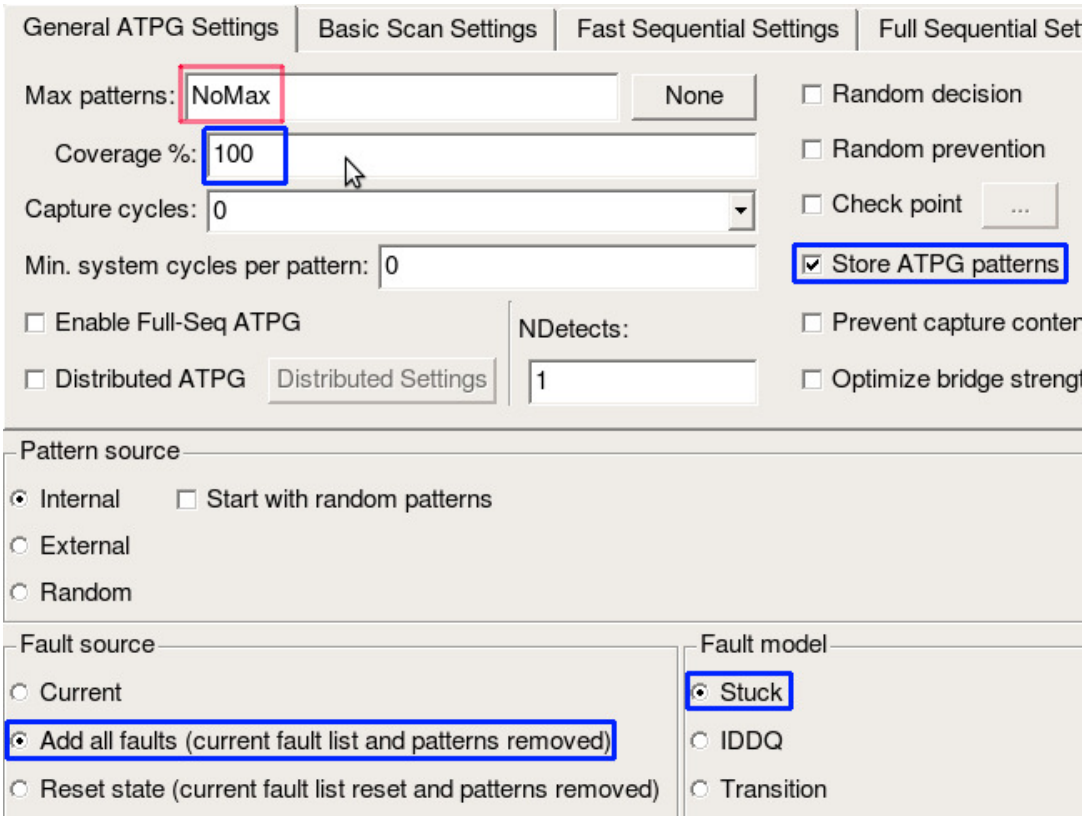


Figure 5.9 – Some important TetraMAX options for the test pattern generation.

According to our experimental work, we obtain 52 test patterns, getting 100% fault coverage. The ATPG information is listed in Listing 5.1. These 52 patterns are converted to 52 “RFTEST” states, inserted in the ATC FSM.

```
        Uncollapsed Stuck Fault Summary Report
-----------------------------------------------
fault class                     code   #faults
-----------------------------------------------
Detected                        DT       1464
Possibly detected               PT          0
Undetectable                    UD          0
ATPG untestable                 AU          0
Not detected                    ND          0
-----------------------------------------------
total faults                             1464
fault coverage                         100.00%
-----------------------------------------------
```

Listing 5.1 – ATPG information of test pattern generation for the routing function.

5.7 Test Pattern Generation for Crossbar Test

The crossbar module is a complex sequential circuit, its test patterns can hardly be generated by ATPG tool. Therefore, we propose an ad-hoc method to generate the test

patterns, considering the characteristics of crossbar function and structure. The principle of this method is to determine the form and the value of test pattern by the analysis of component's function and structure, and to evaluate the fault coverage by fault simulation. In the following subsection, we present the crossbar's interface and function characteristics.

5.7.1 Characteristics of Crossbar Function

A generic full (5 borders) crossbar (Figure 5.10) is composed of 5 input ports & 5 output ports. These input/output ports are connected to the corresponding input/output FIFOs, with flow control and data signals. The data path is actually a set of five (4to1) multiplexers located at each output port. In addition, each input port contains a "Req" signal controlled by the corresponding routing function (RF) module.

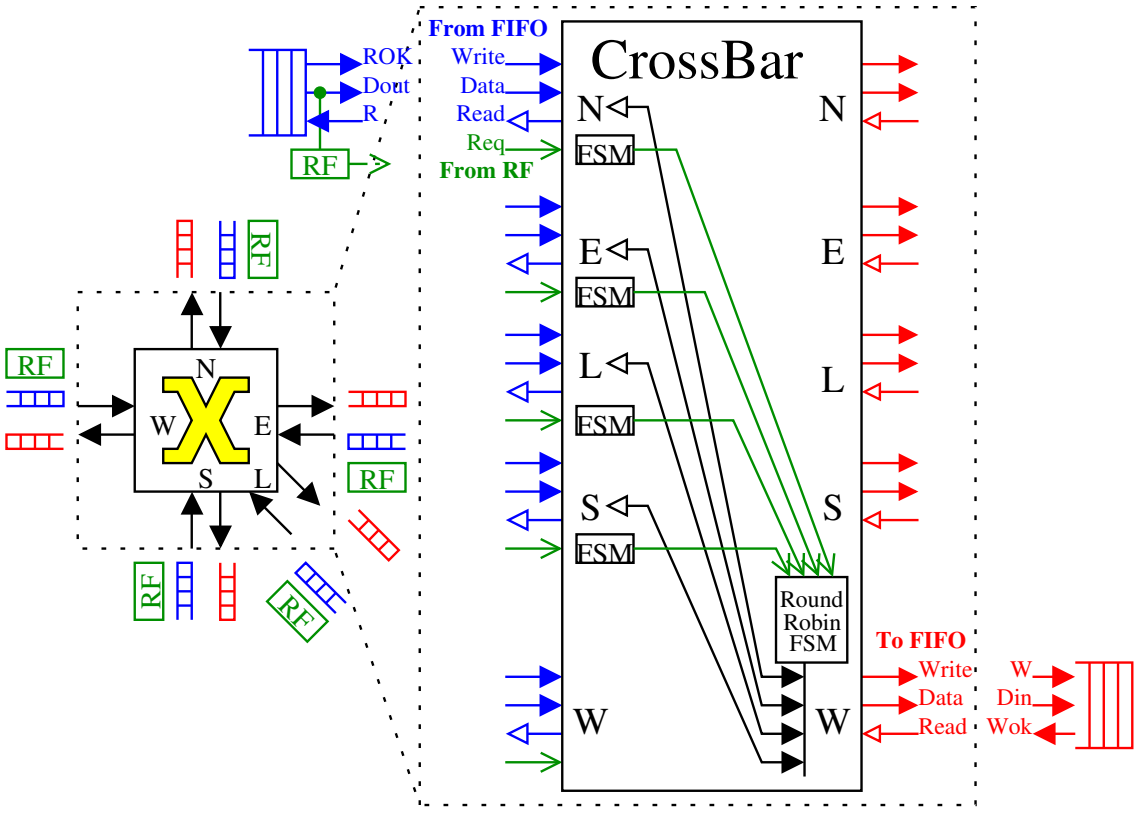


Figure 5.10 – A generic crossbar.

As presented in chapter 3 section 3.3, the router function is to route the packet from an input FIFO to an output FIFO (different border), according to the routing algorithm. The crossbar module function is to create a data-flow path from an input port to an output port. We detail such function with the example of input/output ports : IPN [1], IPE, IPL, IPS and OPW of Figure 5.10 :

1. IP# means the Input Port of # border, OP# means the Output Port of # border.

►1◄ When some packets arrive at the input ports of crossbar (IPN, IPE, IPL, IPS), the corresponding RF modules will analyze the HoP (Header of Packet) flit, and then use the Req signals to send the access request to the target output port (OPW).

►2◄ Beside OPW, the round-robin FSM (Finite State Machine) determines which access request should be accepted. Once an access request (from IPN) is accepted. The data-flow path is created between IPN and OPW, where the Read, Data and Write signals of IPN and OPW are connected.

►3◄ Beside IPN, the req-enable/-disable FSM disables the Req signal, once the data-flow path is created.

►4◄ The packet is transmitted flit by flit through the data-flow path.

►5◄ The packet transmission can be suspended/resumed depending on the input/output FIFO states.

- Once the output FIFO of OPW becomes full (WOK is 0), the packet transmission is suspended (Read of IPN becomes 0). This read suspension is resumed (Read of IPN becomes 1) when the output FIFO of OPW becomes no full (WOK is 1).
- Once the input FIFO of IPN becomes empty (ROK is 0), the packet transmission is suspended (Write of OPW becomes 0). This write suspension is resumed (Write of OPW becomes 1) when the input FIFO of IPN becomes no empty (ROK is 1).

►6◄ Finally, the data-flow path between IPN and OPW is closed by the EoP (End of Packet) flit.

Crossbar Function Test Summary

According to the above mentioned characteristics of crossbar functions, testing this block consists in checking :

- The creation and the closing of data-flow path between each couple of input/output port.
- The correctness of packet transmission on each data-flow path.
- The Req signal disable mechanism during the packet transmission.
- The suspension/resumption mechanism of packet transmission.

Therefore, the crossbar function must be tested through the transmission of a set of well defined dedicated packets.

5.7.2 Characteristics of Crossbar Structure

In this subsection, we present the structures of crossbar input/output ports.

Input port structure

The pseudo gate-level structure of IPN and the req-enable/-disable FSM are detailed as shown in Figure 5.11.

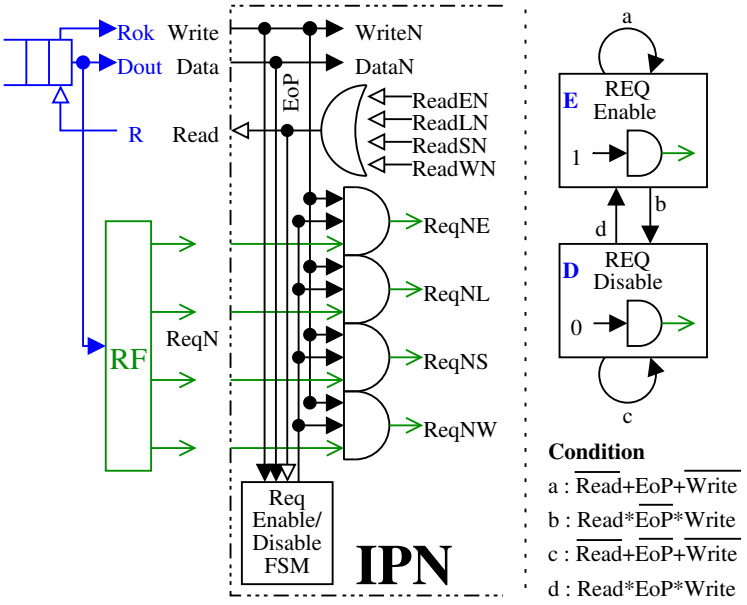


Figure 5.11 – The IPN and the req-enable/-disable FSM.

Beside the input port :

- The Write and Data signals are shared by all output ports (of the different border). They are used by the input port to present a valid (Write = 1) flit to the output ports.
- The Read signal is used by the output ports, to confirm the creation of data-flow path or to consume a flit :
  - "Read is 1" means that the data-flow path is created between the current input port and the target output port. The packet is transmitted.
  - "Read is 0" means that the data-flow path isn't yet created or the packet transmission is suspended.
- The Req signal is controlled by the RF (routing function) module. This module analyzes the input : the "packet destination" fields of the HoP flit, then requires the output port with Req signal. However, as the RF module is implemented by a simple combinational circuit (to save the silicon area), it takes the fields of every passed (valid or invalid) flit, rather than the HoP flit. In order to avoid the wrong requirement, we must disable the Req signal during the packet transmission or when the flit is invalid (Write=0).
- The req-enable/-disable FSM :

- The FSM is composed of 2 states : one (***E***) enables the Req signal and one (***D***) disables the signal.
- The state ***E*** is the idle state.
- When the data-flow path is created to transmit multi-flits packet, FSM enters ***D*** to disable the Req signal.
- When the data-flow path is created to transmit one-flit packet, FSM stays at ***E***.

The structure of output port & round-robin FSM

The pseudo gate-level structure of OPW and the round-robin FSM are described in Figure 5.12.

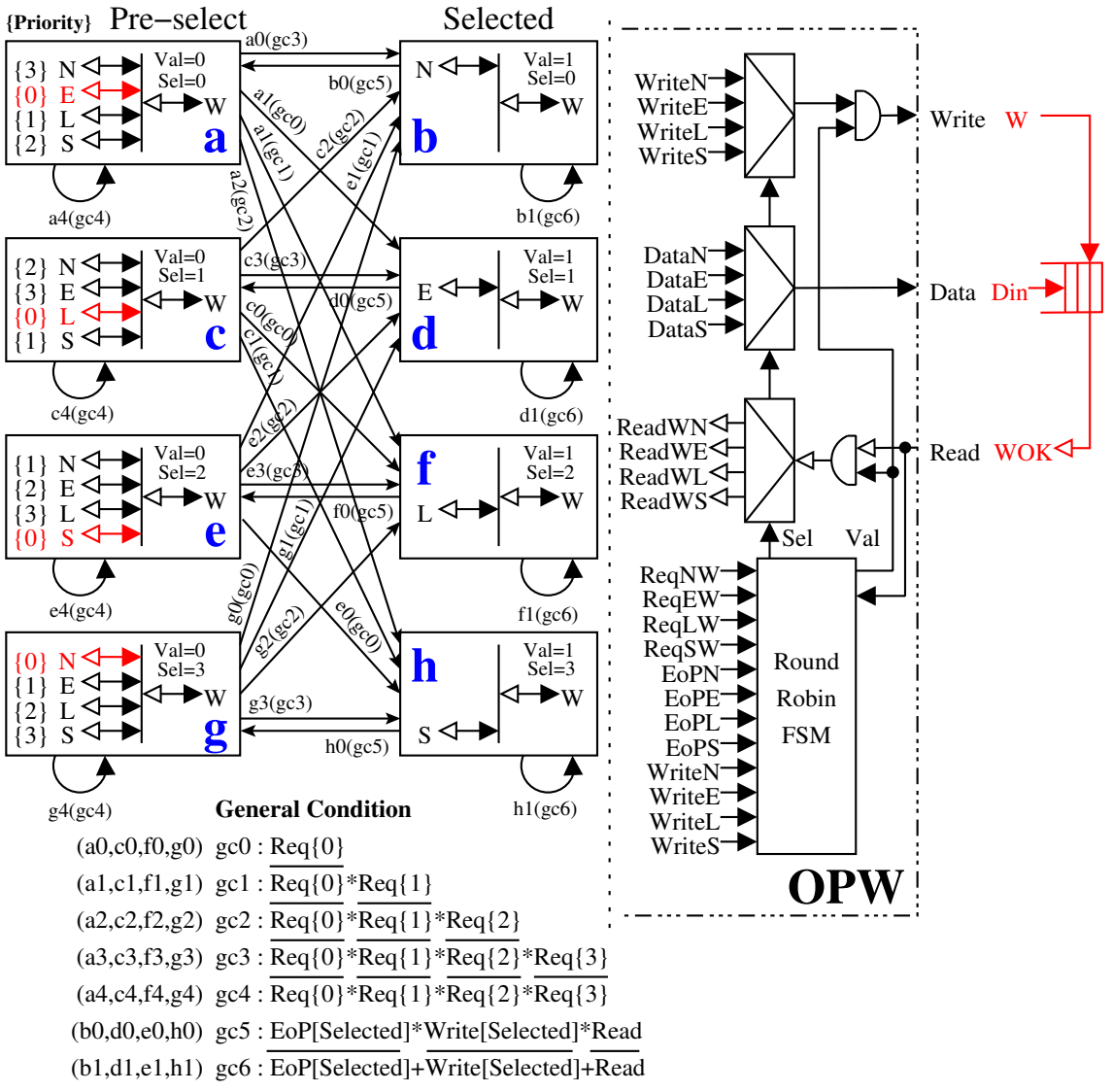


Figure 5.12 – The OPW and the round-robin FSM.

- Each output port of crossbar can be allocated to 4 input ports, two 4to1 multiplexers (for Data and Write signals) and one 1to4 demultiplexer (for Read signal) are thus used to guaranty one-to-one access.
- In order to give each input port a fair chance to access the output, the (multiplexer/demultiplexer) selection is thus determined by the round-robin FSM.
  - The round-robin FSM is composed of 8 states, 4 Pre-select states (***a,c,e,g***), 4 Selected states (***b,d,f,h***).
  - State ***a*** is the idle state.
  - Each state determines a value for the (multiplexer/demultiplexer) selection.
  - Each Pre-select state disables (Val=0) the Read signal and the Write signal.
  - Each Pre-select state provides an access priority for accessible input ports. This priority is enforced by a set of transition conditions.
    For example, ***a*** enforces the access priority : {0}=E, {1}=L, {3}=S, {4}=N. The corresponding transition conditions enforce that :
    -a0(gc0)- OPW accepts IPE, when IPE requires.
    -a1(gc1)- OPW accepts IPL, when IPL requires but IPE doesn't.
    -a2(gc2)- OPW accepts IPS, when IPS requires but IPE and IPL don't.
    -a3(gc3)- OPW accepts IPN, when IPN requires but others don't.
  - A transition from a Pre-select state to a Selected state means that the access request of an input port is accepted, OPW is allocated to the corresponding input port.
  - Each Selected state enables (Val=1) the Read signal and the Write signal to create the data-flow path.
  - A transition from a Selected state to a Pre-select state means that OPW is released by a valid EoP.

  In brief, the strategy the round-robin FSM uses is : when the output port has been released from an input port, this input port is given the lowest access priority at the next time of request. For example : $\boldsymbol{a} \rightarrow \boldsymbol{d}$ and $\boldsymbol{d} \rightarrow \boldsymbol{c}$, where ***c*** provides the new access priority : {0}=L, {1}=S, {3}=N, {4}=E.

Crossbar Structure Test Summary

Finally, we can summarize the test objectives for the crossbar :

For each input port :

- Test each transition of req-enable/-disable FSM
- Test the "OR" gate for Read signal
- Test the "AND" gate for Req signals

For each output port :

- Test each transition of round-robin FSM
- Test the 4to1 multiplexers for Data and Write signals
- Test the 1to4 demultiplexer for Read signals
- Test the "AND" gate for Write and Read signals.

5.7.3 Crossbar Test Scenario

Definition 1 : **A packet transmission** means a procedure of transmitting a packet from the crossbar input port to the crossbar output port. This procedure will require several (N) clock cycles. Therefore, a packet transmission corresponds to several (N) test patterns.

Definition 2 : **A test pattern** means, at a clock cycle, assigning values to crossbar input and observing values at crossbar output. The test patterns will be stored in ATC FSM and applied during the "CBTEST" states.

Summary of Crossbar Test Scenario

We summarize here the principle features of the crossbar test scenario, and detail them in the following document :

- ► The crossbar test scenario is composed of five stages.
- ► At each stage, the test target is one output port.
- ► Each output port is tested by the ***16*** dedicated packets transmissions.
- ► Two basic flits : X : (00...00) and ¬X : (11...11) are used to constitute the ***16*** dedicated packets.
- ► The first ***4*** dedicated packets are composed of two flits (HoP : X, EoP : ¬X).
- ► Each two-flits packet transmission requires ***6*** clock cycles. It corresponds to ***6*** test patterns.
- ► The last ***12*** dedicated packets are composed of one-flit (EoP : ¬X).
- ► Each one-flit packet transmission requires ***2*** clock cycles. It corresponds to ***2*** test patterns.
- ► In brief, at one stage, the ***16*** dedicated packets transmissions correspond to $48 = 4 \times 6 + 12 \times 2$ test patterns. Therefore, the crossbar test scenario corresponds to $240 = 5 \times 48$ test patterns.

An example : the *16* dedicated packets transmissions for the OPW test

The table 5.2 presents the ***16*** dedicated packets transmissions for the OPW test.

Step N°	Target FSM Transition Condition	Assigning								Observing
		IPN		IPE		IPL		IPS		OPW
		Req	Data	Req	Data	Req	Data	Req	Data	Data
1	a → d (a0) d → c (d0)	W	¬X X	W	X ¬X	W	¬X X	W	¬X X	X ¬X
2	c → f (c0) f → e (f0)	W	¬X X	W	¬X X	W	X ¬X	W	¬X X	X ¬X
3	e → h (e0) h → g (h0)	W	¬X X	W	¬X X	W	¬X X	W	X ¬X	X ¬X
4	g → b (g0) b → a (b0)	W	X ¬X	W	¬X X	W	¬X X	W	¬X X	X ¬X
5	a → b (a3) b → a (b0)	W	¬X		X		X		X	¬X
6	a → f (a1) f → e (f0)	W	X		X	W	¬X	W	X	¬X
7	e → f (e3) f → e (f0)		X		X	W	¬X		X	¬X
8	e → b (e1) b → a (b0)	W	¬X	W	X	W	X		X	¬X
9	a → h (a2) h → g (h0)	W	X		X		X	W	¬X	¬X
10	g → h (g3) h → g (h0)		X		X		X	W	¬X	¬X
11	g → d (g1) d → c (d0)		X	W	¬X	W	X	W	X	¬X
12	c → d (c3) d → c (d0)		X	W	¬X		X		X	¬X
13	c → h (c1) h → g (h0)	W	X	W	X		X	W	¬X	¬X
14	g → f (g2) f → e (f0)		X		X	W	¬X	W	X	¬X
15	e → d (e2) d → c (d0)		X	W	¬X	W	X		X	¬X
16	c → b (c2) b → a (b0)	W	¬X	W	X		X		X	¬X

Table 5.2 – The ***16*** dedicated packets transmissions for OPW test stage.

The details of the first 4 dedicated packets transmissions for the OPW test

Table 5.3 details the signal value per clock cycle of the first 4 dedicated packets transmissions for the OPW test stage.

			The 1st packet transmission						The 2nd packet transmission					
Clock Cycle			***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***
IPN	G	Write	1	1	0	1	1	1	1	1	0	1	1	1
		Data	¬X	¬X	¬X	¬X	X		¬X	¬X	¬X	¬X	X	
		Req	0001					0000	0001					0000
	A	Read	0					0	0					0
IPE	G	Write	1	1	1	0	1	1	1	1	0	1	1	1
		Data	X	X	X	X	¬X		¬X	¬X	¬X	¬X	X	
		Req	0001	0001	1111	1111	1111	0000	0001					0000
	A	Read	0	1	0	1	1	0	0					0
IPL	G	Write	1	1	0	1	1	1	1	1	1	0	1	1
		Data	¬X	¬X	¬X	¬X	X		X	X	X	X	¬X	
		Req	0001					0000	0001	0001	1111	1111	1111	0000
	A	Read	0					0	0	1	0	1	1	0
IPS	G	Write	1	1	0	1	1	1	1	1	0	1	1	1
		Data	¬X	¬X	¬X	¬X	X		¬X	¬X	¬X	¬X	X	
		Req	0001					0000	0001					0000
	A	Read	0					0	0					0
IPW	G	Write	0						0					
		Data												
		Req	1111						1111					
	A	Read	0						0					
OPN	A	Write	0						0					
		Data												
E,L,S	G	Read	1						1					
OPW	A	Write	0	1	1	0	1	0	0	1	1	0	1	0
		Data		X	X	X	¬X			X	X	X	¬X	
	G	Read	1	1	0	1	1	1	1	1	0	1	1	1

			The 3rd packet transmission						The 4th packet transmission					
Clock Cycle			***13***	***14***	***15***	***16***	***17***	***18***	***19***	***20***	***21***	***22***	***23***	***24***
IPN	G	Write	1	1	0	1	1	1	1	1	1	0	1	1
		Data	¬X	¬X	¬X	¬X	X		X	X	X	X	¬X	
		Req	0001					0000	0001	0001	1111	1111	1111	0000
	A	Read	0					0	0	1	0	1	1	0
IPE	G	Write	1	1	0	1	1	1	1	1	0	1	1	1
		Data	¬X	¬X	¬X	¬X	X		¬X	¬X	¬X	¬X	X	
		Req	0001					0000	0001					0000
	A	Read	0					0	0					0
IPL	G	Write	1	1	0	1	1	1	1	1	0	1	1	1
		Data	¬X	¬X	¬X	¬X	X		¬X	¬X	¬X	¬X	X	
		Req	0001					0000	0001					0000
	A	Read	0					0	0					0
IPS	G	Write	1	1	1	0	1	1	1	1	0	1	1	1
		Data	X	X	X	X	¬X		¬X	¬X	¬X	¬X	X	
		Req	0001	0001	1111	1111	1111	0000	0001					0000
	A	Read	0	1	0	1	1	0	0					0
IPW	G	Write	0						0					
		Data												
		Req	1111						1111					
	A	Read	0						0					
OPN	A	Write	0						0					
		Data												
E,L,S	G	Read	1						1					
OPW	A	Write	0	1	1	0	1	0	0	1	1	0	1	0
		Data		X	X	X	¬X			X	X	X	¬X	
	G	Read	1	1	0	1	1	1	1	1	0	1	1	1

Table 5.3 – The detail of the first 4 dedicated packets transmissions for the OPW test stage. A : analysing. G : generating.

For example : the 1st dedicated packet transmission :

►Clock Cycle 1◄ "Request cycle" : make each input port (IPN, IPE, IPL and IPS) send the access request to OPW at the same time. The HoP (X) is assigned at IPE and the EoP (¬X) is assigned at IPN, IPL and IPS.

For OPW :

- The round-robin FSM stays at the state ***a***

For IPE :

- The transition ***a*** of req-enable/-disable FSM is operated
- The req-enable/-disable FSM stays at ***E***

►Clock Cycle 2◄ "Access cycle" : the data-flow path between OPW and IPE has been created. The valid HoP (X) is presented by IPE to OPW.

For OPW :

- The transition ***a0*** of round-robin FSM is operated
- FSM enters the state ***d***

For IPE :

- The transition ***a*** of req-enable/-disable FSM is operated
- FSM stays at ***E***

►Clock Cycle 3◄ "Read suspension cycle" : the read suspension is operated at OPW. And the req-disable mechanism of IPE will be tested, by assigning "1111" to the Req signal.

For OPW :

- The transition ***d1*** of round-robin FSM is operated
- FSM stays at the state ***d***

For IPE :

- The transition ***b*** of req-enable/-disable FSM is operated
- FSM enters the state ***D***

►Clock Cycle 4◄ "Write suspension cycle" : the read suspension is resumed. The write suspension is operated. And the req-disable mechanism is tested (If Write of OPN, OPL and OPS are 0, i.e. no data-flow path has been created between IPE and OPN, OPL and OPS, therefore the req-disable mechanism of IPE is detected as OK).

For OPW :

- The transition ***d1*** of round-robin FSM is operated
- FSM stays at the state ***d***

For IPE :

- The transition ***c*** of req-enable/-disable FSM is operated

- FSM stays at the state ***D***

►Clock Cycle 5◄ "Release cycle" : the write suspension is resumed. The EoP (¬X) is assigned at IPE, and the HoP (X) is assigned at IPN, IPL and IPS.

For OPW :

- The transition ***d1*** of round-robin FSM is operated
- FSM stays at the state ***d***

For IPE :

- The transition ***c*** of req-enable/-disable FSM is operated
- FSM stays at the state ***D***

►Clock Cycle 6◄ "Nop cycle" : The data-flow path between IPE and IPN has been closed. Nothing will be done at this clock cycle.

For OPW :

- The transition ***d0*** of round-robin FSM is operated
- FSM enters the state ***c***

For IPE :

- The transition ***d*** of round-robin FSM is operated
- FSM enters the state ***E***

►Clock Cycle 7◄ Start the 2nd dedicated packet transmission, this time IPL has the highest priority.

For OPW :

- The transition ***c4*** of round-robin FSM has been achieved
- FSM stays at the state ***c***

For IPL :

- The transition ***a*** of req-enable/-disable FSM is operated
- FSM stays at ***E***

.....

Conclusion of the first 4 dedicated packets transmissions

We can conclude that, with the first 4 dedicated packets transmissions :

For OPW :

- The transitions of round-robin FSM : (a0, d1, d0 and c4), (c0, f1, f0 and e4), (e0, h1, h0 and g4) and (g0, b1, b0 and a4) are operated and tested.
- The 4to1 multiplexers of Data and Write signals are tested.
  As shown in Figure 5.13, the 4to1 multiplexer of Data signals is synthesized to a set of (oa2a2a2a24) gates in the Synopsis synthesis environment with Sxlib standard cell library [Alliance, 2011]. For this kind of gate, during the first 4 dedicated

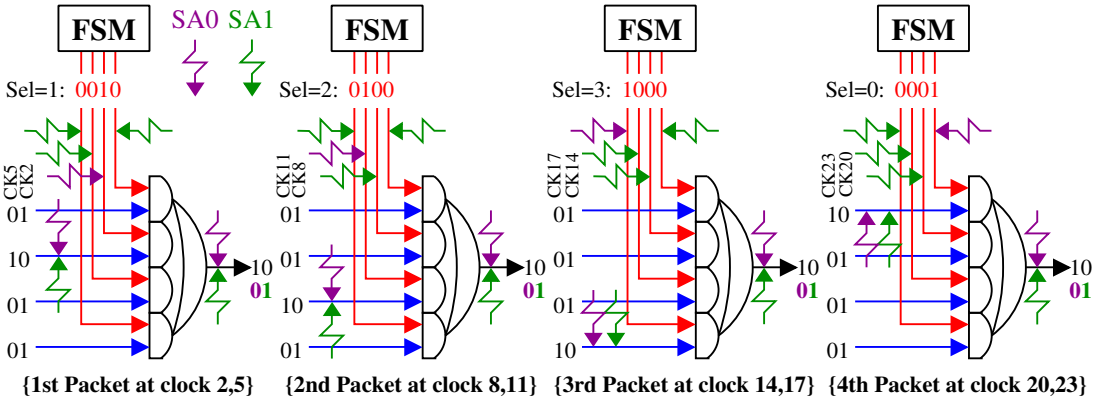


Figure 5.13 – During the first 4 dedicated packets transmissions, all SAFs of the 4to1 multiplexer can be completely tested.

packets transmissions, at clock cycles (2,5), (8,11), (14,17) and (20,23), the gate selection is done using the round-robin FSM. The selected Data input and non-selected Data input are differently assigned using packet flit (X or ¬X). Thus, all SAFs of this gate can be detected.

The same for the 4to1 multiplexer of Write signal. At clock cycles (3,4), (9,10), (15,16) and (21,22), the gate selection is done using the round-robin FSM.

- The 1to4 demultiplexer is also tested (Figure 5.14) in the same way.

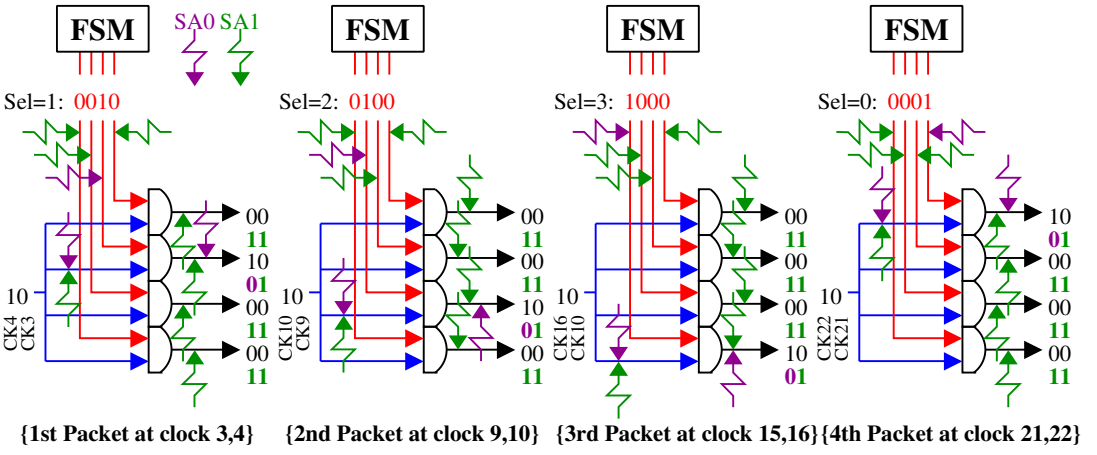


Figure 5.14 – During the first 4 dedicated packets transmissions, all SAFs of the 1to4 demultiplexer can be completely detected.

- The “AND” gates of Write and Read signals have been fully tested.

For IPN, IPE, IPL, IPS :

- The (a, b, c and d) transitions of req-enable/-disable FSM are operated and tested.
- The “OR” and “AND” gates have been partially tested. They will be fully tested at the end of crossbar test scenario.

The last 12 dedicated one-flit packets transmissions for the OPW test

Each one-flit packet transmission requires 2 clock cycles : one “Request cycle” and one “Access cycle”. Each transmission operates one transition of round-robin FSM (detailed in Table 5.2). At the end of these 12 transmissions, all remaining transitions of round-robin FSM are tested.

In conclusion, at each test stage, with the help of the 16 dedicated packets transmissions, one output port can be fully tested and four input ports can be partially tested. Then after the end of five test stages, five input/output ports are completely tested.

5.7.4 Test Patterns Implementation

The crossbar test scenario corresponds to 240 test patterns, requiring 240 clock cycles. These test patterns are generated by the ATC FSM in the “CBTEST” states. Thus, the number of “CBTEST” states is 240. These states will be used by ATC FSM to test crossbar as shown in Figure 5.7.

5.7.5 Fault Coverage Evaluation and Improving

Uncollapsed Stuck Fault Summary Report

fault **class**	code	#faults
Detected	DT	5926
Possibly detected	PT	42
Undetectable	UD	0
ATPG untestable	AU	0
Not detected	ND	40
total faults		6008
fault coverage		99.33%

Listing 5.2 – SAF coverage information of crossbar test.

The SAF coverage of the 240 test patterns is evaluated using Synopsis TetraMAX (Listing 5.2), on the synthesized version of the DSPIN full crossbar. We find out that some test patterns related to “Nop cycle” in the first 4 dedicated packets transmissions, are useless to improve the SAF coverage. Thus, we remove these patterns, and the new total number of patterns is 225.

5.8 Channel Test Process and Test Pattern Generation

The DSPIN communication channel is a point-to-point bi-synchronous data-flow path containing two FIFOs. Therefore, it can be tested through generating&pushing the test patterns at the head of the path, and pulling&observing the result at the end of the path. In our proposed BIST structure, the ATG FSM is in charge of generating&pushing test patterns, and the ATA FSM is in charge of pulling&observing flowed result. ATG/ATA cooperate together to test each channel and then to de-activate faulty one.

As the ATG/ATA FSMs work in two different clock domains, their cooperation is achieved with the help of some asynchronous handshake signals, as shown in Figure 5.4. In the following, we detail such cooperation.

5.8.1 Cooperation between a couple of ATG/ATA

The cooperation between a couple of ATG/ATA, is a ping-pong game, as shown in Figure 5.15. This game is achieved with the help of the asynchronous handshaking signals "READY" and "GO", to drive the "CHTEST" transitions round by round. At the end of the test, if all rounds go through, the channel is considered as "OK". Otherwise, once any one round is broken or timeout, the game is over, the channel is considered as "KO". It should be noted that, ATG always serves first in a round.

In this game :

►1◄ ***Round (ACT)*** is the first round. ATG activates FIFO(out) (see Figure 5.4) in state ACT and checks no WOK/SAF0 (where WOK is 1) in state ACTT. If it's OK, ATG sends "READY" in state RDYACT.

ATA waits for "READY" in state GOACT. Once "READY" is received, ATA activates FIFO(in) in state ACT and checks no ROK/SAF1 (where ROK is 0) in state ACTT. If it's OK, ATA returns "GO" in state GONOP. Once "GO" is received by ATG, the test enters the next round.

►2◄ ***Round (NOP)*** is the second round. ATG does NOP test (do nothing) and ATA verifies again the signal ROK (no self-generating fake packet, where ROK is 0).

►3◄ ***Round (PUSH/PULL)*** : In a PUSH/PULL round, ATG pushes the channel with M test patterns using M states. And then, ATA uses M states to pull and check the flowed data. If this round is "OK", ATG/ATA continue until the end of last (PUSH/PULL) round.

In this cooperation, there are three problems, how to :

► Determine the number M for each PUSH/PULL round ?

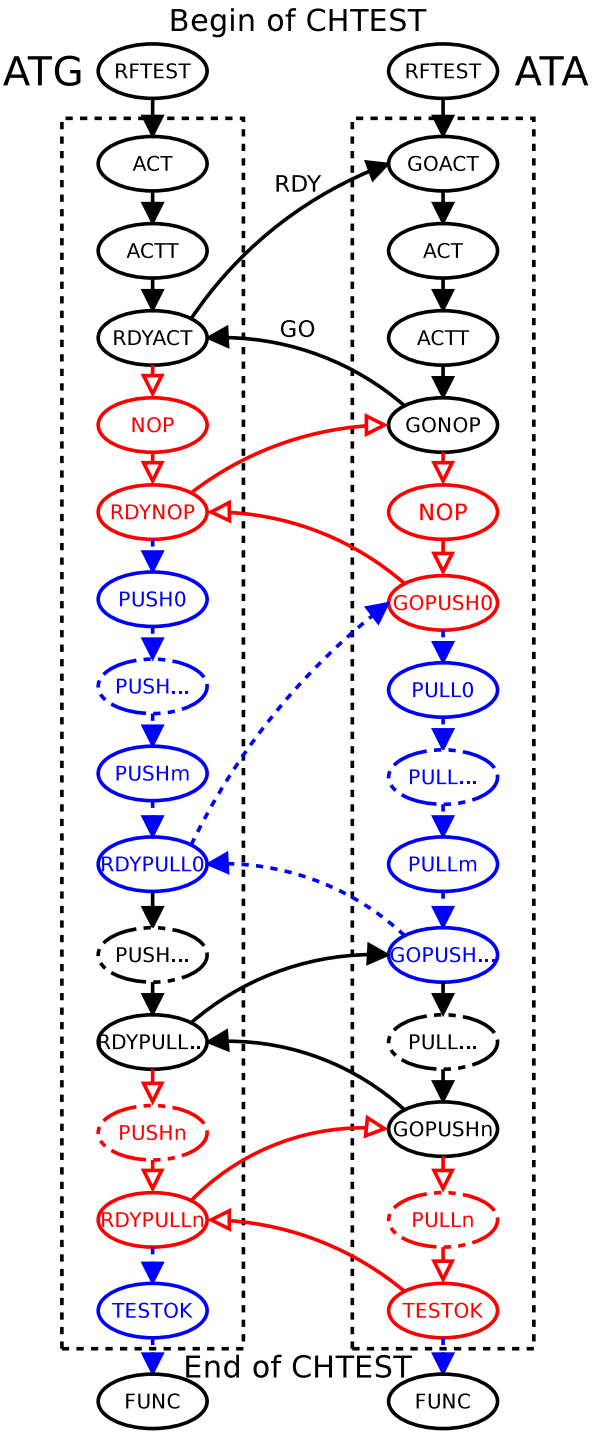


Figure 5.15 – The cooperation achieved between a couple of ATG/ATA FSM.

- Generate M test patterns for each PUSH/PULL round ?
- Determine the total number of PUSH/PULL rounds : N ?

These problems are solved by an ad-hoc method, based on the analysis of the channel function&structure characteristics.

5.8.2 Characteristics of Channel Function

The function of a DSPIN communication channel is to flow a set of ordered flits from input to output. And the goal of channel function test is not only to verify the correctness of flowed flits, but also to check the flowing order. In addition, as a channel is similar to wire-links, we will use two basic flits X {00...00} and ¬X {11...11}, to generate the dedicated test patterns, since these two flits can handle any SAF of wire-links.

5.8.3 Characteristics of Channel Structure

A communication channel is composed of two 4 words FIFOs : one synchronous and one bi-synchronous. Both of them adopt the pointer-shift structure as shown in Figure 5.16.

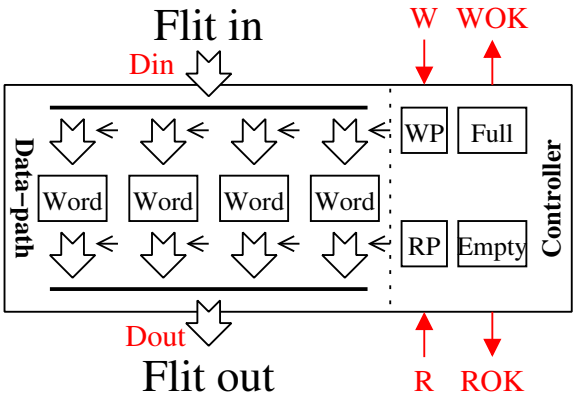


Figure 5.16 – The pointer-shift FIFO structures.

In fact, there is a large number of implementations for a pointer-shift FIFO (the related works can be found in the thesis of Ivan MIRO PANADES [Panades, 2008] chapter 4 and appendix A). In order to simplify the analysis and to propose a general ad-hoc pattern generation, we summarize the common ground of pointer-shift structure of the 4 words FIFO :

For the controller :

- The write/read pointers are initialized to point to the first word. The initializations of Full&Empty detectors make WOK set to 1 (can be written) and ROK set to 0 (cannot be read).
- When a valid flit (W=1) is presented at the Din of FIFO, the flit is written to the 1st word, and the write pointer is shifted to 2nd word. FIFO isn't yet empty, and Empty detector makes ROK set to 1 (can be read).
- After having written 4 valid flits, the value of write pointer reaches the current

value of read pointer, and FIFO is full, Full detector makes WOK set to 0 (cannot be written).

- When FIFO is full, any assigned valid flit is ignored, the write pointer cannot be shifted, and a new word cannot be written.
- When the 1st word is consumed (R=1), the read pointer is shifted to 2nd word. FIFO isn't yet full, and Full detector makes WOK set to 1.
- After having consumed 4 words, the value of read pointer reaches the current value of write pointer, and FIFO becomes empty, Empty detector makes ROK set to 0.
- When FIFO is empty, the read pointer cannot be shifted.

For the data-path :

- The Din input and the 4 words can be modeled as a 1to4 demultiplexer. The write pointer can be considered as the selection signal.
- The 4 words and the Dout can be modeled as a 4to1 multiplexer. The read pointer can be considered as the selection signal.

The test objectives for a 4 words pointer-shift FIFO are :

- Test the write/full mechanism (Filling).
- Test the read/empty mechanism (Emptying).
- Test the shift of write pointer when FIFO isn't full
- Test the shift of read pointer when FIFO isn't empty
- Test the no-shift of write pointer when FIFO is full
- Test the no-shift of read pointer when FIFO is empty
- Test the input 1to4 demultiplexer
- Test the no-written word when FIFO is full
- Test the output 4to1 multiplexer

Taking the experience of crossbar test, 1to4 multiplexer is fully tested when the selection is done and the input is assigned with (X and $\neg$X) ; 4to1 multiplexer can be fully tested when the selection is made and the selected input and no-selected input have different values (X or $\neg$X). Therefore, we propose the following FIFO test scenario.

5.8.4 FIFO Test Scenario

- ► Test scenario is composed of 11 stages : 8 "filling/emptying" stages and 3 "pointer shift" stages.
- ► At each "filling/emptying" stage, the 4 words of FIFO are respectively filled/emptied for one time.
- ► After 2nd, 4th and 6th "filling/emptying" stage, the write/read pointers are shifted for

one step, in a “pointer shift” stage.

Clock Cycle		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Test Stage		1st filling/emptying stage										2nd filling/emptying stage										1st stage of	
Action		1st filling					1st emptying					2nd filling					2nd emptying					pointer-shift	
G	W	1					0					1					0					1	0
	R	0					1					0					1					0	1
	Din	X	¬X	¬X	¬X	¬X						¬X	X	X	X	X							
A	WOK	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1
	ROK	0	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0	0	1
	Dout						X	¬X	¬X	¬X	X						¬X	X	X	X	¬X		
I	WP	1	2	3	4	1	1					1	2	3	4	1	1					1	2
	RP	1					1	2	3	4	1	1					1	2	3	4	1	1	1

Clock Cycle		23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
Test Stage		3rd filling/emptying stage										4th filling/emptying stage										2nd stage of	
Action		3rd filling					3rd emptying					4th filling					4th emptying					pointer-shift	
G	W	1					0					1					0					1	0
	R	0					1					0					1					0	1
	Din	X	¬X	¬X	¬X	¬X						¬X	X	X	X	X							
A	WOK	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1
	ROK	0	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0	0	1
	Dout						X	¬X	¬X	¬X	X						¬X	X	X	X	¬X		
I	WP	2	3	4	1	2	2					2	3	4	1	2	2					2	3
	RP	2					2	3	4	1	2	2					2	3	4	1	2	2	2

Clock Cycle		45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66
Test Stage		5th filling/emptying stage										6th filling/emptying stage										3th stage of	
Action		5rd filling					5rd emptying					6th filling					6th emptying					pointer-shift	
G	W	1					0					1					0					1	0
	R	0					1					0					1					0	1
	Din	X	¬X	¬X	¬X	¬X						¬X	X	X	X	X							
A	WOK	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1
	ROK	0	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0	0	1
	Dout						X	¬X	¬X	¬X	X						¬X	X	X	X	¬X		
I	WP	3	4	1	2	3	3					3	4	1	2	3	3					3	4
	RP	3					3	4	1	2	3	3					3	4	1	2	3	3	3

Clock Cycle		67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86
Test Stage		7th filling/emptying stage										8th filling/emptying stage									
Action		7rd filling					7rd emptying					8th filling					8th emptying				
G	W	1					0					1					0				
	R	0					1					0					1				
	Din	X	¬X	¬X	¬X	¬X						¬X	X	X	X	X					
A	WOK	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0	0	1	1	1	1
	ROK	0	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0
	Dout						X	¬X	¬X	¬X	X						¬X	X	X	X	¬X
I	WP	4	1	2	3	4	4					4	1	2	3	4	4				
	RP	4					4	1	2	3	4	4					4	1	2	3	4

Table 5.4 – Test scenario for FIFO. G : generating, A : analyzing and I : inner pointer. WP : write pointer. RP : read pointer.

The synchronous timing detail of FIFO test scenario is presented in Table 5.4. We detail here the stages of the 1st & 2nd “filling/emptying” and the 1st “pointer shift” :

► In the 1st filling/emptying stage

- In the 1st "filling", the 4 words of FIFO are filled, where 1st word is X, 2nd word is ¬X, 3rd word is ¬X and 4th word is ¬X. To write each word, the selection of 1to4 demultiplexer is made, and the input is assigned with X or ¬X. In other words, a part of all SAFs of 1to4 demultiplexer is tested.
- When FIFO is full, another flit is presented to FIFO to verify that the 1st word cannot be written, and the write pointer cannot be shifted.
- In the 1st "emptying", the 4 words of FIFO are emptied with 4 consuming cycles. To consume each word, the selection of 1to4 multiplexer is done, and the inputs are have different values (X and ¬X). That is to say, a part of all SAFs of 4to1 multiplexer is tested.
- When FIFO is empty, another consumption is achieved to verify that the read pointer cannot be shifted.

► In the 2nd "filling/emptying" stage

- In the 2nd "filling", the 4 words of FIFO are filled, where 1st word is ¬X, 2nd word is X, 3rd word is X and 4th word is X. And the test of 1to4 demultiplexer continues.
- When FIFO is full, we reverify that the 1st word cannot be written, and the write pointer cannot be shifted.
- In the 2nd "emptying", the 4 words of FIFO are emptied with 4 consuming cycles. And the test of 4to1 multiplexer continues.
- When FIFO is empty, we reverify that the read pointer cannot be shifted.

► In the 1st "pointer shift" stage,

- The write/read pointers are shifted for one step. The next "filling/emptying" stage will begin with the incremented write/read pointers. At the end, the shift of write/read pointer is fully tested.

At the end of these 11 (8+3) stages, the FIFO is tested not only at functional level, but also at structural level. Based on these stages, we propose a channel test scenario.

Channel Test Scenario

From the structural point of view, a channel is composed of two FIFOs. Each FIFO must be tested using the 8 "filling/emptying" stages and the 3 "pointer shift" stages. Therefore, we design 4 "filling/emptying" steps and 3 "pointer shift" steps, as the channel test scenario. These steps are detailed in table 5.5.
In this table :

► In a "filling/emptying" step, each FIFO is respectively filled/emptied two times.

► 4 steps of "filling/emptying", can completely achieve the 8 stages of "filling/empty-

Step	Pointer		Flits filled	FIFO-out/-in				Flit Emptying
	WP(O/I)	RP(O/I)	Channel Input	Word0	Word1	Word2	Word3	Channel Output
1	1,1	1,1	X,¬X,¬X,¬X,¬X,X,X,X,(X)	X,¬X	¬X,X	¬X,X	¬X,X	X,¬X,¬X,¬X,¬X,X,X,X,(¬X)
2	One Step Pointer Shift							
3	2,2	2,2	X,¬X,¬X,¬X,¬X,X,X,X,(X)	¬X,X	X,¬X	¬X,X	¬X,X	X,¬X,¬X,¬X,¬X,X,X,X,(¬X)
4	One Step Pointer Shift							
5	3,3	3,3	X,¬X,¬X,¬X,¬X,X,X,X,(X)	¬X,X	¬X,X	X,¬X	¬X,X	X,¬X,¬X,¬X,¬X,X,X,X,(¬X)
6	One Step Pointer Shift							
7	4,4	4,4	X,¬X,¬X,¬X,¬X,X,X,X,(X)	¬X,X	¬X,X	¬X,X	X,¬X	X,¬X,¬X,¬X,¬X,X,X,X,(¬X)

Table 5.5 – Details of the channel test scenario and the test patterns.

ing" for each FIFO test.

- In each "filling/emptying" step, the chain of filled flits contains 9 flits : 8 are pushed to the FIFO ; another one is used to check that the word cannot be written, and the write pointer cannot be shifted, when channel is full. The chain of observed flits contains 9 flits : 8 are pulled from FIFO ; another one is used to check that the read pointer cannot be shifted, when channel is full.
- 3 steps of "pointer shift", can completely achieve the 3 "pointer shift" stages for each FIFO test.

In conclusion, these steps have tested a channel since they have completely achieved test scenario for each FIFO. With these steps, we have solved the problems defined in section 5.8.1.

- **For ATG** :
  - N : the total number of PUSH round is 7 (4+3) : 4 for "filling" channel, 3 for "write pointer shift".
  - M : the number of states of a "filling" PUSH round is 9 (8+1).
  - M : the number of states of a "write pointer shift" PUSH round is 1.
  - Test patterns for each PUSH round are shown in table 5.5.
- **For ATA** :
  - N : the total number of PULL round is 7 (4+3) : 4 for "emptying" channel, 3 for "read pointer shift".
  - M : the number of states of a "emptying" PULL round is 9 (8+1).
  - M : the number of states of a "read pointer shift" PULL round is 1.
  - Test patterns for each PULL round are shown in table 5.5.

Thus, the channel test process has been defined. The "CHTEST" states of ATG/ATA FSMs are completely designed.

5.8.5 Channel Test Process Verification

In order to verify the channel test process, we developed a VHDL RTL model containing one ATG FSM, one ATA FSM and a bi-synchronous channel, as shown in Figure 5.17.

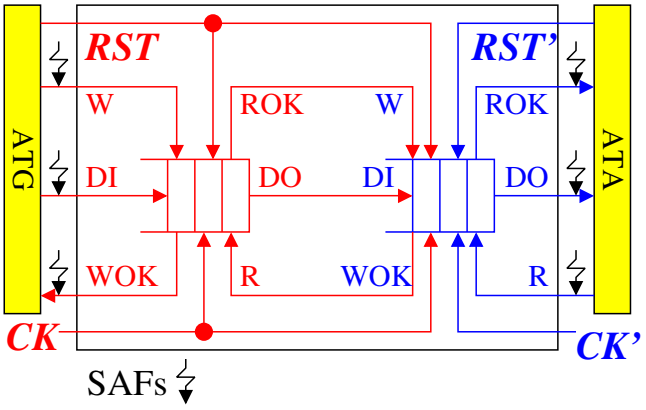


Figure 5.17 – A VHDL RTL model containing one ATG, one ATA and a bi-synchronous channel.

We simulate this model with 7 different couples of clocks, as listed in Table 5.6. In these simulations, we observed that ATG and ATA have gone through with these clocks couples. This result demonstrates that the channel test process is robust against large variations of clock boundaries, it supports the GALS approach.

CK of ATG	500ns	111ns	7ns	5ns	5ns	5ns	5ns
CK of ATA	5ns	5ns	5ns	5ns	7ns	111ns	500ns

Table 5.6 – Clock couples.

5.8.6 Fault Coverage Evaluation

Using Synopsis TetraMAX on synthesized version of a synchronous channel, we evaluate the fault coverage of the channel test patterns. The result is presented in Listing 5.3.

It should be noted that, PT means possibly detected faults. For the channel test, PT is large because there is a large number (almost 320) of undefined-register faults as shown in Figure 5.18. In fact, for a real physical circuit, the X signal - the output of an undefined-register - has a real value 0 or 1. Thus, the X signal is seem to be injected by SA0 or SA1. As the SA0 and SA1 have been tested for signal X, all PT faults are considered as detected faults.

```
Uncollapsed Stuck Fault Summary Report
-------------------------------------------------------
fault class                    code    #faults
-------------------------------------------------------
Detected                       DT         5912
Possibly detected              PT          355
Undetectable                   UD            0
ATPG untestable                AU            0
Not detected                   ND           91
-------------------------------------------------------
total faults                              6358
fault coverage                           98.57%
-------------------------------------------------------
```

Listing 5.3 – SAF coverage information of channel test.

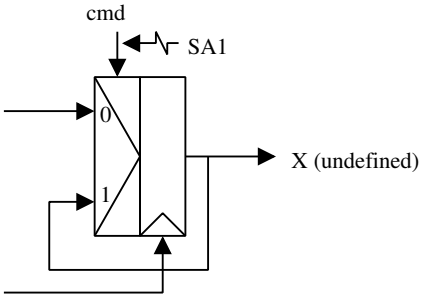


Figure 5.18 – The undefined-register faults due to a SAF.

5.9 Timeout Value

The timeout value is determined by the ratios between the various clock frequencies involved in the NoC. If all clocks have the same frequencies (but different phases), the initialization procedure requires less than 400 clock cycles (the total number of test patterns is $383 = 52_{\text{ For Routing Function Test}} + 225_{\text{ For Crossbar Test}} + 106_{\text{ For Channel Test}}$). If we assume that the biggest ratio between two clock frequencies in two neighboring routers is equal to 100 (1 Ghz and 10 Mhz), the upper bound timeout value is equal to 400×100 local clock cycles. That is to say, this timeout value is implemented in the timer of each router, counted down with the local clock.

5.10 Experimental Result

In this section, we detail evaluations of the proposed 2D-Mesh NoC BIST. These evaluations are done from the point of view of SAF (Stuck-at Fault) coverage, and from the point of view of silicon area overhead.

5.10.1 Fault Coverage Evaluation

We launch a fault simulation of the complete self testable router in the Tetramax environment. In this evaluation, the ATC, ATG, ATA FSMs and the multiplexers are considered as the router components, and the final states of the FSMs are considered as the test signature, as shown in Figure 5.19.

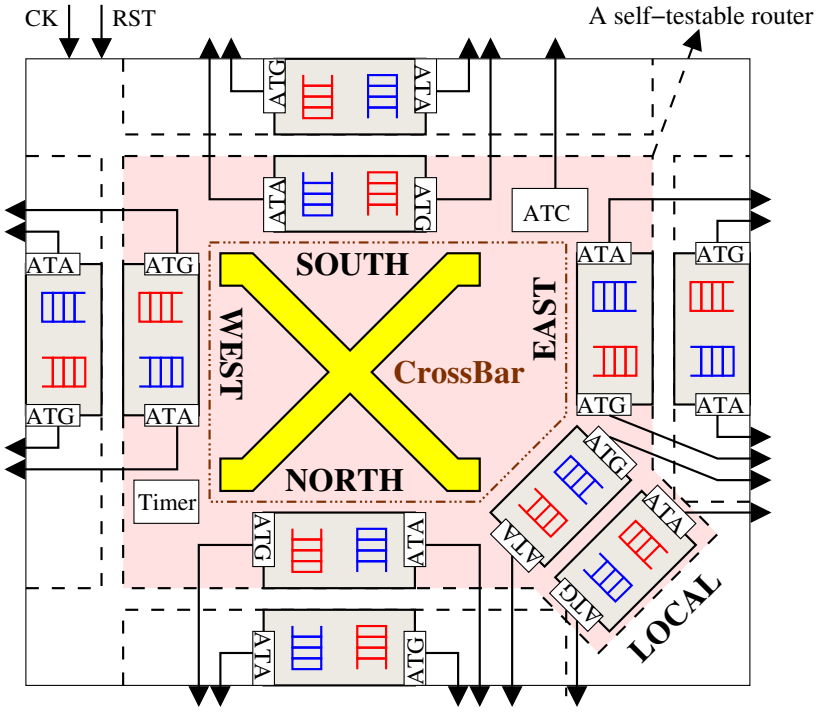


Figure 5.19 – The self testable router.

The fault coverage result is presented in Listing 5.4. The overall SAF coverage of the BIST is 90.95%.

Uncollapsed Stuck Fault Summary Report

fault class	code	#faults
Detected	DT	63551
Possibly detected	PT	2500
Undetectable	UD	8
ATPG untestable	AU	0
Not detected	ND	6561
total faults		72620
fault coverage		90.95%

Listing 5.4 – Fault coverage evaluation of self testable router.

A summary of fault coverage evaluation of each component is presented in Table below.

Components	Fault Coverage
Routing Function	100%
Full Crossbar	99.33%
Channel	98.57%
A Self Testable router (Whole NoC)	90.95%

5.10.2 Cost (extra silicon area) :

The cost from the point of view of the silicon area is evaluated through a synthesizable VHDL model of a complete self testable router using the SXLIB standard cell library [Alliance, 2011], in the Synopsis synthesis environment.

Component	NAND	%
Original DSPIN Router	13500.75	100%
Self Testable Router	20318.1	150.50%
Overhead	6817.35	50.50%

The router (by extension the NoC itself) footprint is increased by 50.50% (that is much smaller than 200%, the overhead due to TMR : Triple Modular Redundancy). This is an affordable cost, since the DSPIN NoC is a very compact design and represents typically less than 3% [Panades et al., 2008] of the silicon area in a typical MP^{2}SoC architecture.

5.10.3 NoC Test Execution Time

The NoC test execution time is determined by the timeout value, that is 4×10^4 local clock cycles. For example, at a clock 500Mhz, the NoC test lasts 0.00008 seconds.

5.11 Conclusion

In this chapter, we detail the proposed fully distributed off-line BIST dedicated to 2D-Mesh NoC in a GALS context. The (BIST) test process is systematically integrated in the NoC as an initialization procedure. This procedure provides automatic detection and de-activation of the faulty routers or communication channels. And it provides a global SAF coverage about 91%, for an acceptable silicon area overhead.

The presented initialization procedure can be performed at fabrication time or “on the field”. In the first case, this procedure is helpful in improving the yield by avoiding

to throw the whole chip when one single component is faulty. In the second case, it's useful to avoid the failure propagation and offer cleaning the malfunctions of the overall Network-on-Chip through a simple reboot of the chip. Thus it makes the NoC ready for self-reconfiguration.

Moreover, the infrastructure : algorithm of test process, test structure, de-activation mechanism and FSMs, can be reused in another reconfigurable or (permanent) fault-tolerant Network-on-Chip, that have the characteristics presented in chapter 3 section 3.5.

Chapter 6

Configuration Infrastructure

Contents

In this chapter, we present the DCCI (Distributed Cooperative Configuration Infrastructure) that has been implemented in a DSPIN-based, shared memory MP²SoC architecture by REFAUVELET Dimitri during its ongoing thesis. We detail also, on top of DCCI, the software-based diagnosis/localization of faulty/fault-free components, which is a software application for "Black Hole" detection. The evaluation of this software, from the points of view of detection coverage, execution times and application code size, are presented and analyzed.

6.1 DCCI Implementation in the DSPIN-based, shared memory MP²SoC Architecture

The final goal of DCCI is to create a spanning communication tree, after the initialization procedure of the DSPIN micro-network. This goal is reached through a distributed

embedded Configuration Firmware (CF).

As shown in Figure 3.1.{B}, in the DSPIN-based, shared memory MP^{2}SoC architecture, each cluster contains an embedded ROM, that stores the local CF code. This code is composed of three parts : cluster test, tree creation and "Black Hole" detection.

6.1.1 Cluster Test

After each system reboot or chip power-on, each cluster has to test itself executing the first part of local CF code, before it can try to participate to build the DCCI communication tree. In each cluster, the local test starts only after the local NoC initialization procedure goes through. This local test has 3 stages :

- **Local intra-cluster test :** It is a first, coarse grain, software-based test (such as presented in [Gizopoulos et al., 2004]) for all IPs belonging to the cluster, to decide if a cluster is usable to participate to the tree building procedure. For instance, a cluster which RAM does not pass the march test is declared unusable.
- **Local leader election :** As each cluster can contain several processors, an operational processor of the cluster is elected. The other operational processors are put in idle state.
- **Access to the external memory :** The locally elected processor tries to establish a connection with the external I/O controller, using the default routing algorithm for the NoC : X-First.

If a cluster successfully passes the two first steps, it is declared to be usable and it's a potential DCCI tree node. If the third test successfully passes, the cluster is a potential DCCI tree root.

6.1.2 Tree Creation

In each cluster, the elected local processor will execute the second part of local CF code : tree creation, as presented in chapter 2 section 2.2.2.

6.1.3 "Black Hole" Detection

Once the DCCI tree has been built, the tree root will execute the last part of its CF code : "Black Hole" detection. This detection phase has 3 stages :

- The tree root loads, from the external storage, the code for "Black Hole" detection.

- The root uses the DCCI tree to distribute the code to each node, and then make them execute the code. In other words, each node begins to do local “Black Hole” detection.
- When all nodes have finished their local detections, the root will centralize each local detection result to obtain the final “Black Hole map”.

The final “Black Hole map” is used by the tree root, to identify the modification of topology. And the tree root can then analyse how to configure the routing function of network, to fit the modification.

In this thesis, we only detail one of the three parts of CF codes : the “Black Hole” detection software.

6.2 “Black Hole” Detection Software

The principal idea is to force each node to communicate with all other nodes in the network to exercise all “X-First” paths. And then, to mark as valid all components of a “non timeout” path.

6.2.1 Theory Presentation

After the DSPIN initialization procedure :

- The fault-free channels are activated.
- The fault-free routers are activated to achieve the default routing algorithm - X-First.
- The faulty channels are de-activated and configured to be a “Black Hole”.
- The faulty routers are de-activated. The associated channels are de-activated and configured to be “Black Holes”. Thus, a faulty router is considered as a “Black Hole”.

We indirectly test each component by checking each “X-First” path. As shown in Figure 6.1, a “X-First” path is presented.

A “X-First” transaction between a couple of clusters, is a round-trip. It’s composed of two half paths : one for command and one for response. For the command path, the initial head is an initiator module, such as a processor/cache couple. The final end is a target module, such as an embedded RAM. For the response path, the head is the target module, the end is the initiator module.

Each half path contains a set of routers and channels. If one of these components is a “Black Hole”, the packet transmission between initiator/target or target/initiator will be

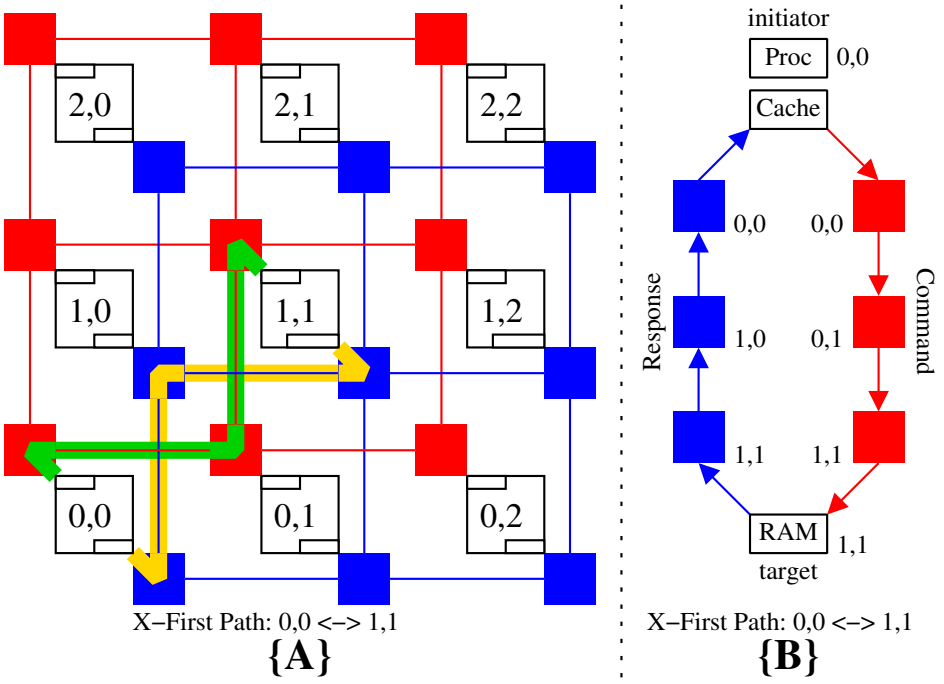


Figure 6.1 – A X-First path between cluster (0,0) and cluster (1,1).

lost. A timeout will be triggered at the initiator side, since the command/response protocol is broken.

In other words, if no timeout is set by the initiator, it proves that the path contains no "Black Hole" in both command path, and response path. And for a "no-timeout" path, all components can be marked as fault-free.

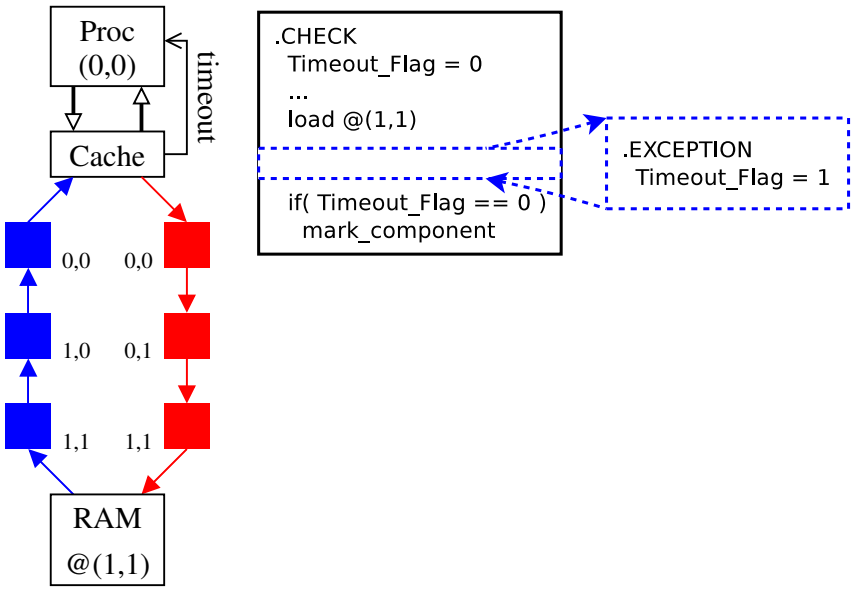
In the following subsection, we detail the "X-First" path check mechanism.

6.2.2 "X-First" Path Checking Mechanism

As shown in Figure 6.2.{a}, each processor receives a timeout exception when "cache miss" timeout occurs. This exception is the key of the "X-First" path check.

In Figure 6.2.{b}, the path check code is presented. It's executed by the processor.

- A global variable : Timeout_Flag is initialized as neuter (0).
- The processor loads a word from the embedded RAM of targeted cluster.
- The load instruction triggers a cache miss, and then the cache generates a read packet to the target RAM.
- If the "X-First" path contains a "Black Hole", the timeout will be triggered by the cache. And the processor will execute the exception code for "cache miss" timeout.
- The processor handles the exception as a non fatal error, where Timeout_Flag is assigned as positive (1). And then the processor continues the check code.



(a) (b)

Figure 6.2 – FIG.{a} presents the “X-First” path and processor/cache couple. FIG.{b} presents the code of “X-First” path check

- It should be noted that, if the path doesn’t contain any “Black Hole”, the cache will receive the response packet from the RAM. And then the processor continues the check code.
- The last step of check code is to verify Timeout_Flag to be neuter (0) or positive (1).
- If the Timeout_Flag is neuter (0) that means the path doesn’t contain any “Black Hole”. In this case, the processor will mark the path components as fault-free.

In the following subsection, we detail the algorithm for marking components.

6.2.3 Algorithm for Marking Components

The final goal of marking components is to create two lists : one is the list of FaultFree Routers (FLR), and the other one is the list of FaultFree Channels (FLC). The algorithm of marking components is based on the fault-free “X-First” path traversing.

$MarkingComponents()$:
Require: $[Y.X]$ is the index of the current cluster.
Require: $[y,x]$ is the index of the target cluster.
Require: FLC is the list of faultless channels.
Require: FLR is the list of faultless routers.
{Comment0 : Initialization of two lists}
1: $FLC \leftarrow NIL$

2: $FLR \leftarrow NIL$
{Comment1 : Add local channel and local router}
3: $FLC \leftarrow Input\_Channel\_of\_Local\_port\_of\_Command\_Router[Y,X]$
$\bigcup Output\_Channel\_of\_Local\_port\_of\_Command\_Router[y,x]$
$\bigcup Input\_Channel\_of\_Local\_port\_of\_Response\_Router[y,x]$
$\bigcup Output\_Channel\_of\_Local\_port\_of\_Response\_Router[Y,X]$
4: $FLR \leftarrow Command\_Router[Y,X] \bigcup Command\_Router[y,x]$
$\bigcup Response\_Router[y,x] \bigcup Response\_Router[Y,X]$
{Comment2 : Traversing the path on X direction}
5: **if** $X < x$ **then**
6: **for** $i = X$ **to** $x-1$ **do**
7: $FLC \leftarrow FLC \bigcup Output\_Channel\_of\_East\_port\_of\_Command\_Router[Y,i]$
$\bigcup Input\_Channel\_of\_East\_port\_of\_Response\_Router[y,i]$
8: $FLR \leftarrow FLR \bigcup Command\_Router[Y,i+1] \bigcup Response\_Router[y,i]$
9: **end for**
10: **else if** $X > x$ **then**
11: **for** $i = X$ **downto** $x+1$ **do**
12: $FLC \leftarrow FLC \bigcup Output\_Channel\_of\_West\_port\_of\_Command\_Router[Y,i]$
$\bigcup Input\_Channel\_of\_West\_port\_of\_Response\_Router[y,i]$
13: $FLR \leftarrow FLR \bigcup Command\_Router[Y,i-1] \bigcup Response\_Router[y,i]$
14: **end for**
15: **end if**
{Comment3 : Traversing the path on Y direction}
16: **if** $Y < y$ **then**
17: **for** $j = Y$ **to** $y-1$ **do**
18: $FLC \leftarrow FLC \bigcup Output\_Channel\_of\_South\_port\_of\_Command\_Router[j,x]$
$\bigcup Input\_Channel\_of\_South\_port\_of\_Response\_Router[j,X]$
19: $FLR \leftarrow FLR \bigcup Command\_Router[j,x] \bigcup Response\_Router[j+1,X]$
20: **end for**
21: **else if** $Y > y$ **then**
22: **for** $j = Y$ **downto** $y+1$ **do**
23: $FLC \leftarrow FLC \bigcup Output\_Channel\_of\_North\_port\_of\_Command\_Router[j,x]$
$\bigcup Input\_Channel\_of\_North\_port\_of\_Response\_Router[j,X]$
24: $FLR \leftarrow FLR \bigcup Command\_Router[j,x] \bigcup Response\_Router[j-1,X]$
25: **end for**
26: **end if**
27: **return FLC**
28: **return FLR**

The two lists FLR and FLC are distributed in each cluster. Finally, the tree node will collect and merge these informations into two global lists : GFLR and GFLC. Any router or communication channel that is not present in these GFLR and GFLC lists is a “Black Hole”. So, theoretically, the Detection Coverage (DC) of “Black Hole” is 100%, with any number of fault. And this is confirmed by the experimental results.

6.3 Experimental Result

6.3.1 Detection Coverage

In this subsection, we present experimental results of Detection Coverage (DC) evaluation for the "Black Hole" detection. We have simulated two types of fault in a 4×4 clusters MP^{2}SoC. First, single fault injection (single faulty router or one faulty channel) ; Second, multi-faults injections. These fault injections were simulated on a dedicated C simulator.

Single Injection

In a $M \times N$ DSPIN 2D-mesh, there are C cmd&rsp channels, and there are R cmd&rsp routers.

$$C = (M \times (N-1) \times 2 + N \times (M-1) \times 2 + M \times N \times 2) \times 2$$

$$R = M \times N \times 2$$

The Detection Coverage has been evaluated for all the situations where the NoC contains one single fault : single faulty router or single faulty channel defining a total of $(C+R)$ different faulty networks. In our example, with $M = 4, N = 4$, there are 160 channels and 32 routers.

In all cases, the "Black Hole" has been identified and located, resulting in a Detection Coverage of 100% for a single fault.

However, in some cases, some fault-free components are wrongly identified as "Black Holes" due to the channel dependencies, which is explained in the following.

As shown in Figure 6.3.{A}, there are mutual dependencies between :

- The input channel of Local port of command router and the output channel of Local port of response router. Such as ***a*** & ***b***.
- The output channel of Local port of command router and the input channel of Local port of response router. Such as ***c*** & ***d***.

For example, channel ***a*** and channel ***b*** depends on each other. If the channel ***a*** is a "Black Hole", the cluster (0,0) can not be connected into the DCCI tree. Therefore, the channel ***b*** is identified as a "Black Hole". But this result is reasonable. Because, if one channel of a local port is faulty, it breaks the command/response protocol and makes the cluster unusable.

Figure 6.3.{B} shows that there are two other dependencies :

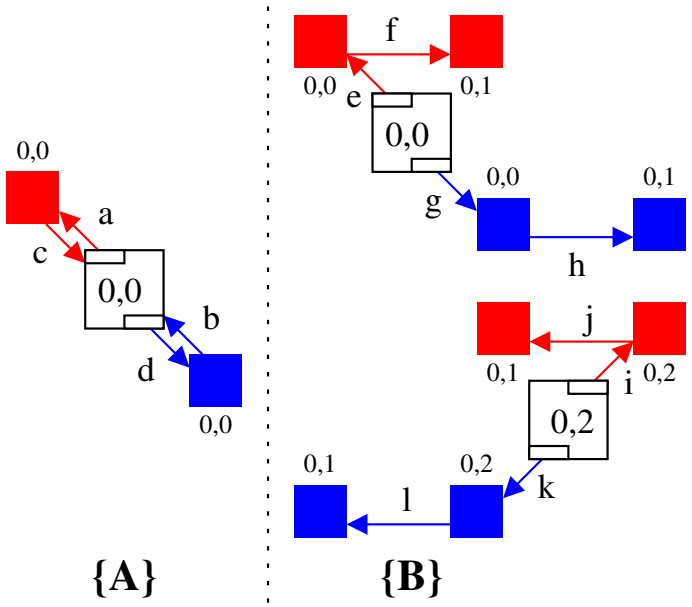


Figure 6.3 – Dependencies exist between some channels.

- For routers located at the West border of the Mesh, the output channel of East port of the router depends on the input channel of Local port of the router. Such as f & e ; h & g.
- For router located at the East border of the Mesh, the output channel of West port of the router depends on the input channel of Local port of the router. Such as j & i ; l & k.

For example, channel f depends on channel e. If the channel e is a "Black Hole", the cluster (0,0) can not be connected into the DCCI tree. According to X-First routing algorithm, all "X-First" paths containing f are sourced by cluster (0,0). That is to say, f can be tested with any "X-First" path, as cluster (0,0) is unusable. Thus, the channel f is identified as a "Black Hole".

Due to the two above mentioned channel dependencies, some "fault-free" channels are identified as faulty. This doesn't induce any destructive problem nor catch out our technique, but only leads to wasting some fault-free channels.

Multi-faults injection

The Detection Coverage of multi-faults injection has been evaluated for all the situations of

- 1 faulty router and 1 faulty channel
- 2 faulty routers
- 2 faulty channels

– 2 faulty routers and 1 faulty channel
– 1 faulty routers and 2 faulty channels
– 2 faulty routers and 2 faulty channels

All these simulations resulted in a 100% "Black Hole" detection coverage.

6.3.2 Application Execution Time Evaluation

The execution time is an important issue for a software procedure that must be executed at each reboot of the system. For this experiment we simulated the complete procedure on a 4×4 2D-mesh MP^{2}SoC architecture containing 16 processors, modeled with the cycle-accurate SoClib virtual prototyping platform [SoClib, 2011]. This architecture allows the injection of different faults to emulate SAF within channels and routers. For one single fault injected, the total localisation time is 7.1×10^6 cycles (without hardware test process) :

– Time for (DCCI) tree construction : 1.9×10^6 cycles
– Time for test task distribution : 1.2×10^6 cycles
– Time for test execution : 3.5×10^6 cycles
– Time for test result centralization : 0.5×10^6 cycles

As this procedure is executed using the system clock, it lasts 0.014 seconds using a clock 500Mhz, which is fully acceptable.

6.3.3 Application Code Size

For a MIPS32 processor, the application code is split into : DCCI : 5 Kbytes per cluster ; Test and localization procedure : 2.5 Kbytes per cluster. Embedding this application in a MP^{2}SoC is thus affordable.

6.4 Conclusion

In this chapter, we detailed a software-based diagnosis/localization of faulty/fault-free components. This process is mandatory to implement the ODDR ("On the field" Detection, De-activation & Reconfiguration) mechanism supporting fault-tolerance in the context of permanent failures.

The "Black Hole" detection relies on a DCCI (Distributed Cooperative Configuration Infrastructure) that dynamically builds a software-based communication tree, covering all the nodes that have successfully passed the local BIST. The tree root is the configuration master, the tree itself is the configuration bus.

The proposed “Black Hole” detection software has been evaluated in a 16 nodes MP^{2}SoC architecture (4×4 mesh) modeled as a SystemC virtual prototype, in the framework of the cycle accurate SoClib environment. It reaches a Detection Coverage of 100% for one or multiple injected faults.

It should be noted that, the method proposed in this chapter, can be used in any shared memory multi-core architecture with a 2D-Mesh NoC.

Chapter 7

Routing Algorithm

Contents

In this chapter, we detail a deterministic, lightweight, deadlock-free and reconfigurable routing algorithm for a 2D-Mesh NoC. This algorithm has been implemented in the DSPIN micro-network, and evaluated from the point of view of performance (penalty on the network saturation threshold), and cost (extra silicon area occupied by the reconfigurable version of the router). In addition, the reconfigurable routing function is proven to be deadlock-free for any single-faulty-router topology.

7.1 The fault model of modification of topology

Once the faulty/de-activated components have been located, the modification of topology is identified. The modification of topology involves five types of fault :

- **Single** faulty channel topology.
- **Multiple** faulty channels topology.
- **Single** faulty router topology.
- **Multiple** faulty routers topology.
- **Mixed** Single or Multiple faulty channel and router topology.

In this chapter, we make the following simplification :

> If an input channel of a router is identified as a “Black Hole”, the complete router is considered as faulty.

Then, the five fault models can be reduced to two types : **Single**-faulty-router topology and **Multiple**-faulty-routers topology. We define now a low cost, reconfigurable routing algorithm that can handle any **Single**-faulty-router topology.

7.2 The reconfigurable routing algorithm for *Single*-faulty-router topology

According to the default routing algorithm of DSPIN : X-First, the packets are firstly routed on the X direction and then on the Y direction. One important feature of the X-First routing algorithm is : the packet path from a node (y,x) to the node (y',x') is a **Unique Path** :

$$L = \{Router_0(y,x), ..., Router_{|x'-x|}(y,x'), ..., Router_{|y'-y|+|x'-x|}(y',x')\}$$

Once an unique path has been broken by a faulty router, the routing algorithm reconfiguration mechanism must recover the broken unique path.

The main idea of the deterministic, reconfigurable routing algorithm is to route the packets through a cycle-free contour surrounding a faulty router, aiming to replace all broken Unique Paths.

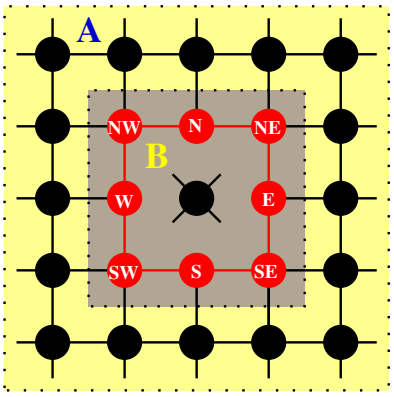


Figure 7.1 – A generic faulty router’s neighbors and the natural contour.

Definition 1 : Neighbors. In a 2D-Mesh, a node (Y,X) has 4 direct neighboring nodes (N,S,W,E) and 4 indirect neighboring nodes (NE,NW,SE,SW). We call those 8 nodes the neighbors, as shown in Figure 7.1.

Definition 2 : Natural Contour. The neighbors of a faulty router (Y,X) define a natural contour as shown in Figure 7.1. It separates the network into two parts : normal part A and defective part B. A single faulty router has $N \times M$ possible locations in a $N \times M$ 2D-Mesh. Thus, a natural contour has 9 possible shapes corresponding to 9 different locations : at each corner, at each side and at other positions, as shown in Figure 7.2.

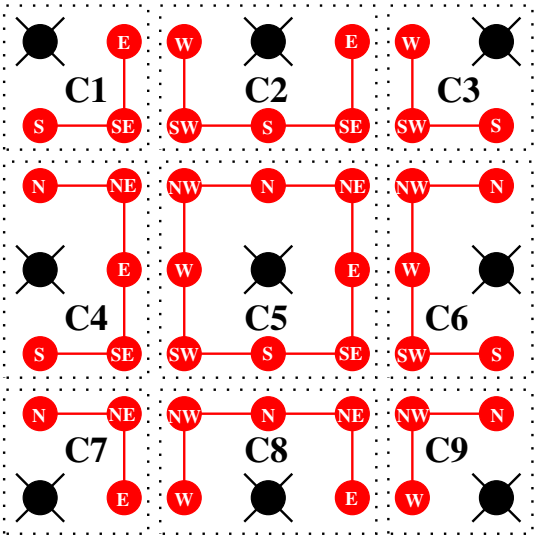


Figure 7.2 – 9 natural contours.

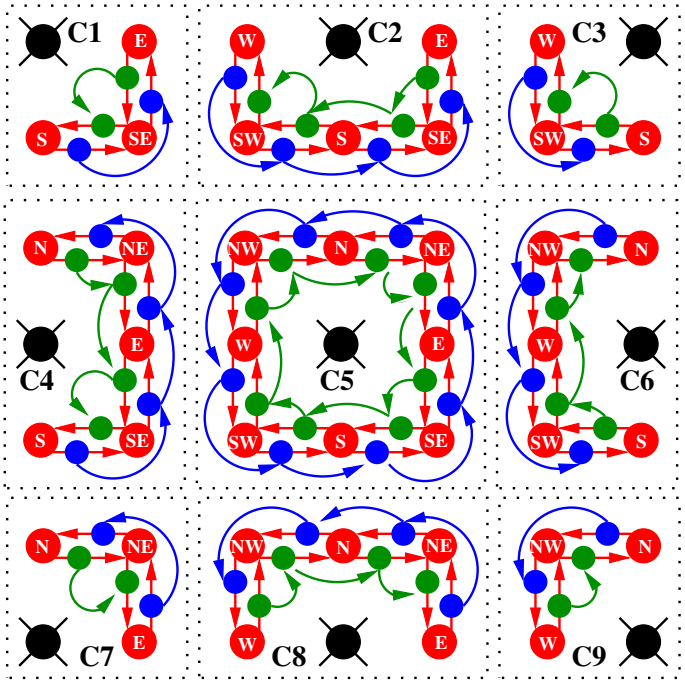


Figure 7.3 – The CDGs of 9 natural contours. 2 cycles are found in the C5's CDG, so C5 can introduce deadlock.

Definition 3 : CDG of Natural Contour. As presented in chapter 2 section 2.2.3 (a node

means a channel, a directed edge from a channel to another, means a possible path defined by the routing algorithm), the Channel Dependency Graphs (CDG) can be used to prove that the 8 natural contours (C1,...,C4,C6,..,C9) are deadlock-free. And the natural contour C5 is NOT deadlock-free, as there is 2 cycles in C5's CDG, as shown in Figure 7.3.

In order to break the 2 cycles in C5's CDG, two NE turns are prohibited as shown in Figure 7.4. As a result, we defined 9 cycle-free contours, corresponding to the 9 possible locations for a faulty router.

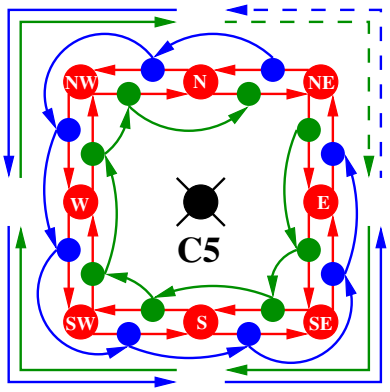


Figure 7.4 – The two turns prohibited (dotted line) in C5's NE can break the 2 cycles.

Definition 4 : New Path. For C5, the 12 broken paths *L*i, listed in Table 7.1, described in Figure 7.5, are replaced by the 12 new path *NewL*i, listed in Table 7.1.

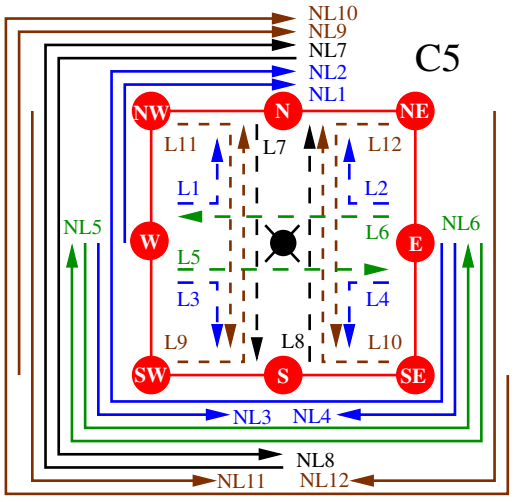


Figure 7.5 – the 12 *L*i (dotted line) broken by a hole are replaced by the 12 *NewL*i (solid lines).

L_1	$\{R_W, R_x, R_N\}$
$NewL_1$	$\{R_W, R_{NW}, R_N\}$
L_2	$\{R_E, R_x, R_N\}$
$NewL_2$	$\{R_E, R_{SE}, R_S, R_{SW}, R_W, R_{NW}, R_N\}$
L_3	$\{R_W, R_x, R_S\}$
$NewL_3$	$\{R_W, R_{SW}, R_S\}$
L_4	$\{R_E, R_x, R_S\}$
$NewL_4$	$\{R_E, R_{SE}, R_S\}$
L_5	$\{R_W, R_x, R_E\}$
$NewL_5$	$\{R_W, R_{SW}, R_S, R_{SE}, R_E\}$
L_6	$\{R_E, R_x, R_W\}$
$NewL_6$	$\{R_E, R_{SE}, R_S, R_{SW}, R_W\}$
L_7	$\{R_N, R_x, R_S\}$
$NewL_7$	$\{R_N, R_{NW}, R_W, R_{SW}, R_S\}$
L_8	$\{R_S, R_x, R_N\}$
$NewL_8$	$\{R_S, R_{SW}, R_W, R_{NW}, R_N\}$
L_9	$\{R_{SW}, R_S, R_x, R_N\}$
$NewL_9$	$\{R_{SW}, R_W, R_{NW}, R_N\}$
L_{10}	$\{R_{SE}, R_S, R_x, R_N\}$
$NewL_{10}$	$\{R_{SE}, R_S, R_{SW}, R_W, R_{NW}, R_N\}$
L_{11}	$\{R_{NW}, R_N, R_x, R_S\}$
$NewL_{11}$	$\{R_{NW}, R_W, R_{SW}, R_S\}$
L_{12}	$\{R_{NE}, R_N, R_x, R_S\}$
$NewL_{12}$	$\{R_{NE}, R_E, R_{SE}, R_S\}$

Table 7.1 – 12 *L*i and 12 *NewL*i.

As shown in Figure 7.6, all broken paths are restored for the 8 other cycle-free contours.

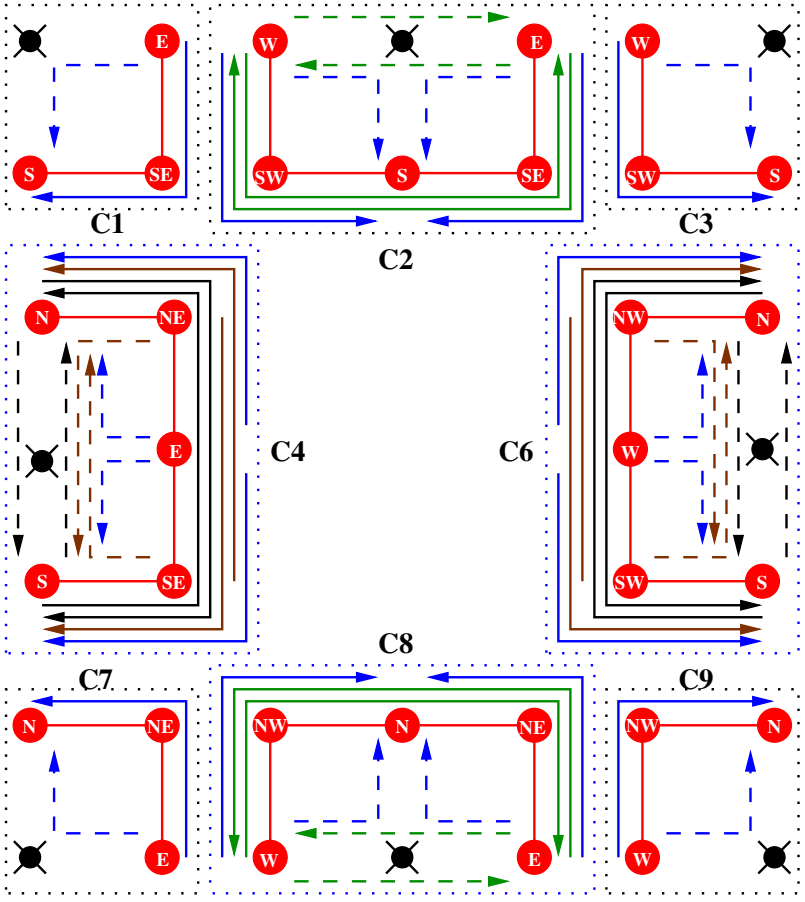


Figure 7.6 – The broken *L*i and *NewL*i in other cycle-free contours.

7.2.1 Routing Function Definition

A given router R can be in 9 different situations : if none of the 8 neighboring routers is faulty, R is configured as NORMAL, it implements the classical X-First routing function. If one of the neighbors is faulty, R is part of a cycle-free contour, and must be accordingly configured (N_OF_x [1], S_OF_x, E_OF_x, W_OF_x, NE_OF_x, NW_OF_x, SE_OF_x, SW_OF_x), where x is the index of the contour (from C1 to C9). In theory, there are 41 different routing functions as shown in Table 7.2.

In fact, some routing functions are identical, there are thus 15 different routing functions presented in Table 7.3.

All 15 routing functions are merged into one unique routing function that is presented in Listing 7.1. This function will be (Logic-block-based) hardware implemented and dis-

1. N_OF_x means that the current router plays the role of ***N*** in the faulty router's contour x

Configuration	Contour	Routing Function
N_OF_x	C4, C5, C6, C7, C8, C9	NC4, NC5, NC6, NC7, NC8, NC9
S_OF_x	C1, C2, C3, C4, C5, C6	SC1, SC2, SC3, SC4, SC5, SC6
E_OF_x	C1, C2, C4, C5, C7, C8	EC1, EC2, EC4, EC5, EC7, EC8
W_OF_x	C2, C3, C5, C6, C8, C9	WC2, WC3, WC5, WC6, WC8, WC9
NE_OF_x	C4, C5, C7, C8	NEC4, NEC5, NEC7, NEC8
NW_OF_x	C5, C6, C8, C9	NWC5, NWC6, NWC8, NWC9
SE_OF_x	C1, C2, C4, C5	SEC1, SEC2, SEC4, SEC5
SW_OF_x	C2, C3, C5, C6	SWC2, SWC3, SWC5, SWC6
Normal		X-First

Table 7.2 – The configuration values and the routing functions.

Configuration	Contour	Routing Function	Difference
N_OF_x	C4	NS1	Packet from N to S : N, NE, E, SE, S
	C5, C6	NS2	Packet N to S : N, NW, W, SW, S
	C7, C8, C9	X-First	No packet from N to S
S_OF_x	C4	SN1	Packet from S to N : S, SE, E, NE, N
	C5, C6	SN2	Packet from S to N : S, SW, W, NW, N
	C1, C2, C3	X-First	No packet from S to N
E_OF_x	C2, C5	EW1	Packet from E to W : E, SE, S, SW, W
	C8	EW2	Packet from E to W : E, NE, N, NW, W
	C1, C4, C7	Y-First	No packet from E to W
W_OF_x	C2, C5	WE1	Packet from W to E : W, SW, S, SE, E
	C8	WE2	Packet from W to E : W, NW, N, WE, E
	C3, C6, C9	Y-First	No packet from W to E
NE_OF_x	C4, C5	NES	Packet from NE to S : NE, E, SE, S
	C7, C8	X-First	No packet from NE to S
NW_OF_x	C5, C6	NWS	Packet from NW to S : NW, W, SW, S
	C8, C9	X-First	No packet from NW to S
SE_OF_x	C4	SEN1	Packet from SE to N : SE, E, NE, N
	C5	SEN2	Packet from SE to N : SE, S, SW, W, NW, N
	C1, C2	X-First	No packet from SE to N
SW_OF_x	C5, C6	SWN	Packet from SW to N : SW, W, NW, N
	C2, C3	X-First	No packet from SW to N

Table 7.3 – The configuration value and the local routing function.

tributed in each router. Its form is :

$$Output = Func(\{Y\_Destination, X\_Destination\}, \{Y\_Local, X\_Local\}, REGISTER)$$

The parameter :

- {Y_Destination, X_Destination} is the coordinates of the packet destination.
- {Y_Local, X_Local} is the coordinates of current router.
- REGISTER is the configuration register value.

In this function, we only need 9 reconfiguration values

```
if (X_Destination > X_Local){                                                    1
    if (REGISTER == NE_OF_x || REGISTER == E_OF_x  || REGISTER == SE_OF_x ||      2
        REGISTER == S_OF_x  || REGISTER == NORMAL)                               3
        OUT = EAST;                                                              4
```

```
    else if (REGISTER == N_OF_x){
        if (Y_Local == 1 || X_Local == 0 || Y_Destination >= Y_Local ||
            X_Destination > X_Local + 1)
            OUT = EAST;
        else
            OUT = WEST;
    }
    else if (REGISTER == NW_OF_x){
        if (Y_Local == 1 || Y_Destination >= Y_Local ||
            X_Destination > X_Local + 2)
            OUT = EAST;
        else
            OUT = SOUTH;
    }
    else if (REGISTER == W_OF_x){
        if (Y_Local == 0 || Y_Destination > Y_Local)
            OUT = NORTH;
        else
            OUT = SOUTH;
    }
    else{
        if (Y_Destination <= Y_Local || X_Destination > X_Local + 1)
            OUT = EAST;
        else
            OUT = NORTH;
    }
}
else if (X_Destination < X_Local){
    if (REGISTER == N_OF_x  || REGISTER == NW_OF_x || REGISTER == W_OF_x  ||
        REGISTER == SW_OF_x || REGISTER == S_OF_x  || REGISTER == NORMAL)
        OUT = WEST;
    else if (REGISTER == NE_OF_x){
        if (X_Destination < X_Local - 1 || Y_Destination >= Y_Local)
             OUT = WEST;
        else
             OUT = SOUTH;
    }
    else if (REGISTER == SE_OF_x){
        if (X_Local == 1 && Y_Destination > Y_Local + 1)
            OUT = NORTH;
        else
            OUT = WEST;
    }
    else{
        if (Y_Local == 0 || ( X_Local == 1 && Y_Destination > Y_Local))
             OUT = NORTH;
        else
             OUT = SOUTH;
   }
}
else if (Y_Destination > Y_Local){
    if (REGISTER != S_OF_x)
        OUT = NORTH;
    else if (X_Local != 0)
        OUT = WEST;
    else
        OUT = EAST;
}
else if(Y_Destination < Y_Local){
    if (REGISTER != N_OF_x)
        OUT = SOUTH;
    else if (X_Local != 0)
        OUT = WEST;
    else
        OUT = EAST;
}
else
    OUT = LOCAL;
```

Listing 7.1 – The routing function of cycle-free contour for single-faulty-router-topology

7.2.2 Deadlock-free Proof

In order to prove that this reconfigurable routing function is deadlock-free, we used the formal proof tool ODI [Taktak et al., 2008], which is developed at LIP6. This tool is dedicated to deadlock analysis in packet switching networks. It is based on the analysis of "Strongly Connected Components" (SCC) of the Extended Dependency Graph defined by the micro-network topology on one hand, and by the routing algorithm on the other hand. That is to say, this tool can prove there is a cycle or not in CDG for the reconfigurable routing algorithm in a topology. It can thus build a sufficient condition of deadlock-free property.

With this tools, we have proved the above mentioned routing function to be deadlock-free in any single-faulty-router topology, for a 5×5 and 10×10 2D-Mesh. Thus, this reconfigurable routing algorithm can fit the minimum fault-tolerant requirement (described in chapter 1 section 1.5.2).

7.3 Experimental Results

7.3.1 Performance (penalty on the network saturation threshold) :

With the cycle-accurate SoClib virtual prototyping platform [SoClib, 2011], we simulated a 2D-Mesh containing 5×5 clusters. Each cluster contains one traffic generator and one target. For each initiator, the offered load (defined as the ratio between the number of injected flits and the total number of cycles) can be precisely adjusted. The traffic has a uniform random distribution (each initiator sends packets to all targets). The packet length is 8. The average network latency is measured as the average number of cycles for a round trip from an initiator to a target, and back to the same initiator.

If we plot the average latency versus the offered load, the saturation threshold is the maximal accepted load where the latency increases to infinity. We simulated all single-faulty-router topologies, and the Figure 7.7 presents the results for 5 cases : no hole [2], hole in (0,0), hole in (0,2), hole in (1,1), hole in (2,2).

According to this simulation, we find out that :

- When the load is not too high ($<10\%$), the impact on the average latency is negligible.
- When the hole is located at the corner of the mesh, the saturation threshold isn't modified.

2. Hole means a faulty router.

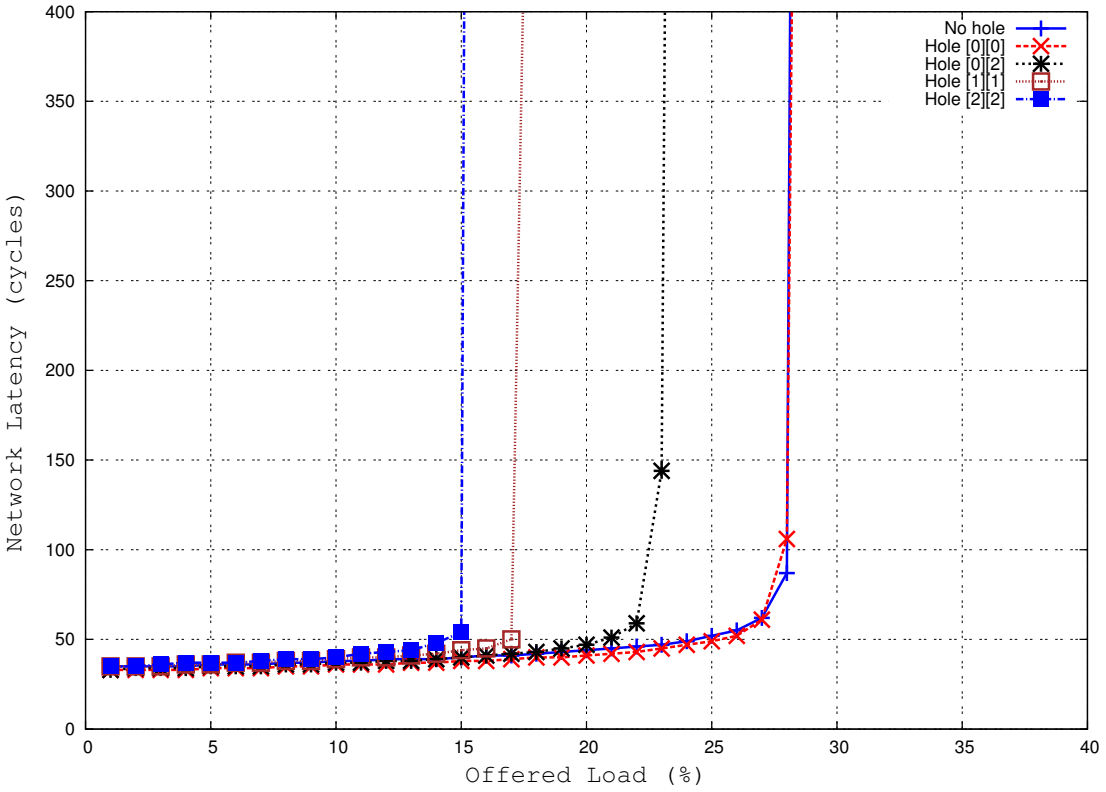


Figure 7.7 – Some saturation thresholds in 5×5 2D-Mesh.

- When the hole is located at the center of the mesh, the saturation threshold is strongly modified (as a 2D-Mesh topology has central symmetry, the center is a hotspot in the simulation of uniform random distribution. When the hotspot is faulty, the saturation threshold is strongly modified).

7.3.2 Cost (extra silicon area) :

The reconfigurable routing function described in Listing 7.1 has been synthesized with Synopsys synthesis environment using the SXLIB standard cell library [Alliance, 2011], to evaluate the cost of silicon area.

Component	NAND
Original DSPIN Router	13500.75
X-First Routing Algorithm	71.25
Reconfigurable Routing Algorithm	240
Overhead (%)	168.75 (1.25%)

Due to reconfigurable routing algorithm, the router (whole NoC) footprint is increased by only 1.25%. This is a very low cost. It should be noted that, this 1.25% is already

contained in all overhead presented in chapter 5 section 5.10.

7.4 conclusion

We propose an ultra-low-cost reconfigurable routing algorithm. It requires only a 4bits configuration register per router. It has been physically implemented in the DSPIN micro-network. The silicon area penalty is only 1.25% of the router footprint.

The impact on the latency and saturation threshold has been evaluated. The reconfigurable routing algorithm is fully scalable. It has been demonstrated in the DSPIN micro-network, but can be used in any 2D-Mesh Network-on-Chip. This routing algorithm is proven to be deadlock-free with any single-faulty-router topology.

In any 2D-Mesh NoC-based, MP^{2}SoC architecture, this reconfigurable routing algorithm capability is mandatory to implement the “on the field” fault-tolerant approach.

Conclusion

In this thesis, we presented a complete ODDR ("On the field" Detection, De-activation and Reconfiguration) mechanism, used to establish permanent fault-tolerance for a 2D-Mesh NoC in shared memory MP^{2}SoC architecture. This mechanism contains three stages :

1 Initialization stage / NoC Test

2 Pre-configuration stage / Configuration Infrastructure

3 Configuration stage / Routing Algorithm

With these three stages, a damaged 2D-Mesh NoC can structurally self-test, partially self-disable, globally self-reconfigure and functionally self-recover, after a simple system reboot or at chip power-on. Moreover, these stages can be used in any 2D-Mesh NoC based, shared memory, multi-cores architecture.

Initialization Stage

A fully distributed off-line BIST dedicated to 2D-Mesh NoC in a GALS context, is proposed. This BIST is implemented as an initialization procedure, and executed at each system reboot or chip power-on. With the help of this procedure, the fault-free components are activated, while the faulty components are detected, de-activated and configured to behave as "Black Hole". Therefore, this procedure avoids the failure propagation and cleans the malfunctions of the overall Network-on-Chip through a simple reboot of the chip, and it makes the NoC ready for pre-reconfiguration. In addition, this test procedure provides a global SAF coverage about 91%, for an acceptable silicon area overhead (NoC footprint increased by 50.50%).

The initialization procedure can be performed not only "on the field", but also at fabrication time. It can thus help to improve the yield by avoiding to throw the whole chip when one single component is faulty.

It should be noted that, the initialization procedure can be reused, as a test infrastructure in various 2D-Mesh Network-on-Chip.

Pre-configuration Stage

With the help of a distributed & embedded Configuration Firmware (CF), the DCCI tree (Distributed Cooperative Configuration Infrastructure developed by D.Refauvelet) is constructed on top of an initialized NoC. With this tree, all usable clusters can be covered and connected. Each tree node is an embedded processor core of a cluster. The tree root is the global configuration master. The tree itself is the global configuration bus.

Through the tree, at first, the root makes each node to execute the software application called "Black Hole" detection. Then, the root centralizes each local result to globally identify the "Black Holes" and the modification of topology. Finally, the root can configure the global routing function of NoC in the configuration stage.

The proposed "Black Hole" detection software has been evaluated in a 16 nodes MP^2SoC architecture (4×4 mesh) modeled as a SystemC virtual prototype, in the framework of the cycle accurate SoClib environment. It reaches a Detection Coverage of 100% for all single or multiple injected faults. Moreover, the software execution time is evaluated, the value is fully acceptable (0.014 seconds at 500Mhz). And the application code size is affordable (7.5 Kbytes).

It should be noted that, this configuration infrastructure and the detection software, can be used in any shared memory multi-core architecture with a 2D-Mesh NoC.

Configuration Stage

An ultra-low-cost reconfigurable routing algorithm is proposed to handle the modification of topology. With this routing algorithm, the network itself is split into two regions : normal region and faulty region. The routers of normal region are not reconfigured, but the border routers of the faulty region are configured to create a cycle-free contour, aiming to bypass the faulty/disable routers.

This routing algorithm requires only a 4bits configuration register per router. And it induces only 1.25% silicon area overhead on the router footprint. Furthermore, the reconfigurable routing function is proven to be deadlock-free with any ***Single***-faulty-router topology. And it can be extended to handle the ***Multi***-faulty-router topology. In addition, this reconfigurable routing algorithm can be used in any 2D-Mesh Network-on-Chip.

In conclusion, with these three stages, we have successfully solved three problems identified in chapter 2, concerning the state of the art of the ODDR mechanism.

1. **NoC Test** : In order to support "on the field" NoC reconfiguration strategy, an off-line, "test for de-activation" BIST mechanism is necessary. As far as we know, there

is not yet such BIST, and we must solve this important problem. This problem has been solved in the initialization stage.

2. **Configuration Infrastructure** : The DCCI will be used in this thesis as the configuration infrastructure. However, DCCI doesn't allow the diagnosis/localization of the faulty/fault-free components. Thus, the modification of topology can not be identified. So, this problem must be solved. This problem has been solved in the pre-configuration stage.

3. **Routing Algorithm** : As far as we analyze, the published fault-tolerant, deterministic reconfigurable routing algorithms either are expensive, or can not handle some single-faulty-router topologies. Moreover, they haven't been formally proven to be deadlock-free. Therefore, we have to define a new fault-tolerant routing algorithm. This problem has been solved in the configuration stage.

Answers to the Questions asked in chapter 1 (Problem Definition)

The related open questions listed in chapter 1 (Problem Definition), on three topics : NoC Test, Configuration Infrastructure and Routing Algorithm, are answered in this section.

NoC Test

Question 1 : How to handle the interaction between the router test and the channel test ?

Answer : As shown in chapter 5 section 5.2 Figure 5.3, we propose an algorithm to implement BIST test process, distributed in each router, to achieve two levels of parallelism. In the first level of parallelism, the test of all routers start at each system reboot or chip power-on. And then, when the router test is OK, the tests of all associated channels start at the same time. Thus, the router test and the channel test can be coordinated without mutual interference.

Question 2 : How to generate efficient test patterns for channel test and router switch module (crossbar) test ?

Answer : As shown in chapter 5 section 5.7 and section 5.8, we propose and detail an ad-hoc test pattern generation, for channel test and crossbar test. The main idea is to determine the form and the value of test patterns by the analysis of component's function and structure, and to evaluate the fault coverage by fault simulation.

Question 3 : How to realize the de-activation approach within the BIST strategy ? And How to configure the fault component to behave as a "Black Hole" ?

Answer : The de-activation and "Black Hole" configuration of the faulty components are realized through controlling the additional multiplexer using the states of ATC, ATG and ATA. The detail is presented in chapter 5 section 5.4.

Question 4 : Can the proposed communication channel test handle the clock boundaries ? And How ?

Answer : The proposed communication channel test can handle the clock boundaries. As shown in chapter 5 section 5.8, the bi-synchronous communication channel is tested by a couple of ATG/ATA FSMs. ATG/ATA work at each border of the channel, in two different clock domains. Their cooperation is achieved with the help of some asynchronous handshake signals, like a ping-pong game, as shown in Figure 5.15. This cooperation is verified to be robust against large variations of clock frequencies as presented in section 5.8.5.

Question 5 : How many test patterns are generated for the NoC test ?

Answer : The total number of test patterns are

$$383 = 52_{\text{ For Routing Function Test}} + 225_{\text{ For Crossbar Test}} + 106_{\text{ For Channel Test}}$$

Question 6 : What is the cost of the NoC test ?

Answer : According to the timeout mechanism, the test times is 40000 clock cycles, 0.00008 seconds at 500 MHz. The NoC footprint is increased by 50.50% (that also includes the overhead due to the reconfigurable routing algorithm). This is much smaller than 200%, the overhead due to TMR : Triple Modular Redundancy. This is an affordable cost, since the DSPIN NoC is a very compact design and typically represents less than 3% [Panades et al., 2008] of the silicon area in a typical MP^{2}SoC architecture.

Question 6 : What is the SAF coverage respectively reached by the channel test and the router test ? What is the SAF coverage reached by the whole network (with the BIST circuit) test ?

Answer : The SAF coverage is presented in chapter 5 section 5.10 : 100% for routing function test, 99.33% for crossbar test, 98.57% for channel test and 90.95% for the whole network test.

Configuration Infrastructure

Question 1 : Can the test software 100% diagnose/locate faulty/disabled component ?

Answer : The software application of "Black Hole" detection can 100% locate faulty/disabled component.

Question 2 : Can the test software 100% diagnose/locate the fault-free component ?

Answer : The software application of "Black Hole" detection can not 100% locate fault-free component. Due to the channel dependencies, some fault-free components are identified as "Black Hole". The detail is presented in chapter 6 section 6.3.1. However, this result doesn't induce any destructive problem, but only leads to wasting some fault-free channels.

Question 3 : What is the performance of the test software ?

Answer : The execution time of the software was simulated on a 4×4 2D-mesh MP^{2}SoC architecture containing 16 processors, modeled with the cycle-accurate SoClib virtual prototyping platform [SoClib, 2011]. This architecture contained one single faulty router. The software execution time is 4×10^6 cycles, 0.008 seconds at 500Mhz. Besides, for a MIPS32 processor, the test software application code requires 2.5 Kbytes.

Routing Algorithm

Question 1 : Which fault model of modification of topology can be handled by the proposed routing algorithm ? And can all of single-faulty-router topologies be handled ?

Answer : As presented in chapter 7 section 7.1, all five fault models are simplified to two types : ***Single***-faulty-router topology and ***Multi***-faulty-routers topology. These two fault models can be handled by our proposed reconfigurable routing algorithm. And the proposed reconfigurable routing algorithm can handle any single-faulty-router topologies.

Question 2 : Can we formally prove that the proposed fault-tolerant reconfigurable routing algorithm is deadlock-free and livelock-free for all of single-faulty-router topologies ?

Answer : We proved that the fault-tolerant reconfigurable routing algorithm is deadlock-free with all single-faulty-router topologies using the ODI tool [Taktak et al., 2008]. As the reconfigurable routing algorithm is deterministic, it's thus livelock-free.

Question 3 : What is the impact or penalty on the network transmission performance ?

Answer : As presented in chapter 7 section 7.3.1, the traffic simulation is done on a 2D-Mesh MP^{2}SoC architecture containing 5×5 clusters. We simulated all single-faulty-router topologies, and we find out that :

- When the load is not too high (<10%), the impact on the average latency is negligible.

- When the hole is located at the border of the mesh, the saturation threshold isn't modified.
- When the hole is located at the center of the mesh, the saturation threshold is strongly modified.

Question 4 : What is the cost on the silicon area (comparing with X-First routing algorithm implementation) ?

Answer : The silicon area increases only 1.25% the NoC footprint.

Question 5 : Is it possible to generalize the proposed fault-tolerant reconfigurable routing algorithm to handle other topologies than the 2D-Mesh ?

Answer : We think that this routing algorithm can be extended to 2D-Torus topology and 3D-Mesh topology, since these two topologies are an extension of 2D-Mesh.

Future Work

The presented ODDR ("On the field" Detection, De-activation and Reconfiguration) mechanism, supporting "on the field" permanent fault-tolerance for 2D-Mesh NoC, is the first complete solution in the world as far as we known. This mechanism can be used to improve both the chip manufacture yield, the chip function reliability and the chip lifetime.

The presented mechanism follows the graceful degradation theory, where the faulty channel or faulty router are completely de-activated. However, this de-activation rule is seen to be strict but wasteful. Because, the de-activation of a channel (of local port) or a router, leads to dropping down the whole associated cluster. But, the area of a router only takes few (3%) of cluster. Thus, we can try to define more fine-grained detection and de-activation mechanisms, such as the de-activation of an inner routing path between a couple of input/output port of the router crossbar. As far as, we know, such mechanisms have not been studied until now.

Glossary

ATE	Automated Test Equipment
ATPG	Automatic Test Pattern Generation
BIST	Built-In Self-Test
CAM	Configuration Access Mechanism
CF	Configuration Firmware
CRC	Cyclic-Redundancy Check
DCCI	Distributed Cooperative Configuration Infrastructure
DFT	Design For Test
DSPIN	Distributed Scalable Predictable Interconnect Network
ECC	Error Control Code
GALS	Globally Asynchronous, Locally Synchronous
IP	Intellectual Property
LFSR	Linear Feedback Shift Register
MP^{2}SoC	Massively-Parallel Multi-Processors System-on-Chip
NIC	Network Interface Controller
NMR	N Module Redundancy
NoC	Network-on-Chip
ODDR	“On the field” Detection, De-activation & Reconfiguration
OS	Operating System
PRSG	Pseudo-Random Sequence Generator
SAF	Stuck-At Fault
SCC	Single-chip Cloud Computer
SoC	System-on-Chip
VLSI	Very Large Scale Integration

List of Publications

International Conferences

2008 Zhen Zhang, Alain Greiner and Sami Taktak. **A reconfigurable routing algorithm for a fault-tolerant 2D-Mesh Network-on-Chip**, *In Proceedings of the 45th Design Automation Conference (DAC'08)*, pages 441-446, USA, 2008.

2010 Zhen Zhang, Alain Greiner and Mounir Benabdenbi. **Fully Distributed Initialization Procedure for a 2D-Mesh NoC, Including Off Line BIST and Partial Deactivation of Faulty Components**, *In Proceedings of the 16th IEEE International On-Line Testing Symposium (IOLTS'10)* pages 194-196, Greece, 2010.

2011 Zhen Zhang, Dimitri Refauvelet, Alain Greiner, Mounir Benabdenbi and François Pecheux. **Localization of Damaged Resources in NoC Based Shared-Memory MP2SOC, using a Distributed Cooperative Configuration Infrastructure** *In Proceedings of the 29th IEEE VLSI Test Symposium (VTS'11)*, USA, 2011.

National Conferences

2008 Zhen Zhang, Alain Greiner. **Un Algorithme de Routage Reconfigurable pour la Tolérance aux Fautes dans le micro-réseau DSPIN**, *In Colloque GDR SOC-SIP*, France, 2008

Bibliography

[Alliance, 2011] Alliance (2011). Alliance CAD. `http://www-asim.lip6.fr/recherche/alliance`. 76, 89, 109

[Amory et al., 2005] Amory, A., Brião, E., Cota, E., Lubaszewski, M., and Moraes, F. (2005). A scalable test strategy for network-on-chip routers. In *Proceedings of the 36th IEEE International Test Conference (ITC'05)*, pages 9 pp.–599. 34

[Benini and Micheli, 2002] Benini, L. and Micheli, G. D. (2002). Networks on Chips : A New SoC Paradigm. *Computer*, 35 :70–78. 1

[Bogdan et al., 2007] Bogdan, P., Dumitras, T., and Marculescu, R. (2007). Stochastic Communication : A New Paradigm for Fault-Tolerant Networks-on-Chip. *VLSI Design*, 2007. 23

[Boppana and Chalasani, 1994] Boppana, R. V. and Chalasani, S. (1994). Fault-tolerant routing with non-adaptive wormhole algorithms in mesh networks. In *Proceedings of the 1994 conference on Supercomputing (Supercomputing'94)*, pages 693–702. 37, 40

[Bruck et al., 1993] Bruck, J., Cypher, R., and Ho, C. (1993). Fault-tolerant meshes and hypercubes with minimal numbers of spares. *IEEE Transactions on Computers*, 42(9) :1089–1104. 32

[Chalasani and Boppana, 1995a] Chalasani, S. and Boppana, R. V. (1995a). Fault-tolerant wormhole routing algorithms for mesh networks. *IEEE Transactions on Computers*, 44(7) :848–864. 23

[Chalasani and Boppana, 1995b] Chalasani, S. and Boppana, R. V. (1995b). Fault-Tolerant Wormhole Routing Algorithms for Mesh Networks. *IEEE Transactions on Computers*, 44(7) :848–864. 33

[Chalasani and Boppana, 1996] Chalasani, S. and Boppana, R. V. (1996). Communication in Multicomputers with Nonconvex Faults. *IEEE Transactions on Computers*, 46(5) :616–622. 33

[Chen and Chiu, 1998] Chen, K. and Chiu, G. (1998). Fault-tolerant routing algorithm for meshes without using virtual channels. *Journal of Information Science and Engineering*, 14 :765–783. 23

[Chiu, 2002] Chiu, G.-M. (2002). The odd-even turn model for adaptive routing. *IEEE Transactions on Parallel and Distributed Systems*, 11(7) :729–738. 39

[Concatto et al., 2009] Concatto, C., Almeida, P., Kastensmidt, F., Cota, E., Lubaszewski, M., and Herve, M. (2009). Improving yield of torus nocs through fault-diagnosis-and-repair of interconnect faults. In *Proceedings of the 15th IEEE International On-Line Testing Symposium (IOLTS'09)*, pages 61–66. 34

[Constantinides et al., 2006] Constantinides, K., Plaza, S., Blome, J., Zhang, B., Bertacco, V., Mahlke, S., Austin, T., and Orshansky, M. (2006). Bulletproof : A defect-tolerant CMP switch architecture. In *Proceedings of the 12th International Symposium on High-Performance Computer Architecture (HPCA'06)*, pages 5–16. 30

[Cota et al., 2007] Cota, E., Kastensmidt, F., Cassel, M., Meirelles, P., Amory, A., and Lubaszewski, M. (2007). Redefining and testing interconnect faults in Mesh NoCs. In *Proceedings of the 38th IEEE International Test Conference (ITC'07)*. 34

[Cota et al., 2008] Cota, E., Kastensmidt, F. L., Cassel, M., Hervé, M., Almeida, P., Meirelles, P., Amory, A., and Lubaszewski, M. (2008). A High-Fault-Coverage Approach for the Test of Data, Control and Handshake Interconnects in Mesh Networks-on-Chip. *IEEE Transactions on Computers*, 57(9) :1202–1215. 34

[Cunningham and Avresky, 1995] Cunningham, C. M. and Avresky, D. R. (1995). Fault-tolerant adaptive routing for two-dimensional meshes. In *Proceedings of the 1st IEEE Symposium on High-Performance Computer Architecture (HPCA'95)*. 33

[Dally and Seitz, 1987] Dally, W. J. and Seitz, C. L. (1987). Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, 36(5) :547–553. 7, 37

[Dally and Towles, 2004] Dally, W. J. and Towles, B. (2004). *Principles and practices of interconnection networks*. Morgan Kaufmann. 7, 28

[Duato, 1993] Duato, J. (1993). A new theory of deadlock-free adaptive routing in wormhole networks. *IEEE Transactions on Parallel and Distributed Systems*, 4(12) :1320–1331. 23

[Duato, 1997] Duato, J. (1997). A Theory of Fault-Tolerant Routing in Wormhole Networks. *IEEE Transactions on Parallel and Distributed Systems*, 8(8) :790–802. 33

[Dumitraş et al., 2003] Dumitraş, T., Kerner, S., and Mărculescu, R. (2003). Towards on-chip fault-tolerant communication. In *Proceedings of the 8th Asia and South Pacific Design Automation Conference (ASPDAC'03)*, pages 225–232. 19

[Fick et al., 2009a] Fick, D., DeOrio, A., Chen, G., Bertacco, V., Sylvester, D., and Blaauw, D. (2009a). A highly resilient routing algorithm for fault-tolerant NoCs. In *Proceedings of the 12th Conference on Design, Automation and Test in Europe (DATE'09)*, pages 21–26. 33

[Fick et al., 2009b] Fick, D., DeOrio, A., Hu, J., Bertacco, V., Blaauw, D., and Sylvester, D. (2009b). Vicis : A reliable network for unreliable silicon. In *Proceedings of the 46th Design Automation Conference (DAC'09)*, pages 812–817. 30, 31

[Furber, 2006] Furber, S. (2006). Living with Failure : Lessons from Nature ? In *Proceedings of the 11th IEEE European Test Symposium (ETS'06)*, pages 4–8. 1, 6

[Gizopoulos et al., 2004] Gizopoulos, D., Paschalis, A., and Zorian, Y. (2004). *Embedded processor-based self-test*. Kluwer Academic Pub. 92

[Glass and Ni, 1993] Glass, C. and Ni, L. (1993). Fault-tolerant wormhole routing in meshes. In *Proceedings of the 23th International Symposium on Fault-Tolerant Computing (FTCS'93)*, pages 240–249. IEEE. 33

[Glass and Ni, 1994] Glass, C. J. and Ni, L. M. (1994). The turn model for adaptive routing. *Journal of the ACM (JACM)*, 41 :874–902. 23, 39

[Grecu et al., 2005] Grecu, C., Pande, P., Wang, B., Ivanov, A., and Saleh, R. (2005). Methodologies and algorithms for testing switch-based NoC interconnects. In *Proceedings of the 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'05)*, pages 238–246. 18, 34

[Herve et al., 2009] Herve, M., Cota, E., Kastensmidt, F. L., and Lubaszewski, M. (2009). Diagnosis of interconnect shorts in mesh NoCs. In *Proceedings of the 3rd ACM/IEEE International Symposium on Networks-on-Chip (NOCS'09)*, pages 256–265. 34

[Holsmark and Kumar, 2007] Holsmark, R. and Kumar, S. (2007). Corrections to Chen and Chiu's fault tolerant routing algorithm for mesh networks. *Journal of Information Science and Engineering*, 23(6) :1649–1662. 23

[Hosseinabady et al., 2006] Hosseinabady, M., Banaiyan, A., Bojnordi, M. N., and Navabi, Z. (2006). A concurrent testing method for NoC switches. In *Proceedings of the 9th Conference on Design, Automation and Test in Europe (DATE'06)*, pages 1171–1176. 34

[Hosseinabady et al., 2007] Hosseinabady, M., Dalirsani, A., and Navabi, Z. (2007). Using the inter-and intra-switch regularity in NoC switch testing. In *Proceedings of the 9th Conference on Design, Automation and Test in Europe (DATE'07)*, pages 361–366. 34

[ITRS, 2009] ITRS (2009). International Technology Roadmap for Semiconductors. `http://www.itrs.net/Links/2009ITRS/Home2009.htm`. 9

[Jarwala and Yau, 1989] Jarwala, N. and Yau, C. (1989). A new framework for analyzing test generation and diagnosis algorithms for wiring interconnects. In *Proceedings of 1989 International Test Conference. Meeting the Tests of Time.*, pages 63–70. 28

[Kariniemi and Nurmi, 2006] Kariniemi, H. and Nurmi, J. (2006). Fault-tolerant 2-D Mesh Network-On-Chip for MultiProcessor Systems-on-Chip. In *Proceedings of the 2006 IEEE Design and Diagnostics of Electronic Circuits and Systems (DDECS '06)*, pages 184–189. 34

[Kohler and Radetzki, 2009] Kohler, A. and Radetzki, M. (2009). Fault-tolerant architecture and deflection routing for degradable NoC switches. In *Proceedings of the 3rd ACM/IEEE International Symposium on Networks-on-Chip (NOCS'09)*, pages 22–31. 30

[Koibuchi et al., 2008] Koibuchi, M., Matsutani, H., Amano, H., and Pinkston, T. M. (2008). A lightweight fault-tolerant mechanism for Network-on-Chip. In *Proceedings of the 2nd ACM/IEEE International Symposium on Networks-on-Chip*, pages 13–22. 32

[Kolonis et al., 2009] Kolonis, E., Nicolaidis, M., Gizopoulos, D., Psarakis, M., Collet, J., and Zajac, P. (2009). Enhanced self-configurability and yield in multicore grids. In *Proceedings of the 15th IEEE International On-Line Testing Symposium (IOLTS'09)*, pages 75–80. 35

[Lien and Breuer, 1991] Lien, J.-C. and Breuer, M. A. (1991). Maximal Diagnosis for Wiring Networks. In *Proceedings of the IEEE International Test Conference on Test : Faster, Better, Sooner*, pages 96–105. 28

[Lin et al., 2009] Lin, S., Shen, W., Hsu, C., Chao, C., and Wu, A. (2009). Fault-tolerant router with built-in self-test/self-diagnosis and fault-isolation circuits for 2D-mesh based chip multiprocessor systems. In *Proceedings of 2009 International Symposium on VLSI Design, Automation and Test (VLSI-DAT'09)*, pages 72–75. 31

[Mejia et al., 2006] Mejia, A., Flich, J., Duato, J., Reinemo, S., and Skeie, T. (2006). Segment-based routing : an efficient fault-tolerant routing algorithm for meshes and tori. In *Proceedings of the 20th IEEE International Parallel and Distributed Processing Symposium (IPDPS'06)*, page 84. 33, 37, 40

[Miro Panades and Greiner, 2007] Miro Panades, I. and Greiner, A. (2007). Bi-synchronous FIFO for synchronous circuit communication well suited for network-on-chip in gals architectures. In *Proceedings of the 1st International Symposium on Networks-on-Chip (NOCS'07)*, pages 83–94. 45

[Ni and McKinley, 1993] Ni, L. and McKinley, P. (1993). A Survey of Wormhole Routing Techniques in Direct Networks. *Computer*, 26(2) :62–76. 46

[Nunez-Yanez et al., 2008] Nunez-Yanez, J., Edwards, D., and Coppola, A. (2008). Adaptive routing strategies for fault-tolerant on-chip networks in dynamically reconfigurable systems. *Computers & Digital Techniques, IET*, 2(3) :184–198. 33

[Panades, 2008] Panades, I. (2008). *Conception et implantation d'un micro-réseau sur puce avec garante de service*. PhD thesis, Université Pierre et Marie Curie. 81

[Panades et al., 2008] Panades, I., Clermidy, F., Vivet, P., and Greiner, A. (2008). Physical Implementation of the DSPIN Network-on-Chip in the FAUST Architecture. In *Proceedings of the 3nd ACM/IEEE International Symposium on Networks-on-Chip (NOCS'08)*, pages 139–148. 89, 114

[Panades et al., 2006] Panades, I. M., Greiner, A., and Sheibanyrad, A. (2006). A Low Cost Network-on-Chip with Guaranteed Service Well Suited to the GALS Approach. In *Proceedings of the 1st International Conference on Nano-Networks (NanoNet'06)*, pages 1–5. 1

[Pande et al., 2005] Pande, P. P., Grecu, C., Ivanov, A., Saleh, R., and Micheli, G. D. (2005). Design, synthesis, and test of networks on chips. *IEEE Design and Test of Computers*, 22(5) :404–413. 34

[Park, 1996] Park, S. (1996). A new complete diagnosis patterns for wiring interconnects. In *Proceedings of the 33rd Design Automation Conference (DAC'96)*, pages 203–208. 28

[Park et al., 2000] Park, S., Youn, J.-H., and Bose, B. (2000). Fault-tolerant wormhole routing algorithms in meshes in the presence of concave faults. In *Proceedings of the 14th International Symposium on Parallel and Distributed Processing (IPDPS'00)*, pages 633–638. 23

[Pasca et al., 2010] Pasca, V., Anghel, L., Rusu, C., and Benabdenbi, M. (2010). Configurable serial fault-tolerant link for communication in 3D integrated systems. In *Proceedings of the 16th IEEE International On-Line Testing Symposium (IOLTS'10)*, pages 115–120. 28

[Paul, 1966] Paul, R. J. (1966). Diagnosis of automata failures : a calculus and a method. *IBM J. Res. Dev.*, 10 :278–291. 14

[Petersén and Öberg, 2007] Petersén, K. and Öberg, J. (2007). Toward a scalable test methodology for 2D-mesh network-on-chips. In *Proceedings of the 9th Conference on Design, Automation and Test in Europe (DATE'07)*, pages 367–372. 34

[Pirretti et al., 2004] Pirretti, M., Link, G., Brooks, R., Vijaykrishnan, N., Kandemir, M., and Irwin, M. (2004). Fault tolerant algorithms for network-on-chip interconnect. In *Proceedings of the IEEE Computer society Annual Symposium on VLSI*, pages 46–51. 23

[Raik et al., 2006] Raik, J., Govind, V., and Ubar, R. (2006). An External Test Approach for Network-on-a-Chip Switches. In *Proceedings of the 15th Asian Test Symposium (ATS'06)*, pages 437–442. 34

[Raik et al., 2007] Raik, J., Ubar, R., and Govind, V. (2007). Test Configurations for Diagnosing Faulty Links in NoC Switches. In *Proceedings of the 12th IEEE European Test Symposium(ETS'07)*, pages 29–34. 18, 34

[Rodrigo et al., 2010] Rodrigo, S., Flich, J., Roca, A., Medardoni, S., Bertozzi, D., Camacho, J., Silla, F., and Duato, J. (2010). Addressing Manufacturing Challenges with Cost-Efficient Fault Tolerant Routing. In *Proceedings of the 4th ACM/IEEE International Symposium on Networks-on-Chip (NOCS'10)*, pages 25–32. 33

[SCC, 2009] SCC (2009). Intel Single-chip Cloud Computer. `http://techresearch.intel.com/articles/Tera-Scale/1826.htm`. 6

[Shekhar, 2007] Shekhar, B. (2007). Thousand core chips : a technology perspective. In *Proceedings of the 44th Design Automation Conference (DAC'07)*, pages 746–749. 6

[Shi and Fuchs, 1995] Shi, W. and Fuchs, W. K. (1995). Optimal interconnect diagnosis of wiring networks. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 3(3) :430–436. 28

[SoClib, 2011] SoClib (2011). SoClib an open platform for virtual prototyping of multiprocessors system on chip. `http://www.soclib.fr`. 99, 108, 115

[Song et al., 2009] Song, W., Edwards, D., Nunez-Yanez, J. L., and Dasgupta, S. (2009). Adaptive stochastic routing in fault-tolerant on-chip networks. In *Proceedings of the 3rd ACM/IEEE International Symposium on Networks-on-Chip (NOCS'09)*, pages 32–37. 23

[Stewart and Tragoudas, 2006] Stewart, K. and Tragoudas, S. (2006). Interconnect Testing for Networks on Chips. In *Proceedings of the 24th IEEE VLSI Test Symposium (VTS'06)*, pages 100–107. 18

[Taktak et al., 2008] Taktak, S., Desbarbieux, J.-L., and Encrenaz, E. (2008). A tool for automatic detection of deadlock in wormhole networks on chip. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 13(1) :6. 108, 115

[TILE-Gx, 2009] TILE-Gx (2009). Tilera tile-gx processor. `http://www.tilera.com/products/processors/TILE-Gx_Family`. 6

[Tran et al., 2006] Tran, X.-T., Durupt, J., Bertrand, F. B., Beroulle, V., and Robach, C. (2006). A DFT architecture for asynchronous networks-on-chip. In *Proceedings of the 11th IEEE European Test Symposium (ETS'06)*, pages 219–224. 34

[Ubar and Raik, 2003] Ubar, R. and Raik, J. (2003). Testing strategies for networks on chip. *Networks on chip*, pages 131–152. 34

[Vangal et al., 2008] Vangal, S. R., Howard, J., Ruhl, G., Dighe, S., Wilson, H., Tschanz, J., Finan, D., Singh, A., Jacob, T., Jain, S., Erraguntla, V., Roberts, C., Hoskote, Y., Borkar, N., and Borkar, S. (2008). An 80-tile sub-100-w teraflops processor in 65-nm cmos. *IEEE Journal of Solid-State Circuits*, 43 :29–41. 6

[Vermeulen et al., 2003] Vermeulen, B., Dielissen, J., Goossens, K., and Ciordas, C. (2003). Bringing communication networks on a chip : test and verification implications. *Communications Magazine, IEEE*, 41(9) :74–81. 33, 34

[Williams and PARKER, 1982] Williams, T. W. and PARKER, K. P. (1982). Design for Testability - A Survey. *IEEE Transactions on Computers*, 31 :2–15. 9

[Wu, 2003] Wu, J. (2003). A fault-tolerant and deadlock-free routing protocol in 2D meshes based on odd-even turn model. *IEEE Transactions on Computers*, 36 :1154–1169. 37, 39

[Zhang et al., 2010] Zhang, Z., Greiner, A., and Benabdenbi, M. (2010). Fully Distributed Initialization Procedure for a 2D-Mesh NoC, Including Off Line BIST and Partial Deactivation of Faulty Components. In *Proceedings of the 16th IEEE International On-Line Testing Symposium (IOLTS'10)*, pages 194–196. 50

[Zhang et al., 2008] Zhang, Z., Greiner, A., and Taktak, S. (2008). A reconfigurable routing algorithm for a fault-tolerant 2D-Mesh Network-on-Chip. In *Proceedings of the 45th Design Automation Conference (DAC'08)*, pages 441–446. 53

[Zhang et al., 2011] Zhang, Z., Refauvelet, D., Greiner, A., Benabdenbiy, M., and Pecheux, F. (2011). Localization of Damaged Resources in NoC Based Shared-Memory MP2SOC, using a Distributed Cooperative Configuration Infrastructure. In *Proceedings of the 29th IEEE VLSI Test Symposium (VTS'11)*. 19, 35, 51