Self-Diagnosis in Network-on-Chips

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Atefe Dalirsani

aus Iran

Hauptberichter:	Prof. Dr. rer. nat. habil. H.-J. Wunderlich
Mitberichter:	Prof. Dr.-Ing. Martin Radetzki

Tag der mündlichen Prüfung: 22.07.2015

Institut für Technische Informatik der Universität Stuttgart

2015

Acknowledgements

First and foremost, I would like to express my special appreciation and thanks to my advisor Professor Hans-Joachim Wunderlich, whose joy and enthusiasm for his research, hard working character, selfless time and care, and meticulous attention to detail was contagious and motivational for me and helped me to grow as a researcher. His brilliant advice and constant support on both research as well as on my career have been priceless.

The support of my colleagues in ITI throughout my doctoral project is very much appreciated, especially the co-authors of my papers for many discussions, feedback and collaboration. My colleague Dominik Ull deserves a big thank you for editing the German translation of the abstract. In addition, I would like to acknowledge German Academic Exchange Service (DAAD) for the doctoral scholarship and for administrative support during the first years of my stay in Germany.

Lastly, I express my deepest gratitude to my family, where the basic source of my life energy resides: words cannot express how grateful I am to my parents, for all they sacrificed for me, for their unconditional love and support all these years. This work, for what it is worth, is dedicated to them. I am also thankful to my brother particularly and my sister who have inspired me to strive towards my goals. And finally my loving husband, thank you so much, not only for being my proofreader and sounding board, but also for demonstrating patience over the last years, endless support and encouragement in the moments when there was nothing to keep me going.

Atefe Dalirsani
Stuttgart, May 2015

Abstract

Network-on-Chips (NoCs) constitute a message-passing infrastructure and can fulfil communication requirements of the today's System-on-Chips (SoCs), which integrate numerous semiconductor Intellectual Property (IP) blocks into a single die. As the NoC is responsible for data transport among IPs, its reliability is very important regarding the reliability of the entire system. In deep nanoscale technologies, transient and permanent failures of transistors and wires are caused by variety of effects. Such failures may occur in the NoC as well, disrupting its normal operation.

An NoC comprises a large number of switches that form a structure spanning across the chip. Inherent redundancy of the NoC provides multiple paths for communication among IPs. Graceful degradation is the property of tolerating a component's failure in a system at the cost of limited functionality or performance. In NoCs, when a switch in the path is faulty, alternative paths can be used to connect IPs, keeping the SoC functional. To this purpose, a fault detection mechanism is needed to identify the faulty switch and a fault tolerant routing should bypass it. As each NoC switch consists of a number of ports and multiple routing paths, graceful degradation can be considered even in a rather granular way. The fault may destroy some routing paths inside the switch, leaving the rest non-faulty. Thus, instead of disabling the faulty switch completely, its fault-free parts can be used for message passing. In this way, the chance of disconnecting the IP cores is reduced and the probability of having disjoint networks decreases.

This study pursues efficient self-test and diagnosis approaches for both manufacturing and in-field testing aiming at graceful degradation of defective NoCs. The approaches here identify the location of defective components in the network rather than providing only a go/no-go test response. Conventionally, structural test approaches like scan-design have been employed for testing the NoC products. Structural testing

targets faults of a predefined structural fault model like stuck-at faults. In contrast, functional testing targets certain functionalities of a system for example the instructions of a microprocessor. In NoCs, functional tests target NoC characteristics such as routing functions and undistorted data transport. Functional tests get the highest gain of the regular NoC structure. They reduce the test costs and prevent overtesting. However, unlike structural tests, functional tests do not explicitly target structural faults and the quality of the test approach cannot be measured. We bridge this gap by proposing a self-test approach that combines the advantages of structural and functional test methodologies and hence is suitable for both manufacturing and in-field testing. Here, the software running on the IP cores attached to the NoC is responsible for test. Similar to functional tests, the test patterns here deal only with the functional inputs and outputs of switches. For pattern generation, a model is introduced that brings the information about structural faults to the level of functional outputs of the switch. Thanks to this unique feature of the model, a high structural fault coverage is achieved as revealed by the results.

To make NoCs more robust against various defect mechanisms during the lifetime, concurrent error detection is necessary. Toward this, this dissertation contributes an area efficient synthesis technique of NoC switches to detect any error resulting from single combinational and transition fault in the switch and its links during the normal operation. This technique incorporates data encoding and the standard concurrent error detection using multiple parity trees. Results reveal that the proposed approach imposes less area overhead as compared to traditional techniques for concurrent error detection.

To enable fine-grained graceful degradation, intact functions of defective switches must be identified. Thanks to the fault tolerant techniques, fault-free parts of switches can be still employed in the NoC. However, reasoning about the fault-free functions with respect to the exact cause of a malfunction is missing in the literature. This dissertation contributes a novel fine-grained switch diagnosis technique that works based on the structural logic diagnosis. After determining the location and the nature of the defect in the faulty switch, all routing paths are checked and the soundness of the intact switch functions is proved. Experimental results show improvements in both performance and reliability of degraded NoCs by incorporating the fine-grained diagnosis of NoC switches.

Zusammenfassung

Network-on-Chips (NoCs) bilden eine Nachrichtenaustausch-Infrastruktur, welche Kommunikationsanforderungen der heutigen System-on-Chips (SoCs), die vorentworfene Blöcke (Intellectual Property, IP) in einem einzigen Chip integrieren, erfüllen können. Da der Datentransport zwischen den IPs durch NoCs realisiert wird, ist deren Zuverlässigkeit entscheidend für die Zuverlässigkeit des Gesamtsystems. Bei im Nanomaßstab gefertigten Transistoren und Leitungen werden transiente und permanente Ausfälle durch verschiedene Effekte verursacht. Derartige Fehler können auch in einem NoC auftreten, und so den normalen Betrieb stören.

Ein NoC umfasst eine große Anzahl schaltender Elemente (Switches), die eine aufspannende Struktur über den Chip bilden. Inhärente Redundanz von NoCs ermöglicht mehrere Pfade für die Kommunikation zwischen IPs. Graceful Degradation ist eine Eigenschaft des Systems, welches den Ausfall einer Komponente auf Kosten eingeschränkter Funktionalität oder Leistung erlauben kann. Wenn ein Switch in einem Pfad fehlerhaft ist, können im NoC alternative Pfade verwendet werden, um IPs zu verbinden, wobei das SoC weiterhin funktioniert. Zu diesem Zweck wird ein Fehlererkennungsmechanismus benötigt, um fehlerhafte Switches zu identifizieren und mit fehlertolerantem Routing fehlerhafte Switches zu umgehen. Da jeder NoC Switch aus mehreren Ports und Routingpfaden besteht, kann Graceful Degradation auch auf granularer Weise berücksichtigt werden. Ein Fehler kann einige Routing-Pfade innerhalb des Switches zerstören, wobei der Rest fehlerfrei bleibt. Auf diese Weise wird die Möglichkeit des Abtrennens von IP-Kernen als auch die Wahrscheinlichkeit eines getrennten Netzwerks verringert.

Die hier durchgeführte Forschung verfolgt effiziente Selbsttest- und Diagnose-Ansätze, sowohl für die Herstellung als auch für den Feldtest. Das Ziel ist die Graceful Degradation von defekten NoCs. Die Ansätze können nicht nur die Präsenz

eines Fehlers, sondern auch die Position der fehlerhaften Komponenten im Netzwerk erkennen. Üblicherweise werden Strukturtests wie beispielsweise ein Scan-Test für die Prüfung von NoCs eingesetzt. Strukturtests zielen auf vordefinierte Strukturfehlermodelle wie Haftfehler ab. Im Gegensatz dazu zielt ein Funktionstest auf bestimmte Funktionalitäten eines Systems, wie zum Beispiel auf den Befehlssatz eines Mikroprozessors ab. In NoCs prüfen Funktionstests Eigenschaften wie Routing-Funktionen und unverfälschten Datentransport. Funktionstests profitieren am stärksten von regulären Strukturen in NoCs. Bei diesen Tests werden die Testkosten reduziert, und ein Übertesten (overtesting) verhindert. Im Gegensatz zu Strukturtests betrachten die Funktionstests nicht explizit Strukturfehler, wodurch die Qualität des Testansatzes nicht bestimmt werden kann. Wir schließen diese Lücke mit einem Selbsttest-Ansatz, der die Vorteile der strukturellen und funktionellen Testmethoden kombiniert, und sich sowohl für die Herstellung als auch für den Einsatz im Feld eignet. Dabei ist die laufende Software der IP-Cores für die Testdurchführung verantwortlich. Testmuster umfassen, ähnlich wie bei Funktionstests, nur funktionale Ein- und Ausgänge von Switches. Zur Mustererzeugung wird ein Modell vorgestellt, das die Informationen über strukturelle Fehler auf die Ebene der Funktionsausgänge der Switches überträgt. Dank dieser Besonderheit des Modells wird eine hohe strukturelle Fehlerabdeckung erreicht, wie durch die präsentierten Ergebnisse gezeigt wird.

Um NoCs zur Laufzeit robuster gegen verschiedene Fehlermechanismen zu machen, ist nebenläufige Fehlererkennung notwendig. In diesem Zusammenhang beinhaltet diese Dissertation eine flächeneffiziente Synthesetechnik für NoCs, wodurch alle einzelnen kombinatorischen und transienten Fehler in Switches und deren Verbindungen während des normalen Betriebs erkannbar sind. Diese Technik beinhaltet eine Datencodierung und die standardmäßige, nebenläufige Fehlererkennung mit mehreren Paritätsbäumen. Die Ergebnisse zeigen, dass der vorgeschlagene Ansatz im Vergleich zu herkömmlichen Techniken zur nebenläufigen Fehlererkennung weniger Flächenaufwand verursacht.

Um feingranulare Graceful Degradation zu aktivieren, müssen intakte Funktionen von defekten Switches identifiziert werden. Dank fehlertoleranter Techniken können fehlerfreie Teile von NoC-Switches weiterhin eingesetzt werden. Allerdings fehlt in der Literatur ein Verfahren zur Bestimmung der fehlerfreie Funktion in Bezug auf die genaue Ursache einer Störung. Diese Dissertation umfasst eine neuartige feingranu-

lare Switch-Diagnose-Technik, die basierend auf struktureller Logik-Diagnose funktioniert. Nach der Bestimmung der Lage und der Art des Defekts in dem fehlerhaften Switch sind alle Routingpfade überprüft und die Korrektheit der intakten Switchfunktionen bewiesen. Experimentelle Ergebnisse zeigen beim Einsatz der feingranularen Diagnose Verbesserungen in der Leistung und Zuverlässigkeit defekter NoCs.

Contents

List of Figures

List of Tables

Abbreviations

ATE Automatic Test Equipment.

ATPG Automatic Test Pattern Generation.

BISD Built-In Self-Diagnosis.

BISR Built-In Self-Repair.

BIST Built-In Self-Test.

CED Concurrent Error Detection.

CLF Conditional Line Flip.

CNF Conjunctive Normal Form.

CRC Cyclic Redundancy Check.

DfT Design for Testability.

DSM Deep-Sub-Micron.

DWC Duplication With Comparison.

EDC Error Detecting Code.

FIFO First-In First-Out.

Flit Flow control unit.

FSM Finite State Machine.

Abbreviations

IP Intellectual Property.

MTBF Mean Time Between Failure.

MUX Multiplexer.

NoC Network-on-Chip.

PE Processing Element.

SAT Boolean Satisfiability.

SBST Software-Based Self-Test.

SDC Silent Data Corruption.

SLAT Single Location at a Time.

SoC System-on-Chip.

SUT Switch Under Test.

TMR Triple Modular Redundancy.

TPG Test Pattern Generator.

TRA Test Response Analyser.

TSC Totally Self-Checking.

VHDL Very High Speed Integrated Circuit Hardware Description Language.

VLSI Very Large Scale Integrated Circuit.

1

Introduction

Today's semiconductor industry has enabled integration of thousands of processing elements into a single die [Borkar07, Lindholm08, Nickolls10, Keckler11] to respond the ever-increasing request for high performance computations. As silicon technology downscales, delay and power dissipation of global on-chip interconnects become a serious bottleneck. Tackling the problems of intra-chip communication, global wire delay, and synchronization of several frequency islands, around 1999, Network-on-Chips (NoCs) emerged as a message-passing infrastructure to supersede traditional bus structures in huge System-on-Chips (SoCs) [Benini02, Pande05, Vangal08, Agarwal09].

The NoC contains a large number of switches and interconnects that form a communication infrastructure spanning across the chip [Dally03a]. Due to its regular geometry, several issues like communication performance, electrical properties and design and verification time of NoCs become most likely predictable [Jantsch03]. Moreover, lots of the NoC design, verification and test investments can be shared by many products that use the same NoC platform for communication. These features make the NoC a practical approach to fill the productivity gap of communication in today's massively integrated systems.

Despite several advantages of technology scaling, it causes some unpleasant side-effects: devices become more vulnerable to latent defects, variation and ageing effects, power supply noise, and crosstalk, resulting in transient or permanent failures of transistors and wires. Due to these failures, some switches and links of the NoC may

become non-functional. To maintain acceptable yield, such large scale structures must provide redundancy to tolerate spot defects. Several identical components in the NoC have paved the way for fault tolerance, making the NoC an appropriate alternative for robust communication over the chip. However, performance may be traded off against yield. The most popular example is speed binning for high-end processor chips. Another example is flash memories that may be delivered even if a few blocks are not functional and not accessible. Cache structures are inherently fault tolerant and defects can be masked by disabling the corresponding lines and reducing the cache capacity [Agarwal04].

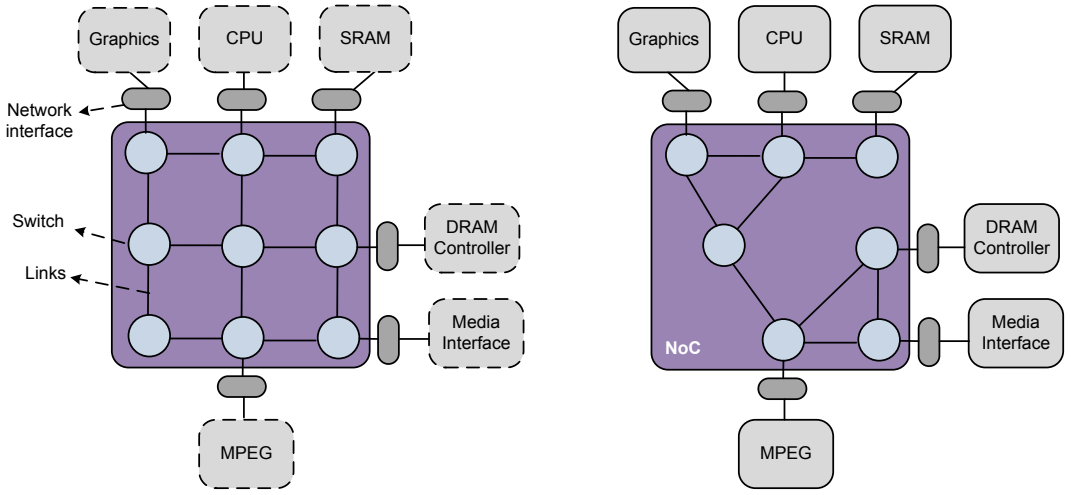
The property of tolerating the components' failures in a system at the cost of limited functionality or performance is called *graceful degradation*. Graceful degradation and fault tolerance are two sides of the same coin. Fault tolerant features must be available to discard failing components and perhaps replace them with the spare ones. In a naively designed system without fault tolerant features, even a single fault may cause a total breakdown. Due to the inherent redundancy of the NoC, defective links and switches can be discarded [Zhang08b] after testing. The NoC can compensate for this loss to a certain degree by fault-tolerant routing as long as available redundancies ensure connectivity, perhaps with degraded performance. Packets are routed over alternative paths to ensure connectivity among as many resources as possible [Zhang08b, Fukushima09, Lin09, Palesi10]. This is an example of graceful degradation in NoCs.

The purpose of this study is to conduct efficient test and diagnosis procedures both in the manufacturing phase and in the field to not only improve the robustness of the NoC products but also enable graceful degradation in a granular way. A prerequisite for graceful degradation of NoCs is to identify the location of defective *components*, which is called *Network Diagnosis*.

1.1 NoC Components

In the NoC, *switches* are connected to each other via interconnect *links* and constitute various kinds of NoC topologies. Figure 1.1 demonstrates a homogeneous (1.1a) and a

heterogeneous (1.1b) NoC topology that have established the communication platform of a sample SoC. The most commonly used homogeneous NoC topology is the two dimensional (2D) mesh grid in which every non-boundary switch has four neighbouring switches.



(a) Homogeneous NoC: 2D mesh **(b)** Heterogeneous NoC

Figure 1.1: NoC-based System-on-Chips

An NoC switch might have a local connection to one of the system resources. The so-called *network interface* modules provide the connection among the system resources and the network. Every NoC switch comprises a number of *ports* for data transmit and an internal control logic to manage the data flow. As shown in Fig. 1.2, the port consists of two channels, one for incoming data and one for outgoing data. Links establish the connection between incoming and outgoing ports of two adjacent switches or a switch and a network interface.

1.2 Network Diagnosis

Structural testing of VLSI circuits is conducted based on a fault model (e.g. stuck-at faults), and proves the absence of defects represented by this type of faults. *Diagnosis* comprises both fault detection and fault location. Based on the test responses, the suspects, which may cause the erroneous output, are identified. The resolution of a

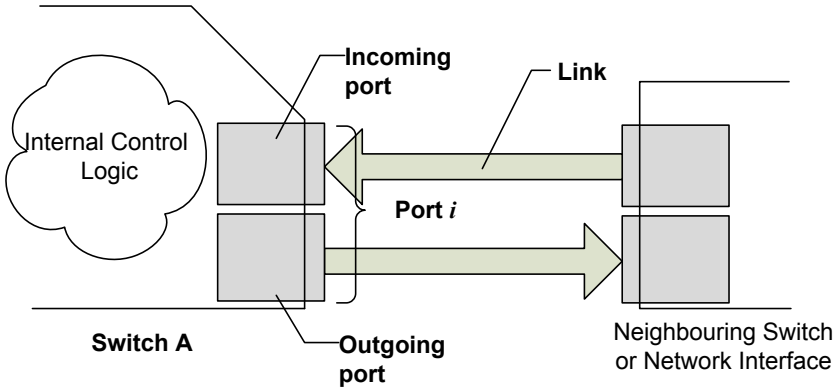


Figure 1.2: Network components: switches, links, and ports

diagnostic reasoning algorithm with the given test set describes the number of possible culprits [Holst09]. High fault coverage of the test set and high resolution of the diagnosis algorithm are the basic requirements of any VLSI test and diagnosis approach [Maly87, Gruning91, Holst09]. The quality of test and diagnosis depends on the accuracy of the fault model. If real electrical and physical failures on the chip are targeted, the fault model must accurately correlate with them [Abraham86, Maly87]. More accurate fault models impose higher test and diagnosis costs, but the outcome contains more fine-grained information about the exact defect location in the chip.

The fault model can be defined at different abstraction levels, from architectural level down to physical chip level. A component-oriented fault model at architecture level makes the test and diagnosis of complex systems tractable. However, some defects at physical chip level may not be identified by high-level tests. Moreover, the diagnosis result identifies a component as the culprit rather than a more precise defect location, for example a signal at gate-level, and therefore the quality of such test and diagnosis methods is compromised.

To increase the quality of test, the fault model at higher level must be realistic and introduce representatives for various defects at lower levels. However, in practice, it is impossible to have a representative for every defect at the physical chip level. Structural fault models (mainly at gate-level) can explain various kinds of real physical defects. Thus, we rely on this fault model and conduct test methods that target structural faults. For the network diagnosis, we rely on the component-oriented fault

models with representatives for the target structural faults.

Figure 1.3 illustrates the conceptual view of network diagnosis. From the physical chip point of view, a test approach must deal with all faults from the predefined structural fault model, ensuring the test quality. From architectural level, it is sufficient to locate the defective component, which is essential for graceful degradation. The defective component is introduced as the abstracted fault model for network diagnosis. During network diagnosis, every structural fault is mapped onto at least one defective component.

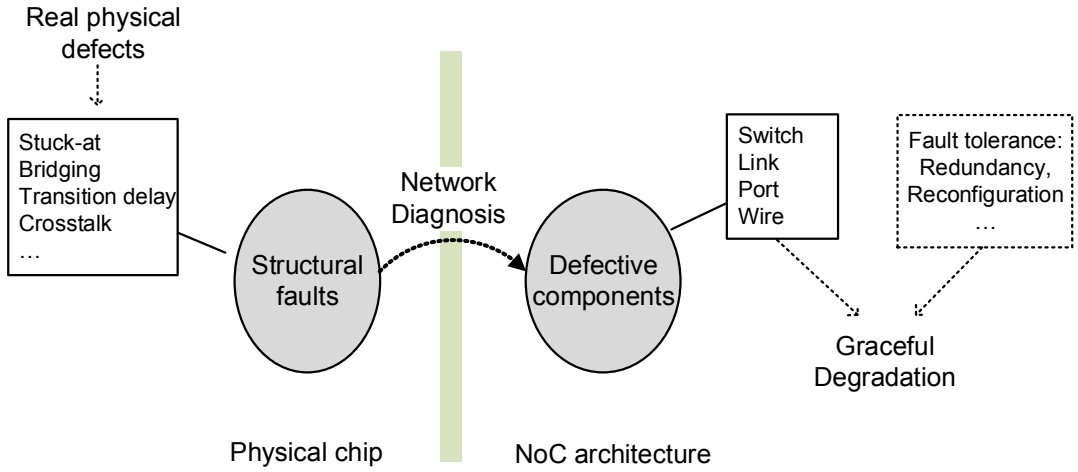


Figure 1.3: Conceptual view of network diagnosis: from structural faults at physical chip level to defective components at architecture level

Defective switches and links are typically used as the component-oriented fault models for network diagnosis. For a fine-grained diagnosis, even smaller parts (e.g. switch ports) can be considered as the abstracted fault model. This indeed imposes higher costs for more granular diagnosis, but allows to use more fault-free parts of the network and thus mitigating the drawbacks of the resource loss due to the faults. In other words, it enables graceful degradation in a rather granular way. System requirements determine the required granularity of the diagnosis at architectural level, i.e. which component to choose as the abstracted fault model.

With respect to the abstracted fault model and the property of having a representative for every structural fault in the network, we define network diagnosis at two

levels:

- *Switch-accurate*: A faulty switch is the abstracted fault model. Network diagnosis indicates the location of the faulty switch in the network. By mapping the faults in the incoming links of each switch onto the faults in the incoming ports of that switch, the model introduces a representative for structural faults in the interconnect links. Faults in the inter-switch links are equivalent to those in the incoming or outgoing switch ports at the two endpoints of the link. For example as depicted in Fig. 1.4, faults in the link L_1 are equivalent to the faults appear in the outgoing port 2 of switch B or the incoming port 4 of switch A. Here, faults on L_1 are mapped to the downstream switch, i.e. switch A.

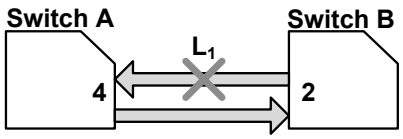


Figure 1.4: Faults in the link are equivalent to the faults in the switch ports

- *Port-accurate*: A faulty switch port is the abstracted fault model. Defective switch ports delegate structural faults inside the switches as well as in the links. A structural fault in an interconnect link can be represented by two defective switch ports at its two endpoints. The port-accurate diagnosis can find representatives for structural faults inside the switches as well, which is discussed in Chapter 4 of this dissertation.

1.2.1 Switch-Accurate vs. Port-Accurate Diagnosis

By disabling defective switches, the overall performance of the system decreases, because some resources get isolated from the network and the traffic must be diverged over the remaining resources. Moreover, in the worst case, the system may fail entirely if it loses the connection to its vital resources.

In the design of a system which is subject to faults, considering only the performance or the reliability constraints will result to nonoptimal solutions [Meyer80]. A

fault may cause permanent changes in the system structure. The system may compensate faults by discarding some components at the cost of performance degradation. The ability of a system to operate at a degraded performance level (or degraded functionality) postpones the time to failure and hence increases the reliability as well. Therefore, *performability* has been introduced as a composite measure of performance and reliability [Meyer80] to evaluate the ability of maintaining a system's functionality while tolerating a certain number of defects.

Performability can be increased, if not only complete switches and links but also defective ports can be disabled individually [Kohler10]. Figure 1.5 gives an example for such a scenario. Let's suppose Switch $(1, 1)$ is identified to be defective by the production test and the subsequent network diagnosis (switch-accurate). If it is disabled, Core B will be isolated and three network links become unusable. In addition to performance degradation, this may cause a permanent system failure, as Core B is disconnected. Now, let us assume it is possible to prove that the defect in Switch $(1, 1)$ affects only the northern port (port-accurate diagnosis). In this case, the northern port can be disabled while the others are still in use. Furthermore, Core B is not isolated. This improves the performability of the defective system.

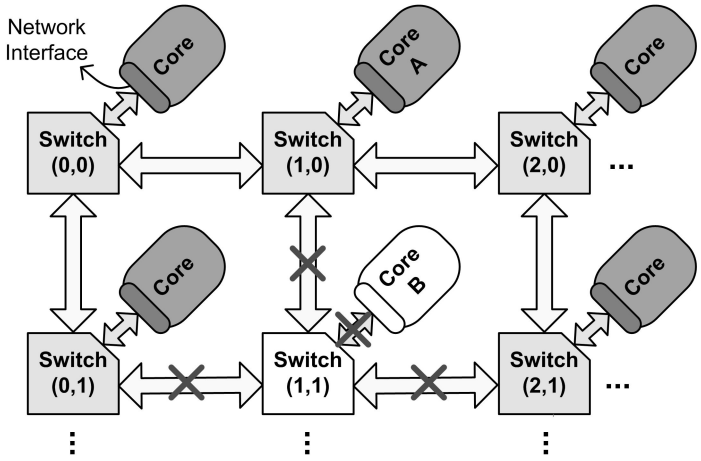


Figure 1.5: A defective switch: performance and reliability degradation

1.3 Self-Diagnosis of NoCs

The hardware structure of each switch as well as communication links are tested after production [Grecu05], at power-up or on demand [Fick09b] so that the defective components are identified. Test infrastructures like scan-design are typically employed to apply structural test patterns in an acceptable test time [Amory05b, Hosseinabady07]. Structural test targets faults of a predefined structural fault model like stuck-at faults and allows measuring the fault coverage as a quality metric. In contrast, functional testing targets certain functionalities of a system, for instance instructions of a processor. Functional testing reduces test time, enables at-speed test, and prevents overtesting. However, as it is impossible to test the functionality completely, it is hard to estimate the final quality of such a functional test, and the structural fault coverage is rather limited. The techniques for functional testing of NoCs rely on a set of functionalities of NoC switches, for example error free data transmit or correct routing decisions [Raik09]. Although functional testing of NoCs provides some confidence about the correct functionality of certain parts of switches, a high structural fault coverage is not explicitly targeted.

1.3.1 Software-Based Self-Test

The benefits of structural and functional testing are combined by a so-called structural Software-Based Self-Test (SBST) in microprocessors, where ATPG provides deterministic, structural test patterns which are transformed into arguments of a sequence of valid instructions. In a similar way, by transforming deterministic test patterns into valid packets of NoCs, a structural SBST approach is tailored for NoCs, which is one of the major contributions of this dissertation. Structural faults of NoC switches and interconnect links are targeted and tested by test patterns in the form of NoC-packets. For test pattern generation, sequential behaviour of the switch is modelled by replicating the combinational switch logic for several time intervals (i.e. time-frame expansion). Using this model, the impact of each structural fault on the functional outputs of the switch in the specified time period can be taken into account.

In the SBST approach, processing elements surrounding the switch under test are

reused for test generation and evaluation as well as test access. When testing a switch using its surrounding processing elements, test patterns are transported via the links so that faults in the links will be also detected. Here, faults in the switch and its incoming links are mapped to a faulty switch and therefore, the network diagnosis is switch-accurate. The SBST approach suits both manufacturing and in-field testing as it reduces the test time and test hardware overhead without loss of structural fault coverage. Furthermore, as SBST relies on functional inputs and outputs of the switches, it offers all advantages of functional testing methods.

1.3.2 Concurrent Error Detection

A periodic test of NoC switches at start-up or on-demand is effective to locate defective components in-field. However, transient and permanent failures of transistors and wires, caused by variety of effects, affect reliability of the systems [Gielen08, Borkar05] as well as the NoC structure during operation. To make NoCs more robust, concurrent error detection is necessary to detect a fault as soon as it causes an erroneous operation of NoC switches or interconnect links. In this way, fault tolerant mechanisms can be activated immediately to avoid inappropriate consequences at the right time, preventing a system crash for example.

Concurrent Error Detection (CED) techniques are widely used to detect errors that appear during the circuit operation, irrespective of their temporal nature (permanent, transient, or intermittent) [Goessel08]. Duplication with comparison is a conventional approach for CED of single- and multi-bit errors. The theory of self-checking design defines two properties for CED techniques [Nicolaidis98] as follows: A circuit is *fault-secure* if for every fault of the target fault model, the circuit never produces an incorrect codeword output for the code input. Furthermore, a circuit is *self-testing*, if for every fault it generates a non-codeword output for at least one code input. Lastly, it is *Totally Self-Checking* (TSC) if it is both self-testing and fault-secure. A non-redundant fault-secure circuit is also TSC.

To conduct CED, the schemes based on hardware redundancy, such as duplication or triplication, have been tailored for NoCs [Frantz06b, Zhang08a, Yanamandra10]. But these methods impose a huge area overhead, as the NoC may integrate hundreds

of switches into a single chip. Indeed, replicating the switch without paying attention to its huge datapath elements leads to an unacceptable area cost.

Data encoding using error detecting codes in combination with data retransmission is widely used in NoCs to detect and correct errors during the system operation [Frantz06b, Ghofrani12, Lehtonen10, Rossi07, Palesi11, Bertozzi05, Grecu07a]. However, up to now, these methods are mainly devoted to detect faults in the inter-switch links and intra-switch datapath elements such as multiplexers and cannot ensure fault-secureness for the entire switch. A certain portion of the switch logic is dedicated to manage the flow of data, for example the routing algorithm, scheduling, and congestion control. Faults in these parts of a switch can be more severe than those in the datapath elements, which mainly cause data corruption. Data corruption may be detected and corrected by an error correcting code even at the system-level. However, faults in the control logic may have uncorrectable effects leading to a system crash.

The second major contribution of this dissertation is devoted to the synthesis of an area efficient fault-secure NoC switch. The method utilizes data encoding to detect faults in datapath elements and a low-area concurrent error detecting structure with multiple parity trees to handle faults not covered by data encoding. The resulting structure detects any error stemming from single-point combinational and transition delay faults in switches or interconnect links. By using the fault-secure structure for all switches in the network, a switch-accurate network diagnosis is achieved, as faults in the switch and its incoming links are mapped to faulty switches. The proposed fault-secure structure reduces the area overhead significantly as compared to the state-of-the-art approaches.

1.3.3 Structural Diagnosis of NoC Switches

The third major contribution of this study enables the port-accurate network diagnosis by conducting a fine-grained structural diagnosis in NoC switches to identify fault-free and faulty ports. Here, switch ports are considered as the abstracted model so that any fault in the network can be mapped to a faulty switch port. Although it is easily observed that a non-functional link can be described by non-functional switch ports at the two endpoints of that link, it is not straightforward to map faults inside the

switch to faulty switch ports. This is called *switch diagnosis* and enables fine-grained graceful degradation of NoCs. Recently, a few works have discussed a similar type of graceful degradation [Fick09b, Kohler10, Kakoee11b]. However, none of the previous studies have used the standard test flow and fine-grained structural diagnosis to reason about fault-free and faulty switch ports. The technique presented here bridges this gap by mapping the structural defect information to NoC switch functions. The detailed information of fault-free ports is used for reconfiguration, resulting in performability and yield improvement.

1.4 Applications of Network Diagnosis

Network diagnosis is an enhancement for both the manufacturing process and in-field testing to increase the profit margins of the manufacture. Figure 1.6 illustrates the role of network diagnosis in the manufacturing phase. Testing after production is inevitable in order to find fault-free devices. Moreover, additional tests might be required [Roy95] to be applied to some devices which have already passed the primary test. Some chips may fail very quickly thereafter [Barrette96] because of sensitive designs or process variations. As secondary tests are more expensive, the additional test is applied to only some devices of certain application domains. For example, stress test is applied to only those devices that are liable to operate in the high voltage and temperature condition. Burn-in test identifies the defects that may appear in the early life time (infant mortality problems) and is mandatory for the devices of the safety-critical application domains such as automotive or medical applications.

To increase the yield, structural test data of the failing devices is further used to perform structural diagnosis. Structural diagnosis is the process of locating the defect within the random logic of the circuit. It helps, on the one side, to refine the fabrication process and avoid the defects in the future fabricated chips. On the other side, it is used for an effective binning of reconfigurable systems, which can tolerate a certain number of faulty components. For example, in flash memories when structural diagnosis pinpoints the defect location, only the defective memory block is disabled and the device is delivered in a lower capacity. Similarly, during manufacturing of NoC-

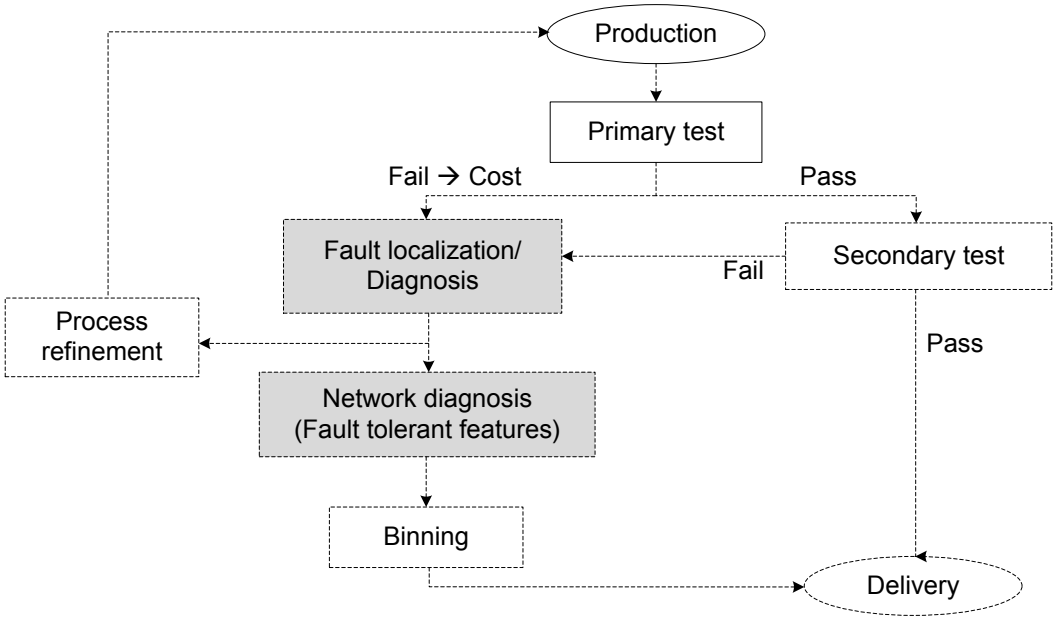


Figure 1.6: The role of testing and diagnosis: manufacturing phase

based systems, the information about the defect location can be used for port-accurate diagnosis that in turn enables fine-grained graceful degradation of NoCs leading to an efficient binning. The impact of switch-accurate and port-accurate diagnosis to maintain the reliability and performance of defective NoCs is investigated in Chapter 7 of this dissertation.

Despite manufacturing test, defects that appear in the lifetime of the system are still unavoidable. Steadily decreasing the feature size of the transistors and interconnect wires increases the probability of having latent defects and test escapes. Moreover, devices are more vulnerable to transient faults. Any failing device in the field, even with no security risk, imposes additional costs on the company for customer support services and threats the company's reputation as well.

As illustrated in Fig. 1.7, network diagnosis can be also employed during the lifetime. In the presence of fault tolerant features, the system might be reconfigured to bypass the defective components identified by network diagnosis. A reconfigured system continues its intended operation at a degraded performability level. A failing system should be returned to the company for repair or replacement. With the port-accurate

diagnosis enhancement however, the chance of discarding a defective switch decreases and thus the system failure may be postponed. In other words, the Mean Time Between Failure (MTBF) [Lala01a] increases. Moreover, the defective NoC operates at a higher performance level as compared to the case with switch-accurate diagnosis only. The above discussion reveals that the proposed techniques have a versatile application as they can be well integrated in the standard test flow for manufacturing test and during the lifetime. The diagnosis approaches provide a means to fine-grained reconfiguration and outperform classical NoC test and diagnosis approaches.

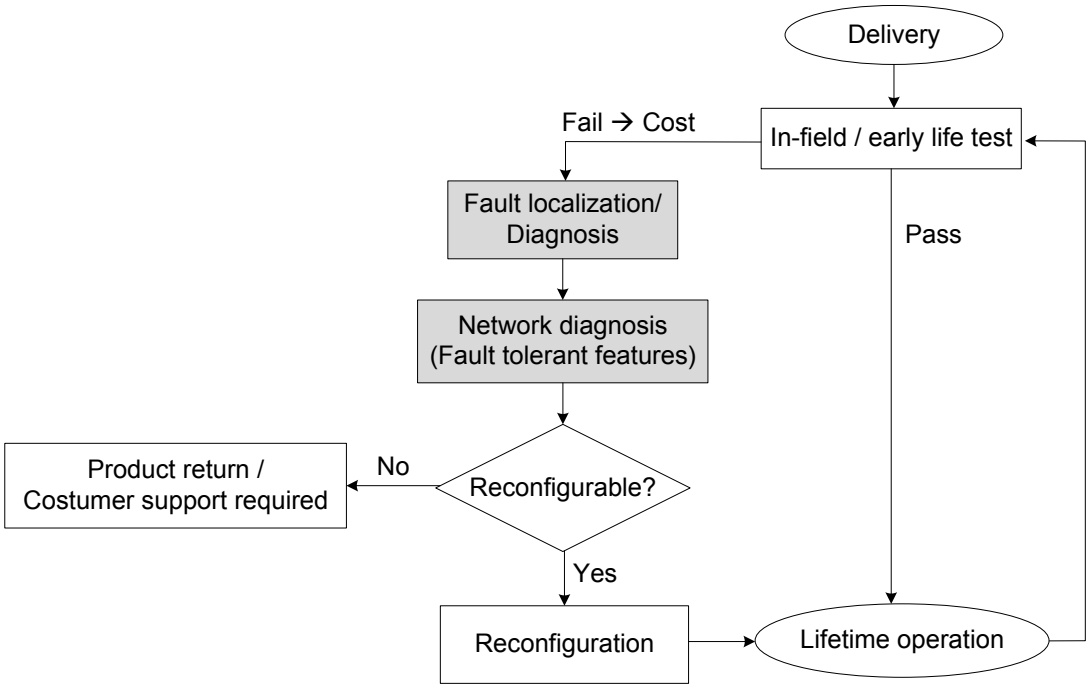


Figure 1.7: The role of testing and diagnosis: after delivery

1.5 Dissertation Overview

Chapter 2 discusses the basic concepts, terminologies and notations that are used in this dissertation. The structure of the Network-on-Chip switches, principles of the fault modelling and some famous fault models, as well as common aspects of test and diagnosis are presented in this chapter. In addition, the performability concept and

fault tolerance is discussed and the principle of the Boolean satisfiability, which is used for the proposed SBST pattern generation and fault-secure synthesis schemes, is introduced.

Chapter 3, state-of-the-art, reviews briefly available fault tolerant techniques that enable usage of functional switches and functional switch ports in the network. Furthermore, it explores state-of-the-art test and diagnosis approaches for NoCs, in-field testing techniques, and the methods that can identify the defective switch functions. This chapter emphasizes existing gaps which are bridged by this dissertation work.

Chapters 4 to 6 are devoted to the proposed techniques for network diagnosis. Figure 1.8 gives an overview of how the discussed aspects of this dissertation enable diagnosis of NoCs both in the production phase and in the field.

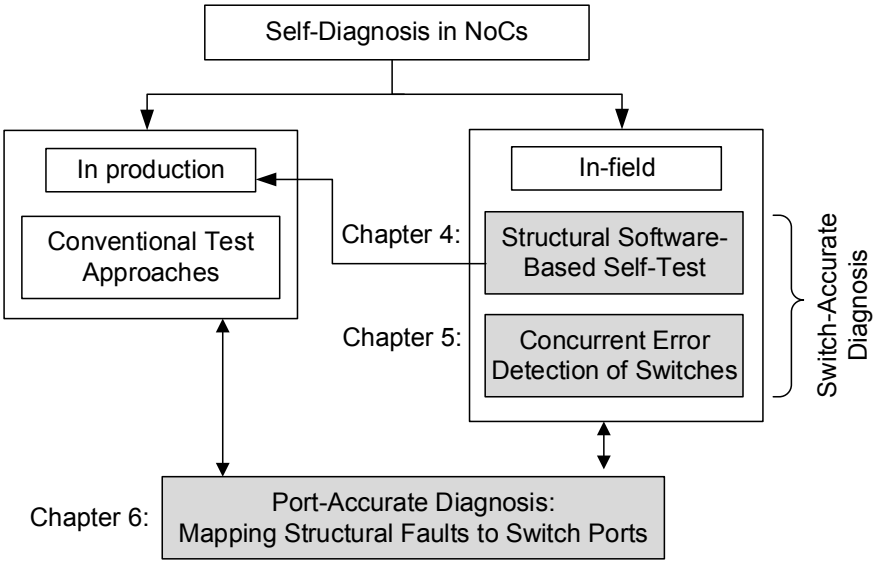


Figure 1.8: Dissertation Overview

In the production phase, the proposed SBST approach (Chapter 5) can supersede the conventional approaches as it offers the advantages of functional testing while targeting structural faults. When product requirements necessitate graceful degradation in a more granular way, the port-accurate diagnosis can be performed for failing chips. The diagnosis procedure analyses structural test fail data to identify the defect location and its nature within the random logic of the switch. For each possible function of the switch, the port-accurate diagnosis method (Chapter 4) determines whether it may be

corrupted by the identified fault. This information is stored in a dictionary for the fast look-up after applying the test.

During the lifetime, the structural SBST technique conducts periodic testing of NoC structure to identify defective components. Lower test time and less test hardware make the SBST approach appropriate for in-field testing. When fine-grained graceful degradation is necessary, test fail data can be analysed in an embedded diagnosis service point to identify the fault location. Finally, corresponding to the fault location, the defective switch ports are looked up from a precomputed diagnosis dictionary and the network is reconfigured in a fine-grained way. If the system requires higher robustness, network diagnosis can be performed concurrent with the normal operation by employing the proposed fault-secure switch structure (Chapter 6). This guarantees detecting any failure in the NoC infrastructure stemming from single point transient, permanent, or intermittent faults.

Investments to achieve fine-grained reconfiguration of NoCs would be worthwhile solely when it offers a noticeable performability improvement in faulty cases. Chapter 7 of this dissertation explores by simulations the impact of switch-accurate and port-accurate diagnosis on the performability of defective NoCs.

2

Concepts and Terminology

This chapter contains the basic definitions and terminologies on which this dissertation is built. After introducing the NoC switch structure and basic terminologies of NoCs, we discuss fundamental concepts of fault modelling, test and diagnosis. Next, performability of the NoC is defined in terms of the reliability and performance. This chapter comes to its end by introducing the Boolean satisfiability problem and some definitions, that constitute the foundations for the SBST pattern generation algorithm and the fault-secure synthesis presented respectively in Chapters 5 and 6.

2.1 Network-on-Chip Basics

In Network-on-Chips (NoCs), instead of assigning dedicated resources to each communication pair of system's cores, communication resources (switches and links) are shared among all cores. Each message, being sent from the source toward the destination, passes through several switches across the shared resources. The composition of the switches in the network determines the network *topology*. With respect to the topology, there can be several possible paths, i.e. sequences of switches and links, for message transfer between a pair of cores. In the network shown in Fig. 2.1, three communication paths between the source and the destination are shown. The *routing algorithm* decides which path to take from a source to a destination. As the length of

the path influences the message latency, routing algorithms usually choose the shortest path. However, this is not the only parameter to be considered during routing. A well-designed routing algorithm should also balance the traffic load across the network. Because of an imbalanced load, some network resources might be loaded with a lot of data so that an increase in the network load does not lead to the throughput improvement, rather it may degrade the throughput. This is called *congestion*.

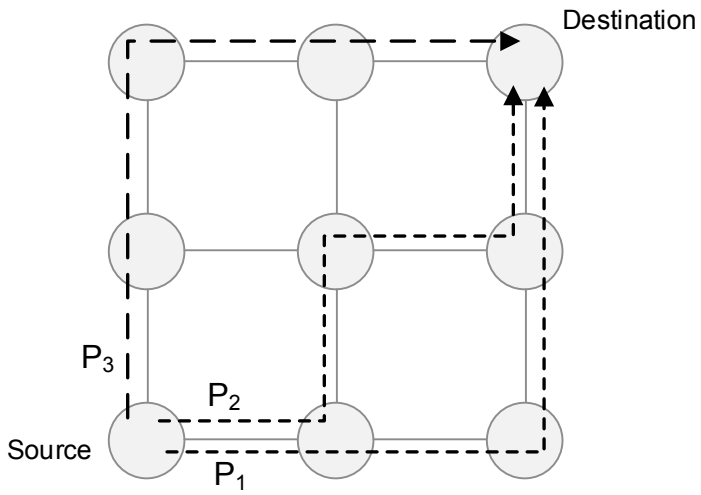


Figure 2.1: Several possible paths from a source to a destination

2.1.1 Switch Structure

Figure 2.2 shows a typical NoC switch and two interconnect links. The NoC switch incorporates several ports (e.g. five ports in the 2D mesh topology), and some controlling logic to manage the data flow between the incoming and outgoing ports. Each switch is connected to its neighbouring switches and the local core via interconnect links. Each interconnect link consists of some parallel data bits in the width of a *phit* (physical digit) and the handshake signals. The phit is the unit of data that can be transported over the link in one cycle. Data is transferred via parallel data bits and enters a switch through the data inputs of the switch port. It is stored into the incoming buffers to be routed afterwards.

A *Packet* is the basic unit of data which is transported over the network. In order to

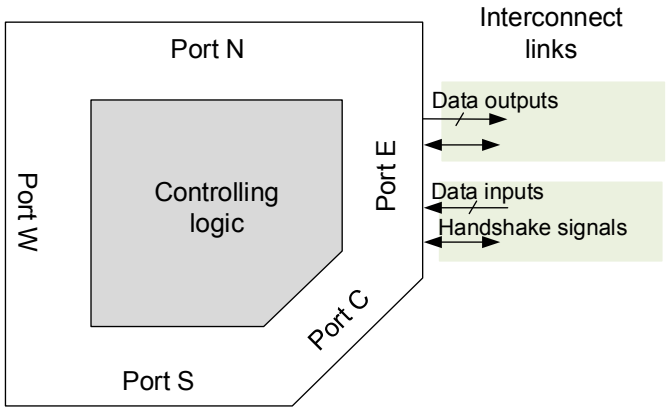


Figure 2.2: A typical NoC switch and interconnect links

improve the efficiency of the resource allocation, packets are divided into equal-sized bit groups which are called *flit* (flow control unit). To simplify the protocols, the flit and phit sizes could be considered identical.

Figure 2.3 presents the packet structure and its flits format. The network allocates the routing path to the entire packet. Flits of a packet are transported contiguously over the allocated routing path. They arrive one after the other via the data inputs of the switch port. The first flit of a packet is called *head* flit and is followed by an arbitrary number of *data* flits. The last flit is called *tail* flit. The flits can be identified by means of a few control bits named *flit id*. For example, 01, 10, and 00 can be used as the flit identifiers for the head, tail, and data flits. The rest of the bits carry the message or routing information. The flit transfer may be interrupted for a few cycles when the downstream buffers are full.

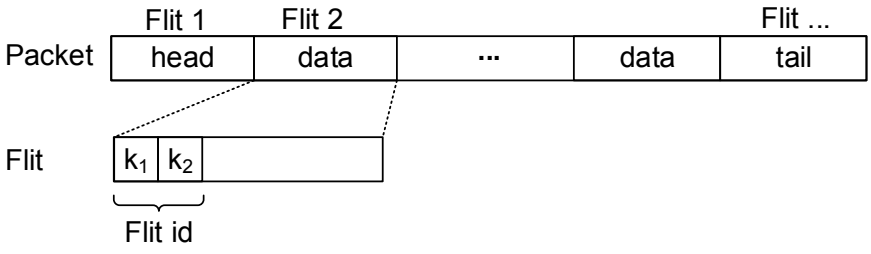


Figure 2.3: NoC packet format

Figure 2.4 shows the internal structure of the NoC switch. The router, the crossbar and the port controllers are the *controlling logic* of the switch (grey area in Fig. 2.4).

The port controllers generate and process the handshake signals of the ports. The *switch interface* (white area in Fig. 2.4) implements a circular *FIFO* (First In First Out) to store the incoming flits. The outgoing data is passed through an *output buffer*. The switch interface may include additional logic to control the data flow across the physical link, for instance additional logic to adjust the flit size to the phit size or vice versa. The interface is modified according to the network specifications. Some previous works in the literature have used the term 'router' referring to the entire switch logic. In this dissertation, the word 'switch' is used wherever the entire logic of the switch is discussed and 'router' refers to the router logic only.

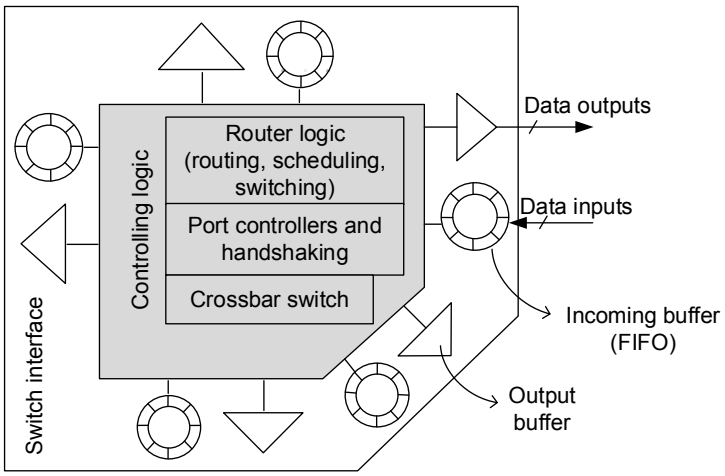


Figure 2.4: A typical NoC switch: the internal structure

The router logic implements the routing algorithm and the switching mechanism and processes the incoming ports in a round robin fashion. The incoming packets, stored in the FIFO buffers, are analysed in the routing logic. According to the destination address of the head flit, the routing algorithm computes the outgoing port through which the packet should be sent out. When the outgoing port is not occupied by another packet, the crossbar is set up to establish the route between the incoming and the outgoing port. This setup is preserved for the packet length.

The designed switch here follows the *wormhole* flow control policy for the resource allocation. Accordingly, the switch does not wait for the entire packet to arrive, instead, it starts forwarding the data as soon as the routing decision is made and the target outgoing port is idle. One primary advantage of the wormhole switching as compared

to other switching techniques [Duato03a] is the smaller buffer size, since just a few flits must be stored at each switch. When a port in the routing path is busy, the packet stalls immediately, that is its flits are stored in either one or several switches in the path.

The logic for the port controller and handshaking is built up for every port separately. This design technique enables simple modification of the number of switch ports according to the target architecture. Figure 2.5 demonstrates the structure of the port controller and handshaking for one switch port. To build the outgoing channel, the port comprises a multiplexer to select data from any other incoming ports. The multiplexers of all ports build up implicitly the switch crossbar. The port controller of each port analyses the data in its incoming FIFO and transfers the top element of the FIFO as the next flit to be processed by the router. It also generates the corresponding handshake/control signals announcing the current status of the incoming and outgoing ports.

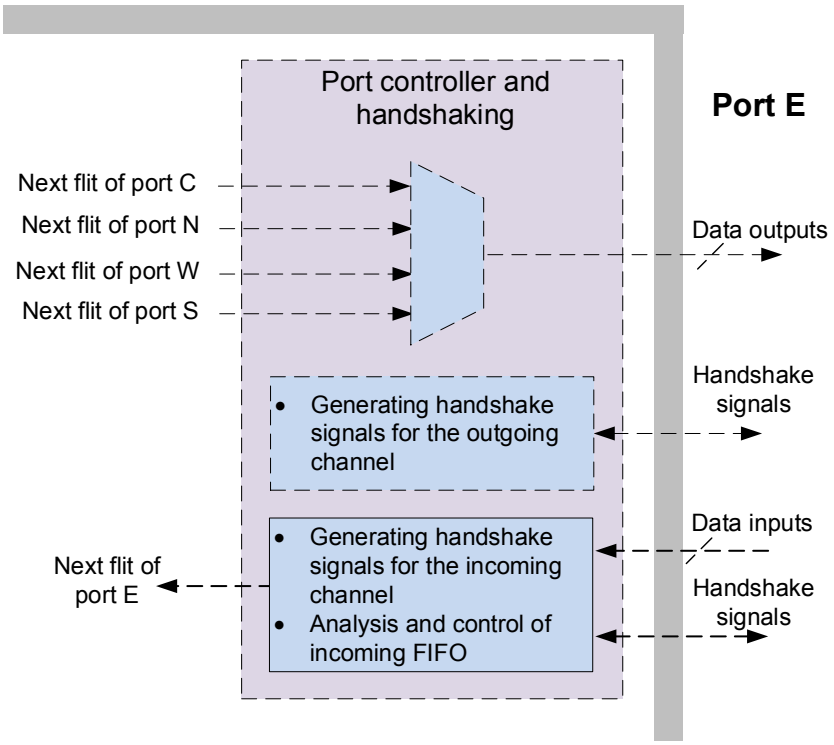


Figure 2.5: The port controller and handshaking logic for one switch port

2.1.2 Routing Algorithms

Deterministic routing algorithms [Dally03b] such as XY routing are widely used in the NoCs because of their simple implementation. Deterministic routings always establish the same path between a pair of switches and commonly choose the shortest path. As the path is static, such routing cannot deal with a faulty component in the network. Besides, the routing is quite incapable of balanced traffic distribution. In contrast, *adaptive routing* algorithms [Dally03c] choose the routing path with respect to the current network state. They bring the possibility to make an alternative decision when a faulty component appears in the path [Duato03b, Agarwal09]. Moreover, adaptive routings can distribute the local traffic in a balanced way such that congestion is avoided.

When developing the adaptive routing algorithms, deadlock and livelock [Ni93, Dally03d] must be prevented by making relevant rules and restrictions. *Deadlock* occurs when some packets get trapped in a waiting loop, that is every packet in the loop is waiting for some resources which have been occupied by some other packets in the loop. *Livelock* occurs when packets keep moving but do not proceed to their destination. Almost all adaptive routings use a deadlock avoidance mechanism (i.e. by eliminating cycles in the dependency graph [Dally87]) to prevent deadlock. However, a monitoring mechanism can be developed to detect deadlock situations and correct them by packet dropping for example. Some networks set a time to live parameter for every message. The packet is dropped when its time to live expires. This method is usually used to recover from the livelock situation as well.

There exist two mechanisms to implement the routing algorithm: table lookup, and algorithmic routing implemented as a combinational logic circuit. For deterministic routings, the routing table contains a single entry per destination, while for adaptive routings, multiple entries per destination are provided. With respect to the network state, the appropriate entry is looked up. The routing decision may be made either at each switch or once at the source terminal. The latter is called *source routing* in which the entire routing path is precomputed at the source terminal and the routing information is appended to the packet content.

2.2 Defects, Faults and Errors

Downscaling the feature size of the transistors makes the products more sensitive to the variations in the manufacturing process and in the lifetime. Small deviations of the actual dimensions of the optical masks due to mechanical or thermal stress for example, or dust particles in the environment may change the position and the amount of materials during etching, deposit and implant. This may permanently change the functionality of a transistor or may let it function only under specific conditions [Segura04]. A physical disorder in the silicon chip is called *defect*. Common examples are short and open defects due to extra or missing materials as shown in Figure 2.6a. In addition, physical disorders may cause variations in one or a set of circuit parameters such as transistor length ratio, threshold voltage, or the space between closely located interconnect lines. Such disorders, categorized as *parametric defects*, mainly lead to speed-related malfunctions of the circuit due to variations of the signal rise and fall times [Segura04].

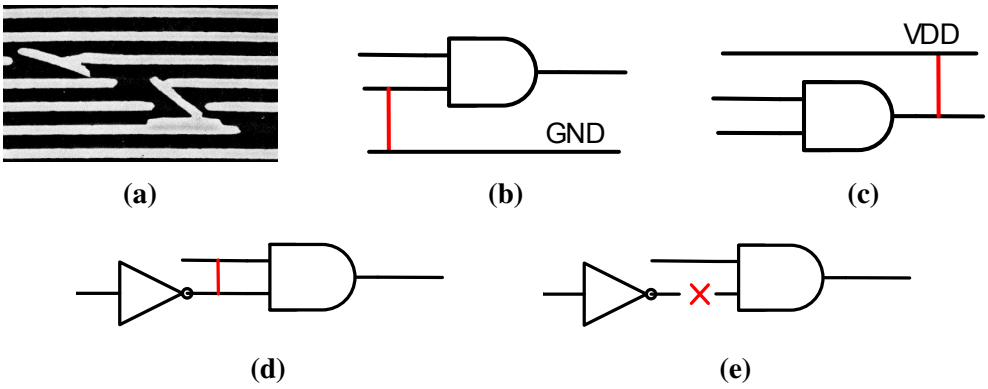


Figure 2.6: (a): short and breaks of metal lines [Maly87]. (b): short circuit, a signal line is tied to ground, i.e. stuck-at 0. (c): short circuit, a signal line is tied to VDD, i.e. stuck-at 1. (d): two signal lines are tied together, i.e. bridging fault. (e): open circuit, a floating signal line.

Testing after production is conducted to find and exclude the defective chips. However, it is difficult (expensive) to generate tests for real physical defects. Real defects are too numerous and often too difficult to be analysed. A *fault* is a representative for the structural disorders. It describes the observable behaviour of the defect as a change in the logic function and offers a useful abstraction model for the physical

defects. Traditionally, defect analysis and test generation has been at the gate-level and based on the gate-level netlist. In addition, faults have described alternations of logic signals in the gate-level netlist. Figure 2.6b to 2.6e demonstrate some faults at the gate level stemming from physical disorders. For decades, the stuck-at fault has been used as the fault model for defect analysis and test generation. A stuck-at fault ties a signal line to a constant logic value, either a logic 0 (stuck-at 0) or a logic 1 (stuck-at 1). More complex fault models are necessary to describe the behaviour of defects more accurately. Several fault models describe delay faults (altered gate delay) [Park88, Krstic98], bridging (unintended connection of two signal lines) [Acken91, Maxwell93, Lavo96, Renovell10] and crosstalk (capacitive coupling between neighbouring lines) [Chen97, Kundu05].

A generalized fault model called *Conditional Line Flip (CLF)* [Wunderlich10] presents a notation to express the defective behaviour of several structural disorders. It formally describes defects as pairs consisting of a location or victim line v and an activation condition. Whenever the condition evaluates to true, the value of the victim line is inverted. A CLF is noted by the name of the victim line v and an XOR-symbol followed by a condition clause as: $v^f := v \oplus [cond]$. In this notation, v^f represents the value of the victim line. The condition is of arbitrary nature (Boolean, temporal, or even random) and is defined as an arbitrary function over time. This fault model helps to develop generalized test and diagnosis schemes targeting arbitrary defects in the circuit. Table 2.1 presents the condition of the CLF fault model for some well-known fault types.

Table 2.1: Conditional line flip for some fault types

Fault type	victim signal	$[cond]$	notes
Stuck-at 0	a	a	
Stuck-at 1	a	$\bar{a}$	
Crosstalk	b	$(a_{-1} \oplus a) \wedge (a \oplus b)$	a: aggressor a_{-1}: the last value of line a
Static bridge a, b	a b	$f_a(b) \wedge (a \oplus b)$ $f_b(a) \wedge (a \oplus b)$	$f_a(b) \in \{0, 1, \bar{b}, b\}$ $f_b(a) \in \{0, 1, \bar{a}, a\}$

A crosstalk fault happens where a transition on an aggressor line a causes glitches

on a victim line b [Wunderlich10]. In the CLF model, a crosstalk is represented according to the following formula, in which a_{-1} denotes the last value of line a:

$$b^f := b \oplus [(a_{-1} \oplus a) \wedge (a \oplus b)]\,. \tag{2.1}$$

When a bridging fault occurs, two signal lines get the faulty values. A static bridge between two signal lines a and b is defined by the following generalized CLF formulas [Wunderlich10]:

$$\begin{aligned} a^f &:= a \oplus [f_a(b) \wedge (a \oplus b)] \\ b^f &:= b \oplus [f_b(a) \wedge (a \oplus b)] \end{aligned} \tag{2.2}$$

in which f_a and f_b are two Boolean functions that determine the actual behaviour of the bridge. There are exactly four basic expressions for each function: $f_a(b) \in \{0, 1, \bar{b}, b\}$ and $f_b(a) \in \{0, 1, \bar{a}, a\}$, resulting in 16 possible configurations for f_a and f_b [Wunderlich10]. For example, the 4-way bridge in which a and b swap the values is implemented by $f_a(b) = f_b(a) = 1$.

With respect to the duration of the occurrence, faults are categorized into three types: permanent, intermittent and transient. *Permanent* faults are due to the unwanted physical changes in the material and last forever. *Intermittent* faults stem from physical disorders as well, but may appear under certain circumstances in regular or irregular intervals. For example, a loose connection which causes a short circuit defect, may only appear in high temperature. *Transient* faults appear for a short time and may produce an unwanted change on the signal values. Radiations for example may alter the memory cell contents spuriously.

Testing after production is not sufficient to detect all possible defects in the chip, because on the one side such a test will be very expensive and on the other side some defects may not be detectable direct after production. A defect-free chip may experience other defects during the lifetime. Latent defects are not recognisable during the manufacturing test as they may appear after several operating cycles. Besides, some defects may appear due to the electromigration, material ageing or under certain physical or thermal conditions. A permanent, a transient or an intermittent fault may cause an incorrect state in the circuit or incorrect information in computation, which is called *error*. A wrong bit on a bus or in a memory cell, a wrong signature or an

incorrect check-sum are examples of errors [Wunderlich05]. An observed malfunction during the circuit operation is called a *failure*. It is noted that not every fault in the circuit causes a failure. For example, a fault in a memory cell that is never read, will not be recognized as failure.

2.3 Structural Test and Diagnosis

Finding a set of input assignments based on the structural circuit information such that the circuit responses in the faulty circuit differ from the non-faulty circuit is called *structural testing*. Each input assignment is called a *test pattern*. Structural testing targets faults of a predefined structural fault model like stuck-at fault. Depending on the required product quality, more complex fault models like delay faults, bridging faults or crosstalk faults may be applied as well.

The fault detection capability of a test pattern set is quantified with the *fault coverage* metric, which is defined as the ratio of the detected faults to the total number of faults in the circuit. For some faults, there exists no input assignment to distinguish the faulty and non-faulty circuits. These faults are called *undetectable* [Wang06a]. A circuit is called *redundant*, if it has some undetectable faults [Friedman67] and *non-redundant* otherwise. The *fault efficiency* is defined as the ratio of the detected faults to the total number of faults excluding those which are undetectable.

Automatic Test Pattern Generation (ATPG) algorithms generate test patters for combinational circuits with high fault coverage in a reasonable amount of time. To take advantage of advanced ATPG algorithms, the standard scan testing infrastructure is used as the Design for Testability (DfT) technique for sequential circuits [Williams83]. Scan design has become a typical step of VLSI design flow, because it reduces the complexity of sequential test pattern generation to combinational ATPG with high fault coverage [Wang06b].

In *full-scan* design, all storage elements of a sequential circuit are converted into scan cells. A *scan cell* (see Fig. 2.7a) has two driving inputs, one is driven by the circuit's combinational logic (data input) and the other one is driven by the output of another scan cell (scan input). A third input (mode selection) selects then the data

input in the normal/capture mode, or the scan input in the shift mode. Scan cells are connected to each other forming one or more shift registers called *scan chains*. As shown in Fig. 2.7b, the scan input of the first scan cell in the scan chain is connected to a primary input and the output of the last scan cell is connected to a primary output. As a result, all storage elements can be loaded with test data using a primary input. Furthermore, the test responses can be captured by shifting out the stored values in the scan chains. This reduces the complexity of test pattern generation as the storage elements become controllable and observable [Wang06b]. The access time is linear to the length of the scan chain.

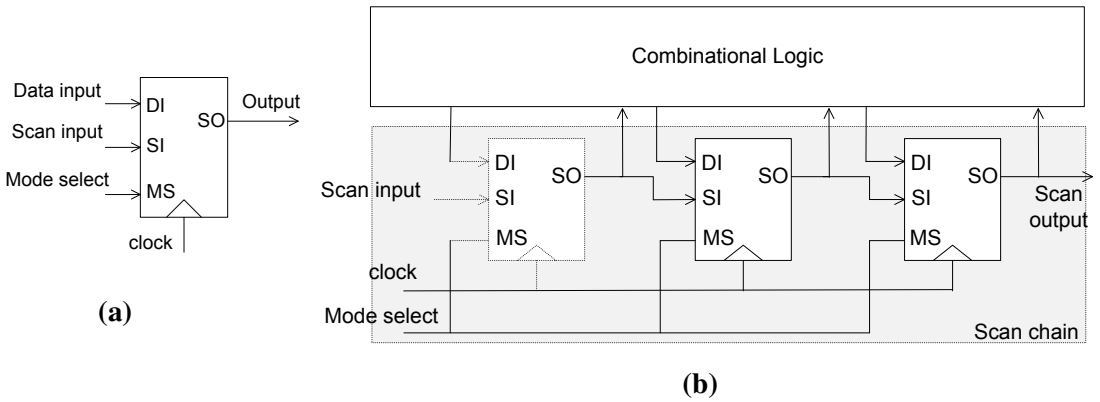


Figure 2.7: (a): scan cell. (b): scan chain over the storage elements of a sequential circuit

Structural logic diagnosis [Huisman05, Holst09, Bartenstein01, Huisman04] uses the structural test fail data to locate the defects within the random logic of faulty circuits. For many diagnosis applications, it is sufficient to locate one fault site, to guide physical failure analysis or to provide data for volume diagnosis [Hora02]. However, some applications need to have evidence that the remaining circuit structures are defect-free. In this case, the diagnosis algorithm has to classify the faulty behaviour of the circuit to be completely explainable by the identified fault sites [Holst09]. Otherwise, some unknown interactions among multiple fault sites might be present in the circuit which are not part of the diagnosis result.

2.4 Performability

A degradable system compensates faults in the system components at the cost of degraded performance [Goyal87]. In such a system, fault tolerant features are capable of isolating defective parts or replacing them with the spare ones. According to this definition, the NoC is a degradable system. Several fault tolerant features [Radetzki12] can identify and bypass defective switches and links so that the system resources can still communicate in the presence of faulty components. However, the communication performance degrades, because less communication resources (switches and links) are available.

Design of a degradable system by relying only on performance constraints and without considering inevitable failure effects, results in a drastic performance loss when a defect appears. At the other extreme, relying only on dependability constraints without considering the performance, tends to be pessimistic. Therefore, to adequately characterize degradable systems, both performance and dependability must be measured in a composite metric, which is *performability* [Meyer80].

As defined in [Laprie85] dependability is *"the quality of delivered service such that reliance can justifiably be placed on this service"*. On the terminology, dependability encompasses five attributes: *reliability*, *availability*, *safety*, *integrity* and *maintainability* [Avizienis04]. With respect to the continuity of the correct service, dependability means reliability [Laprie91, Avizienis04]. In the literature and in practice, reliability is known as the probability of correct functionality of a system for a specific period of time [Lala01a]. Thus, it can delegate the dependability concept in performability measurements of NoCs. In the rest of this section, common aspects of the reliability and performance of NoCs are discussed.

2.4.1 Reliability

The number of failures per time unit is called *failure rate*. The Bathtub curve shown in Fig. 2.8 [Lala01a] represents the variations of the failure rate of the electronic components over time. A high failure rate is expected in the early life period (also known

as infant mortality time), since imperfect manufacturing tests may cause some devices not to work as intended. During the useful life period of a device, the failure rate is almost constant. The failure rate starts growing again in the last life cycles, known as wear-out time period.

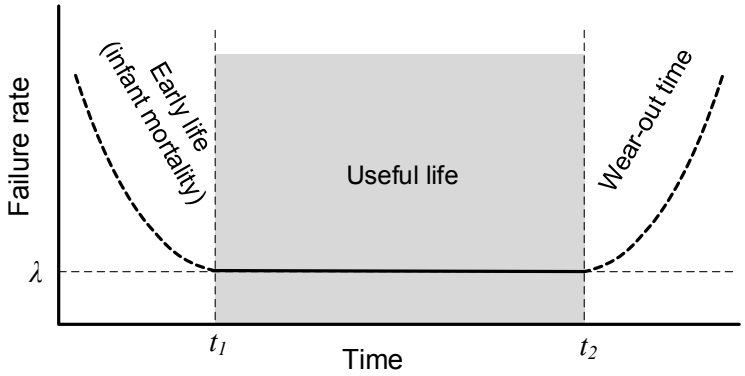


Figure 2.8: The Bathtub curve, failure rate variations over time

According to the exponential failure law [Wang10], the reliability of a component in the useful life period, when the failure rate λ is relatively constant, is expressed by:

$$R(t) = e^{-\lambda t}\,. \tag{2.3}$$

The reliability of a system depends on the reliability of its components. The reliability of a system, R_{sys}, containing N components in series, in which the failure of any individual component leads to the system failure, is defined as:

$$R_{\text{sys}} = \prod_{i=1}^{N} R_i\,, \tag{2.4}$$

where R_i is the reliability of the component i, and components' failures are independent. The reliability of a parallel system, in which the system fails only if all of its components fail, is defined as:

$$R_{\text{sys}} = 1 - \prod_{i=1}^{N} (1 - R_i)\,. \tag{2.5}$$

A complex System-on-Chip (SoC) cannot be recognized as a pure series or parallel system. In general, the reliability of an SoC, R_{SoC}, consisting of N components, can be defined as a function of the reliability of its components (i.e. R_i) as:

$$R_{\mathrm{SoC}} = f(R_1, R_2, ..., R_N)\,. \tag{2.6}$$

Architectural and fault tolerant features such as error handling mechanism, time or hardware redundancy determine the reliability of the system as well as the reliability of its components.

In a similar manner, the NoC reliability is a function of the reliability of its components, i.e. the reliability of the switches and the reliability of the links. The network topology defines the composition of the components in the network and thus influences the reliability. Fault tolerant features [Constantinides06, Ejlali07, Dalirsani07, Yu11] such as the routing algorithm to bypass defective components, or introducing extra redundancies, as well as the system application as the traffic distributor [Refan08, Bhardwaj12] influence the NoC reliability as well.

In an NoC-based system, if the switch is the only connection of a core to the system, a failing switch will disjoint one of the system resources. This may severely damage the system reliability, and sometimes the system may not be functional any longer. Therefore, in a simple reliability model, the NoC can be thought as a system comprising switches in series, in which the NoC fails if one switch fails. In this scenario, the reliability of the NoC, $R_{\mathrm{NoC}}(t)$, consisting of N switches is defined as a function of the failure rate of each switch, $h(t)$, as follows:

$$R_{\mathrm{NoC}}(t) = e^{-tNh(t)}\,. \tag{2.7}$$

Let us consider the case where the network is assumed functional as long as it can deliver a target performance level even if some switches fail. Furthermore, let us assume that in order to achieve a certain performance level in a network of size N, at least $N - k$ switches must be functional. In this case, the NoC reliability can be defined as the probability of which at time t, k switches or less fail. For $h(t)$ being the failure rate of the switch, the probability of having a functional switch at time t is

$e^{-t\,h(t)}$ and the probability of having a non-functional switch is equal to $1 - e^{-t\,h(t)}$. As a consequence, and inspired by [Lala01b], the NoC reliability can be computed as:

$$R_{\mathrm{NoC}}(t) = \sum_{i=0}^{k} \binom{N}{i} (1 - e^{-t\,h(t)})^i \, e^{-t\,h(t)\,(N-i)}\,, \tag{2.8}$$

given that the switches fail independently.

2.4.2 Performance

Throughput and latency are traditional measures of performance. In computer systems, throughput is defined as the total amount of the work completed in a given time [Hennessy12] and latency refers to the time between the start and the completion of a single task [Hennessy12]. In other words, the average time from the moment a task is requested until the moment in which the response is ready is called latency.

Throughput and latency of the NoC are defined similar to the computer networks where the transported message is considered as the unit of work [Tanenbaum03]. *Throughput* of the NoC is the average number of messages that are delivered to the connected cores over a given time period. On the other hand, *latency* is the average time that a message requires to be transported from the source to the destination [Dally03e]. Since in NoCs, messages are transported in the form of packets, we abstract latency as the average time that a packet requires to be transferred from the source to the destination.

Throughput is highly related to the level of concurrency that the system provides [Hennessy12]. In NoCs, architectural features such as the network topology, routing algorithm, and traffic distribution, as well as the running application play the main role to achieve a certain level of throughput [Dally03e]. On the other hand, latency of the network is determined by the components' delay (delay of the switches and links), the number of components a packet traverses from the source to the destination (hop count), and the time a packet must wait until it is served by a certain switch, named waiting time. Waiting time is in close relation with the packet injection rate, architectural features and application constraints.

In NoCs, latency and throughput are usually measured for various packet injection rates. The packet injection rate, known as *network load*, is defined as the number of injected packets per core per cycle. Higher throughput can be achieved by increasing the network load. As long as the network resources are sufficient for the injected packets so that the congestion is avoided, waiting time remains unchanged and therefore the throughput increases consistently.

At a certain point of time, because the number of packets in the network is too high, the waiting time starts increasing. The reason is that multiple packets are requesting the same paths. They are queued until enough resources become available in the network. As a consequence, the throughput diminishes and the latency increases. More drastically, the network may be enforced to drop several packets to avoid congestion and deadlock. In this case, several retransmission requests are triggered to compensate the packet loss. Eventually, the network is saturated. From this point forward, increasing the load does not bring higher throughput, conversely the overhead of handling more injected packets and several retransmission requests deteriorates the throughput.

2.5 Boolean Satisfiability

The Boolean Satisfiability (SAT) problem is a decision problem that determines whether a given propositional formula, called *SAT instance*, can be evaluated to true. A SAT instance is called *satisfiable* when there exists a variable assignment (called *satisfying assignment*) under which the respective propositional formula is evaluated to true. The SAT instance is called *unsatisfiable* if such an assignment does not exist.

The SAT problem is an NP-complete problem [Schaefer78]. Nevertheless, in practice advanced search-based algorithms and new solver technologies [Biere09] have paved the way to utilize the Boolean satisfiability for solving various problems in the Electronic Design Automation (EDA) domain in recent years [Chen96, M-Silva00, Smith05, Jo12]. Efficient algorithms [M-Silva99, Zhang97, Moskewicz01] of modern SAT solvers are capable of solving much larger instances of different applications, from circuit verification [Prasad05] [Biere09] to test pattern generation [Larrabee92, Eggersglüß13, Chen13].

In many propositional decision procedures like SAT problems, the formulas are required to have a particular structure. A variable or its negation is called a *literal* and a disjunction (sum) over a set of literals is called a *clause*. In SAT problems, the formula is represented in *Conjunctive Normal Form (CNF)* that is a conjunction (product) over a set of clauses [Clarke01]. A CNF formula is evaluated to true (satisfiable), when all of its clauses are evaluated to true.

Tseitin transformation [Tseitin68] is one of the first algorithms to convert an arbitrary propositional formula into CNF. For the SAT problems of the EDA application domain, a logic circuit needs to be represented in CNF. Toward this end, the Boolean formula of each circuit output must be extracted first, as presented in [Larrabee92]. The CNF representation of the circuit would be then a conjunction over the respective CNF formulas of all circuit outputs. Table 2.2 depicts the CNF representation for the Boolean formulas of two simple logic gates, the two-input AND gate and the inverter.

Table 2.2: CNF formulas for two basic logic gates

Gate	Input variables	Output variable	Boolean formula	CNF formula
AND	a, b	z	$z = a \wedge b$	$(\bar{z} \vee a) \wedge (\bar{z} \vee b) \wedge (z \vee \bar{a} \vee \bar{b})$
NOT	a	z	$z = \overline{a}$	$(z \vee a) \wedge (\bar{z} \vee \bar{a})$

Definition 2.5.1. *(CNF of a logic gate)* The term $\mathrm{CNF}(g, I_g, O_g)$ refers to the CNF representation of a logic gate g, where I_g and O_g are two sets of literals. I_g represents the literals corresponding to the input signals of g and O_g represents its output signals. For example, the CNF formula for a two-input AND gate with the input literals a and b and the output literal z is represented as $\mathrm{CNF}(\mathrm{AND}, \{a, b\}, \{z\})$.

Definition 2.5.2. *(CNF of a logic circuit)* Let C be a logic circuit and G be the set of logic gates in C. For each $g \in G$, I_g and O_g refer to the sets of literals for the input and output signals of g, respectively. As the Boolean formula of each gate in the circuit must be evaluated to true, the characteristic formula of the circuit in CNF is the conjunction of the CNF formulas of all gates in the circuit:

$$\mathrm{CNF}(C) = \bigwedge_{g \in G} \mathrm{CNF}(g, I_g, O_g)\,. \tag{2.9}$$

3

State-of-the-Art

The first requirement for graceful degradation of defective Network-on-Chips (NoCs) is a fault/error detection mechanism to locate faulty components. In addition, fault tolerant features must be integrated to discard defective components and reconfigure the network such that the cores can still remain connected through alternative routing paths made by fault-free components [Rusu08, Kang10].

Surveys of studies [Cota11, Radetzki12] have been devoted to fault detection and fault tolerance in NoCs. The purpose of this chapter is to review fault tolerant features that support switch-accurate and port-accurate network diagnosis. Moreover, this chapter explores state-of-the-art techniques for the granular fault detection in NoCs and emphasizes the gap that is bridged by the present work.

3.1 Fault tolerant Features in NoCs

Although NoCs can still be served as the data transfer medium in the presence of defective components, in some applications even a marginal performance loss is not acceptable. Thus, to retain the performance, spare components are integrated into the NoC in order to replace defective parts such as switches [Constantinides06, Chang11], controlling parts of the switches [Yanamandra10, Zhang08a, Kakoee11b], defective links between switches [Kakoee11b], or defective wires of a link [Lehtonen10]. Spare

components can be used for fault detection as well. However, in order to identify the defective component, three similar modules are required, which analyse identical input data. This is analogous to the concept of Triple Modular Redundancy (TMR) in which three identical components are engaged. In case of an error, a majority voter identifies the defective component to be turned off and the system is reconfigured to operate with two of its healthy components while preserving the same performance level, albeit with degraded reliability.

By means of redundancy, highly reliable NoCs can be constructed that may even preserve the performance in the presence of faults. However, redundancy is not an efficient solution for cost-sensitive designs. In NoCs, there exists an abundant number of switches and using spare switches is an obsolete fault tolerant alternative. Therefore, most of the fault tolerant solutions for NoCs eliminate defective components and use the inherent redundancy of the NoC to provide the required pear to pear communication among the cores. This however necessitates some prerequisites. Firstly, the faulty component has to be isolated in order not to interfere with the operation of other components [Lin09, Zhang10]. Secondly, an alternative routing path must be selected in case the default routing path is defective.

A defective link becomes abandoned as its two connected endpoint switches are aware of its faulty status, thus do not use it for data transfer any more. A defective switch port can be abandoned in a similar manner. Furthermore, a defective switch can be avoided by its neighbours when they are aware it is faulty. The defective switch itself is also shut down in order not to generate spurious traffic. After a defective component is turned off, the routing paths must be selected in a way that the defective components are avoided. Choosing alternative routing paths has been thoroughly studied under the subject of fault tolerant routing algorithms which is reviewed briefly in the next section.

3.1.1 Fault Tolerant Routing Algorithms

Despite their simple implementation, deterministic routing algorithms [Dally03b] such as XY routing are quite incapable of dealing with faulty components. Adaptive routing algorithms [Dally03c], however, bring the possibility to make an alternative decision

in case a faulty component appears in the path [Duato03b, Agarwal09].

In order to develop a fault tolerant routing algorithm, the advantages of both deterministic and adaptive routings are utilized [Hu04]. Most of the algorithms work based on a simple deterministic routing. In case of a fault, an adaptive algorithm undertakes the routing and adjusts the routing path according to the fault location. This policy prevents any overhead arising from the algorithm complexity in non-faulty cases.

When developing a fault tolerant routing algorithm, deadlock and livelock [Ni93, Dally03d] must be prevented by making relevant rules and restrictions. A wide range of fault tolerant routings for mesh topology exploit the turn model policy [Glass92, Chiu00] to avoid deadlock [Hu04, Chaix10, Fick09a, Zhang08b, Fukushima09, Ren14]. In the turn model approach, two out of eight possible turns in the switch are prohibited in order to prevent loops of packets waiting for the network resources.

In [Zhang08b], only the components around the faulty one perform the fault tolerant routing. In contrast, the algorithm in [Fick09a] broadcasts the information about faulty components to all network components and enables a global reconfiguration. The former, skips the overhead of broadcasting the faulty situation, however, it may spoil the performance, because without any information about the network status, traffic distribution cannot be managed appropriately and congestion may occur thereof.

Topics such as fault tolerant routing algorithms, deadlock and livelock avoidance, balanced traffic distribution to prevent congestion, and preserving performance in the faulty cases have been widely studied in the last decades [Ni93, Agarwal09]. With respect to the application, the probability of having faulty components in the network as well as power and latency constraints, the appropriate routing algorithm can be selected.

3.2 NoC Reliability

Fault tolerant features of the NoC, such as the routing algorithm to bypass defective components, or introducing extra redundancies, influence the reliability of the NoC

[Constantinides06, Ejlali07, Dalirsani07, Yu11]. Furthermore, the system application, as the traffic distributor, has an impact on the NoC reliability [Refan08, Bhardwaj12].

Constantinides et al. [Constantinides06] use a simulation-based analysis to compute the reliability of an NoC-based system that incorporates a self-repair switch architecture. The reference [Ejlali07] introduces the performability of on-chip interconnects as the probability to transmit L useful bits during a time interval T in the presence of noise. In their work, performability is computed as a function of the flit error rate and the operational frequency. Moreover, the flit error rate depends on the bit error rate, which is defined according to the Gaussian noise model. This performability parameter is used to compare error control schemes such as Automatic Repeat Request (ARQ) [Bertozzi05], Forward Error Control (FEC) [Worm05], and the hybrid ARQ/FEC schemes.

In [Dalirsani07], the reliability factor is defined as the probability that a fault is masked by utilizing internal architectural redundancies of the NoC without any effect on the system functionality. Reference [Refan08] computes the reliability of the NoC for three multimedia applications. The reliability of each system is defined as a function of the reliability of the switches in the available communication paths among processing elements. The reliability is recomputed while using spare links for processing elements. Yu et al. [Yu11] equip the routers with a self-correcting round-robin architecture and compute the reduction in the rate of which the routers may fail. Reference [Bhardwaj12] presents a comprehensive reliability analysis for the NoC mesh architectures with respect to the number of operational components (switches and links) at a given time. A traffic threshold for each component determines whether it is operational or not. This study demonstrates the impact of both fault tolerant routing and congestion-aware traffic distribution on the reliability of NoCs.

3.3 Test and Diagnosis of NoCs

In order to take advantage of the NoC as a degradable system, it is not sufficient to identify whether the NoC operates properly or not. Instead, fault detection schemes for network diagnosis are developed based on structural or functional testing of indi-

vidual components such that at the end of the testing process, defective components are identified. Finding defective components in the NoC is referred as the NoC diagnosis in the literature [Amory05a, Karimi08, Lin09, Ghofrani12]. However, the term must not be mixed up with the structural logic diagnosis [Wang06c, Huisman04] where the test fail data is analysed to locate defects within the random logic of a faulty component.

During the last decade, researchers have studied several test methodologies for NoCs considering area, speed and energy trade-off. In the rest of this chapter, state-of-the-art techniques for detecting faulty switches and links are discussed.

3.3.1 Structural Testing

To be compatible with the Automatic Test Equipments (ATE), structural testing methods for NoC switches use the standard scan testing infrastructure as the Design for Testability (DfT) technique [Amory05a, Hosseinabady06, Hosseinabady07, Grecu07b, Strano11]. The stuck-at fault model is most commonly used to describe defects in the switches. The crosstalk and interconnect shorts describe defects in the wires of the interconnect links [Grecu06b, Frantz07, Cota08].

To test the NoC entirely, most of the techniques conduct testing of identical switches in parallel to reduce the test application time. Amory et al. [Amory05a] equip the NoC, as a flat core, with the IEEE Std. 1500 compliant wrapper [Marinissen02] to enhance the NoC test access. The test structure applies test data to all switches in parallel and test responses of the switches are compared against each other by means of integrated comparators. Once a fault is recognized, switches are retested one by one in order to locate the defective one. To further reduce the test application time, methods in [Hosseinabady07, Strano11] take advantage of intra-switch regularity (i.e. identical port structures) to apply identical test vectors to multiple switch ports in parallel and locally compare the test responses. In [Hosseinabady06, Hosseinabady07] a test access switch broadcasts test data to all switches via the minimum broadcast tree.

Grecu et al. [Grecu05, Grecu07b] present a progressive reuse of already tested NoC switches to transport test data to the switch under test. Test patterns for the router

logic block and FIFOs (First In First Out) are generated separately and are optimized for each block. To restrict the I/O overhead for the test access, one network interface is dedicated for the ATE access [Grecu05].

Tran et al. [Tran06b, Tran08] develop a non-scan DfT architecture, consisting of a test wrapper, for testing switches and links. The method is dedicated to a special switch architecture with source routing. Test patterns in the packet format are generated by a custom program which is aware of both the switch architecture and the controlling bits of the packets.

Built-In Self-Test

By developing the Built-In Self-Test (BIST) architecture, general purpose Design for Testability (DfT) structures can be reused for in-field structural testing. BIST integrates the so-called Test Pattern Generator (TPG) and Output Response Analyser (ORA) into the chip [Wang06b]. Despite noticeable hardware overhead of the BIST circuitry, the use of BIST is justified because of several advantages for manufacturing tests as well as in-field testing. Self-test designs eliminate the need for expensive external test equipments. Moreover, as the test frequency is not dominated by the external test source or I/O pins, the test can be conducted more efficiently. Furthermore, BIST simplifies test and diagnosis at different steps during integration, which in turn will improve the final production yield [Wunderlich98, Hetherington99].

BIST of NoCs can be performed either at start-up or on-demand. Instead of only a go/no-go BIST, pinpointing the defective component is also of interest [Petersén07]. The regularity of the NoC architecture, which incorporates several identical components, is well utilized to reduce the BIST hardware overhead [Lin09, Zhang10, Grecu05, Petersén07, Grecu06b].

The methods of [Grecu05, Zhang10] offer a distributed BIST with a shared test pattern generator and test controller for all identical switches. BIST for testing the inter-switch links targeting crosstalk has been presented in [Grecu06b]. A global test controller injects test patterns while the test fail detectors are implemented at both endpoints of the links to analyse test responses. In addition, reference [Zhang10] im-

plements one FSM (Finite State Machine) for each switch as the BIST controller and conducts testing of communication channels, including switch ports and interconnect links.

Cota et al. [Cota08, Concatto09] place the test pattern generator and output response analyser in network interfaces to tackle interconnect faults in the switch-to-switch and switch-to-core links. To reduce the test time, test data (walking one sequence) is placed in the packet payload and the head flit directs the payload to to be applied the link under test.

3.3.2 Functional Testing

Functional testing targets certain functionalities of a system, for instance the instructions of a microprocessor [Brahme84]. As it is impossible to test the functionality completely, e.g. all possible arguments of an operation or all possible instruction orders, it is hard to estimate the final quality of such a functional test and the structural fault coverage is rather limited [Maxwell00]. Testing the NoC switch based on its functionality has attracted a lot of attention, because in case of a promising structural fault coverage, not only the test application time will be reduced, but also at-speed testing becomes possible [Maxwell00].

In general, the NoC switch can forward data from every input port to any other output port except that some paths are prohibited by the controlling logic. Therefore, the switch functionality can be generalized as a set of data transmission paths between pairs of input and output ports. For example, in a five port switch of the 2D mesh in which a 180 degree turn is prohibited, 20 distinct paths can be defined, which connect each input port to any other output port. Figure 3.1 presents four data transmission paths from the northern input port to the eastern, IP core, southern, and western output ports.

Many functional test approaches have been established on such functional switch model. Conventionally, functional test approaches conduct several test sessions such that undistorted data delivery over all transmission paths is ensured during the test. Stewart et al. [Stewart06] generate test patterns to trace all paths between the input

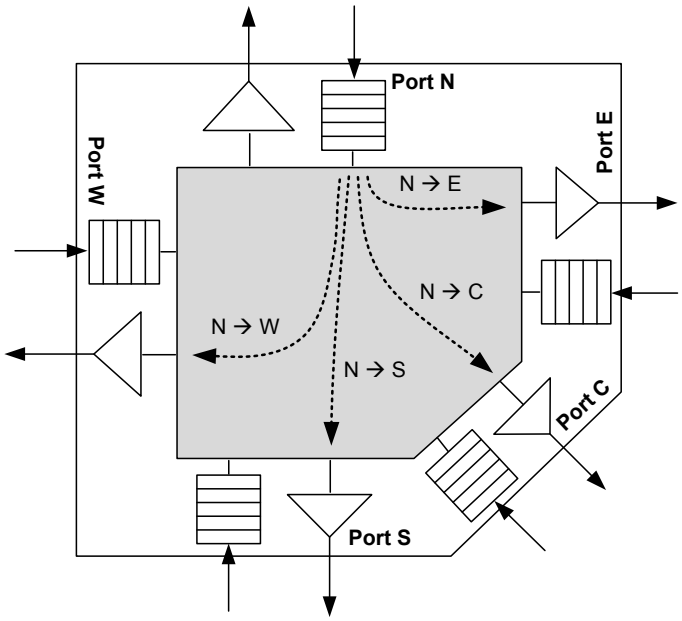


Figure 3.1: Data transmission paths from the northern input port of a switch

and the output ports. Raik et al. [Raik06, Raik07, Raik09] present an external functional test method for 2D mesh NoCs and a BIST circuitry to support in-field functional test [Raik12]. Targeting single stuck-at faults in the multiplexers and registers of the switch datapath, three test configurations including (a) testing straight paths, (b) testing direction changes, and (c) testing core connections are applied to trace the correctness of all values on the selector lines of the multiplexers. Each test configuration targets several switches in a test path. By assigning a value to one input that differs from the assigned values to the other inputs, faults on the data lines of the crossbar multiplexers are detected. In [Raik07, Raik09], once a test configuration indicates a faulty path, a diagnostic procedure is performed to find the location of the faulty switch. In this method, the router must support the deterministic XY routing. The functional test configuration delivers a high fault coverage for the targeted stuck-at faults in the multiplexers and registers of the switch datapath, while the rest of the stuck-at faults still remain untested.

In [Zheng10] with the assumption of having access to all ports of the boundary switches, several parallel straight and U-shape test paths are established to test the

switch functionality. This work, however, does not offer a method for testing the core connections. To completely test the datapath of the switch, a BIST method in [Kakoee11a] fills test patterns with all-zero and all-one sequences.

In order to detect structural faults in the controlling logic of the switch, some methods target more complex functionalities of the switch and generate test patterns so that the functional failures (also called *functional faults*) are detected [Kakoee11a]. The concept of functional faults has been used to develop online error detection mechanisms as well [Frantz06a, Karimi08]. Ghofrani et al. [Ghofrani12] introduce dropped flits/packets, spurious flits/packets, and misrouted packets as control faults and develop a fault detection mechanism using error syndrome collection and packet/flit counting. In [Qiaoyan11], two functional faults, packet loss and misrouting, are considered to develop a transient error detection/correction mechanism by means of retransmission. In [Abdel-Khalek12], more complex erroneous syndromes such as livelock, saturation and deadlock are investigated by processing the local log data at every processing element. As the method needs a comprehensive process of a huge amount of data, it can only be used for post-silicon validation.

Although test development based on the switch functionality seems to detect many defects in the basic switch elements such as crossbar multiplexers, datapath registers, and the simple routing, it is hard to estimate the quality of the functional test schemes with respect to the structural fault coverage. Reference [Aisopos11] searches for such correspondence by introducing a framework for mapping timing violations at the circuit-level to some system-level faults. However, looking into literature reveals that none of the functional testing methods target directly structural faults during the test pattern generation. This gap is filled by a structural Software-Based Self-Test approach [Dalirsani14a] which is discussed in detail in Chapter 5 of this dissertation.

3.3.3 Concurrent Error Detection

Some systems, like those for safety-critical applications, require higher levels of robustness. In such systems, reliability and data integrity are of high importance, but may be compromised by latent defects, wear-out or transient faults such as crosstalk, power supply noise or radiation effects [Borkar05, Nassif01, Baumann05]. Therefore,

Concurrent Error Detection (CED) techniques are widely used to detect errors that appear during the circuit operation [Goessel08].

Conventional CED Techniques

To implement CED with less area overhead than duplication while still detecting all single bit errors, Error Detecting Codes (EDC) such as parity are widely used. The principle is to distribute the outputs among one or more parity groups. A parity code (parity tree) computes the parity over the output bits of each group. A single bit parity can detect only errors of odd multiplicity. A fault in the circuit may propagate to an arbitrary number of outputs so that the error is not detectable by the parity bit. To ensure all erroneous output patterns are detectable by the parity code (i.e. fault-secureness, see definition in section 1.3.2), logic sharing between any two outputs in the same parity group must be avoided.

One solution is to use a single parity bit and prohibit logic sharing among the outputs by replicating the shared logic [Touba93, De94]. This solution imposes a huge area overhead. Therefore, many schemes distribute the outputs among multiple parity groups [Touba97, Bolchini97, Almukhaizim03, Ghosh05] and the outputs of each group are encoded individually. In this scheme, the circuit outputs are partitioned into the groups such that no logic sharing is allowed within the logic block of each group, while sharing is maximized between the logic of different groups. Figure 3.2 illustrates the principle of CED using single and multiple parity bits. Using multiple parity bits, logic sharing between the parity groups is permitted. Logic replication is needed only to remove the logic sharing within the groups, which reduces the hardware overhead. However, this is traded off against the overhead of the parity checker and prediction logic, which increases as the number of parity groups increases.

Touba et al. [Touba97] enhance the efficiency of multiple parity groups by employing a cost function to trade-off, for an extra parity group, the additional cost necessary for the parity prediction and checker against the cost saving for logic sharing. A greedy algorithm uses the heuristic of pairwise combining parity groups in search for a minimal area overhead. The drawback of this method is that more redundancy than required might be introduced. Moreover, the greedy algorithm can easily get caught

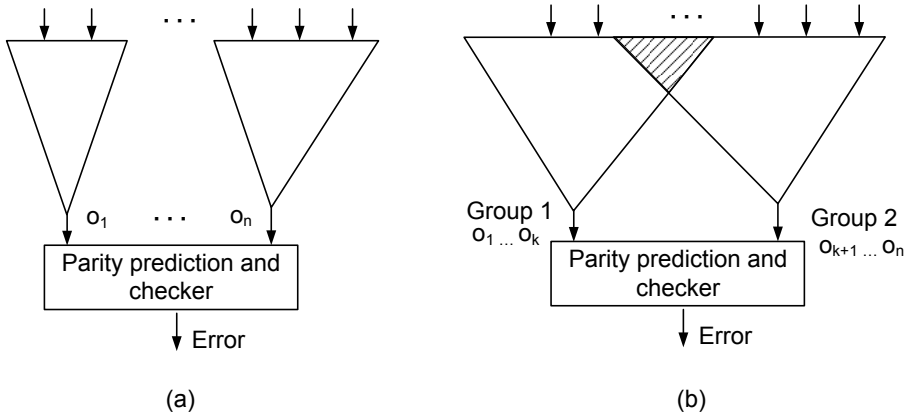


Figure 3.2: CED using (a) single parity bit: logic sharing is not allowed between the outputs. The shared logic is replicated, i.e. circuit is modified. and (b) multiple parity bits: logic sharing is not allowed within the same parity group, but is permitted between the groups.

in a bad local optima, since no look-ahead information is used during each merge. In [Bolchini97], the circuit is modified so that each fault is observable at an odd number of outputs for each input configuration. The modification strategy introduces either an auxiliary output or a node replication in order to achieve the required observability condition.

In [Almukhaizim06], the parity code search is guided by an entropy-based metric to find a low-power implementation of the CED structure. This technique requires the explicit computation of a large error detection table with entries for each fault and input pattern, which is infeasible for large circuits.

Mitra et al. [Mitra00] provide a comparison of the overhead of various CED schemes with respect to the protection against common-mode failures. Their investigation shows that duplication with comparison sometimes has marginally higher area overhead compared to parity-based schemes while providing higher protection against common-mode failures.

When the circuit structure and functionality is known, one may choose a function-inherent code checking method like [Metra08] which exploits internal redundancies of the circuit for CED. However, this is not applicable for arbitrary circuits.

To reduce the hardware overhead of CED, some techniques detect only as many faults as possible at a reasonable overhead. The technique in [Vemu08] targets only transient faults which have a high probability of causing an error in the architectural state of the processor. A partial duplication method targeting the nodes that have the highest transient error susceptibility has been introduced in [Mohanram03b]. The method reduces the rate of the circuit failure caused by the targeted transient errors with only a fraction of the overhead required for full duplication. The methodology presented in [Mohanram03a] is also effective for only the targeted faults with high sensitization probability. In [Richter09], a split-parity code is investigated that detects all faults producing odd errors with certainty and additional faults producing even errors with a probability of 50%. However, these techniques do not ensure the fault-secureness property, which means it is not guaranteed that for any input pattern, every error at the circuit outputs is detectable by the error detecting code.

CED of NoCs

Despite periodic testing of NoC components at start-up or on-demand, transient and permanent defects may still threaten the reliability of the NoC during the operation. To conduct CED in NoCs, hardware redundancy schemes (such as duplication with comparison, Triple Modular Redundancy (TMR)) have been examined [Zhang08a, Frantz06b, Yanamandra10]. Based on the hardware redundancy concept, a fault detection circuitry for routing computation and round-robin scheduler functions of the switch was implemented in [Qiaoyan11], which detects transient faults in the controlling logic of the switch.

Data encoding using error detecting codes in combination with data retransmission is typically used to detect and correct errors that appear during the NoC operation [Lehtonen10, Rossi07, Palesi11, Bertozzi05, Grecu07a]. Toward this end, check bits are computed and appended to data bits which are transported over the network. Upon an erroneous data reception or a data loss, the receiver sends a message, invoking the sender to retransmit the data. This technique imposes less hardware overhead as compared to the hardware redundancy schemes, but it may not detect all errors in the system.

The methods in [Kang10, Kohler10] rely on CRC (Cyclic Redundancy Check [Lala01c]) as the error detecting code to detect faults in datapath elements and communication links. The method in [Ghofrani12] uses an error correcting code in order to detect and correct data errors during transmission.

Authors of [Grecu06a] use a code disjoint design with the parity and examine incoming data to the switch. The structure indicates a fault in the incoming data link when the locally extracted parity bit and the transported parity bit mismatch. Furthermore, data is rechecked upon departure from the switch. In case of a mismatch, the data transmission path inside the switch is considered faulty. A Hamming code is used in [Frantz06b] to detect and correct crosstalk effects in the links as well as transient errors in the sequential elements of input buffers of the switch. Furthermore, by adding three sampling registers at the input ports along with a voter, it implements TMR (Triple Modular Redundancy) to increase the efficiency of the fault tolerant technique.

Several researchers have investigated error detecting/correcting codes in terms of silicon area, encoder/decoder delay, performance, and energy consumption to find a good trade-off between costs and reliability [Palesi11, Bertozzi05, Grecu07a]. Data encoding schemes reduce the area overhead of concurrent error detection as they consider special characteristic of NoCs (i.e. transporting pure data over datapath elements). However, up to now, these methods are mainly devoted to detect faults in the inter-switch links and intra-switch datapath elements such as multiplexers, while the rest of the faults remain undetected. On the other hand, methods based on hardware redundancy like duplication are able to detect many errors in the NoC switches. But, they do not consider special characteristics of NoCs and impose a huge area overhead. The advantages of data encoding, to reduce the overhead, and hardware redundancy, to detect any error resulting from a single fault in the NoC switches and links, are combined for the first time in a hybrid synthesis technique [Dalirsani14b], which is discussed in detail in Chapter 6 of this dissertation.

3.4 Granular Fault Detection

To take further advantage of degradable NoCs, fault detection mechanisms must be performed in a more granular way such that non-faulty wires of a link and non-faulty ports of a switch are used for communication.

Detecting defective wires of a link has been discussed in [Grecu07b, Lehtonen10, Vitkovskiy10] as *bit-level link diagnosis*. An interconnect fault model called Maximum Aggressor Fault (MAF) was presented in [Cuviello99] that accounts for crosstalk effects between a set of aggressor lines and a victim line. The MAF fault model assumes all aggressors are transitioned simultaneously in the same direction. It considers four faulty values on the victim line: positive glitch, negative glitch, slow to fall, slow to rise. As a consequence, for an N-line wide set of interconnects, the fault model introduces $4N$ faults. Inspired by MAF model, Grecu et al. [Grecu07b] employ two vector test sequences to sensitize the MAF on a victim line of a link. In each test vector, the value of the aggressor lines is opposite to the value of the victim line. Thus, to test each wire on the link, eight test vectors (respectively eight test cycles) are required. The test procedure periodically tests all wires of the link.

Lehtonen et al. [Lehtonen07b, Lehtonen07a] present a fault tolerant link structure. They use a Hamming code for fault detection and retransmit the erroneous packets due to transient faults. To tackle intermittent and permanent faults, spare wires are introduced. An In-Line Test (ILT) method is presented in [Lehtonen10] for link diagnosis. It periodically tests the pairs of the wires in a link by shifting the test pattern to the place of the targeted pair of wires. The ILT implementation tests the open and short circuit faults in the links.

The Partially Faulty Link Recovery Mechanism (PFLRM) [Vitkovskiy10, Vitkovskiy12] combines the fault detection, diagnosis and recovery of the links in the bit-level granularity. In this mechanism, test vectors contain two patterns of alternating zeros and ones with one-bit shift difference between the two. A bit-wise XOR of the link wires in the downstream switch identifies then the defective wires.

To reduce the bit-level link diagnosis effort, the work in [Chen12] divides every link into several sections of multi-bit width and performs the diagnosis at section-

level. Although this method reduces the diagnosis cost by moving from bit-level to section-level, once a section becomes faulty, all wires of that section become out of use. The proposed architecture implements a serialization/de-serialization pair for each unidirectional link. Adjacent flits are serialized and merged to fit the narrowed faulty links.

The method in [Shamshiri11] utilizes four interleaved error locality aware codes for online error detection/correction in 64 bit wide links of NoCs. Unlike the link diagnosis methods that rely on test vectors to exactly identify defective wires of a link, this method presents a scoreboard to accumulate how often a wire in the link has been suspected for the defect. When an error occurs along a path, this number is then used to locate defective wires (i.e. the ones with the higher number in the scoreboard).

The method in [Palesi10] utilizes partially faulty links and quantitatively shows the performance improvement compared to the case where a faulty link is discarded. Similar to partially faulty links that bring performance advantages, utilizing intact functions of the faulty switches is also of interest. Some researchers have employed partially-faulty switches in the degraded NoCs [Fick09b, Parikh13], however searching the exact cause of the faulty behaviour has been neglected.

A Built-In Self-Test and Diagnosis (BIST/BISD) circuit to test and locate faulty FIFOs and multiplexers of the switches in 2D mesh NoCs has been developed in [Lin09]. All-0 and all-1 test vectors are applied to each of the 20 data transmission paths (explained earlier in section 3.3.2) one after the other while targeting stuck-at faults. By analysing the fail test responses, the FIFO of an input port or the MUX of an output port is recognized as the faulty part.

In [Ghofrani12], the authors utilize end-to-end error correcting codes to deal with data faults. To handle control faults three syndromes named dropped or spurious flits, dropped or spurious packets, and misrouted packets are defined. The syndromes are detected by counting the number of sent/received flits. A counter at each switch increments upon each packet arrival and decrements upon each packet departure. For a deterministic routing, all misroutes can be detected using a lookup table or a simple hardware assertion. A scoreboard computes and accumulates the fault probabilities of different components in the path from a source to a destination. The component (link

or routing path) with a higher fault probability is intended to be defective. The method requires sufficient erroneous traffic to achieve acceptable diagnosis resolution. It does not target structural faults and because the scoreboard works based on probabilities, there is no confidence that the available network components are not faulty. Furthermore, detection of a permanent fault may postpone while the network keeps injecting erroneous traffic into the system.

Inspired by the fault detection mechanism using scoreboarding which was introduced in [Shamshiri11, Ghofrani12], Parikh and Bertacco [Parikh13] propose the 'uDirect' architecture which enables the use of partially faulty switches in the network. They assume that all link and switch-level faults are mapped to uni-directional links. Based on this fault model, a software-based reconfiguration scheme and a fault-tolerant routing algorithm is developed. The experiments show an improvement in performance and connectivity of the faulty networks while using partially-faulty switches.

Strano et al. [Strano11] divide the switch into several identical sub-blocks and conduct parallel testing of sub-blocks. One comparator at each switch checks the test responses and can identify the defective sub-block. One disadvantage of this method is that all sub-blocks of the switch need to provide test access interface for testing. Moreover, to avoid interference of a defective sub-block, it needs to be isolated by means of the wrapper cells, which imposes extra overhead. Furthermore, as the NoC switch is a relatively small component, it is usually designed as a unique component and dividing it into several sub-blocks for test and diagnosis purposes is not feasible.

Fick et al. [Fick09b] present an NoC architecture named 'Vicis', that can tolerate the loss of defective switch ports. By means of a BIST procedure that tests individual sub-blocks of the switch, including the routing table, crossbar controller and FIFO controller, the method identifies the defective sub-block of the switch and respectively reconfigures it such that the defective components are discarded. The rest of non-faulty switch components are then used for further communication. The test and diagnosis method in [Fick09b], which is based on testing the sub-blocks in the switch, suffers from the same disadvantages as the method in [Strano11]. However, the reconfigurable architecture enables the use of partially-faulty switches in NoCs. It uses port swapping to mitigate the penalties of having defective ports in the neighbouring switches. Moreover, a crossbar-bypass technique eliminates defective ports. This work shows that an

NoC with Vicis reconfigurable architecture can continue its operation even when approximately half of its switches are faulty. In fact, Vicis provides the reconfigurable platform to utilize partially-faulty switches in the network.

Looking into the literature clearly reveals that finding the exact cause of a faulty behaviour in the switch and respectively reasoning about the faulty and non-faulty switch functions has been neglected in the previous studies. In fact, none of the previous approaches have used the standard test flow and fine-grained structural diagnosis to identify the intact functions. The first method that reasons about the faulty and non-faulty switch ports and thus ensures the correct functionality of the non-faulty ports was presented in [Dalirsani11, Dalirsani12] and is discussed in detail in Chapter 4 of this dissertation.

3.5 Summary

Fault tolerant features make NoCs usable even in the presence of faulty components. Test and diagnosis approaches determine the location of faulty components in the network, rather than providing only a go/no-go response.

Structural tests of NoC switches provide high structural fault coverage, while functional tests can fully utilize the NoC functionality for test, but compromise the fault coverage. The structural Software-Based Self-Test approach in Chapter 5 utilizes on-chip resources to test structural faults in the NoC and thus contains the advantages of functional tests without the loss of structural fault coverage.

To make the NoC more robust, concurrent error detection (CED) schemes have been developed. State-of-the-art CED schemes for NoCs either utilize data encoding or the hardware redundancy. The former can cover only a subset of faults while the latter imposes a huge area overhead. Chapter 6 develops a hybrid synthesis technique which reduces the hardware overhead as compared with the hardware redundancy schemes without the loss of fault coverage.

For more granular graceful degradation, intact functions of defective switches can be employed in the network. Existing techniques for identification of the faulty parts of

the switch rely either on the functional testing or on the switch sub-blocks. Although functional testing may deliver some confidence about the correct functionality of the switch, it cannot be proved which functions are intact and which ones are affected by the fault. On the other hand, dividing a switch into sub-blocks imposes unnecessary overhead for test and diagnosis. The novel technique presented in the following chapter uses structural test and diagnosis data to identify and retain the fault-free switch ports, enabling port-accurate diagnosis.

4

Structural Diagnosis of NoC Switches

In this chapter, the novel port-accurate diagnosis method is presented. The method identifies the ports of a faulty switch that must be discarded in order to mask all fault effects. By mapping the structural defect information to the NoC switch functions, detailed information for fault tolerant routing and reconfiguration is provided.

4.1 Overview and Preliminaries

Network-on-Chips (NoCs) are implicitly fault tolerant and can overcome defective links and switches due to their inherent redundancy. The same capabilities which are needed to isolate a defective link from the network, can be reused to support individual port deactivation (see section 3.1). As a defective link can be represented by two defective switch ports at its two endpoints, faults in the interconnect links are easily mapped to individual switch ports. However, finding a mapping between structural faults inside the switch and a defective switch port is not straightforward. Such mapping allows to disable defective ports of a switch after test and utilize the intact functions. To achieve this, on the one side, the precision of structural testing is required to localize the fault inside the switch, i.e. *structural logic diagnosis*. On the other side, the fault impact on the functional behaviour of the switch must be precisely analysed to reason about the intact functions, i.e. *functional mapping*. Finally, the damaged functions should be

mapped to individual switch ports to be used for *port-accurate reconfiguration*.

The main challenge here is to identify the remaining capabilities of a defective switch from structural test data, while it must be guaranteed that deactivating certain ports indeed confines all erroneous effects of the fault.

4.1.1 Overall Flow

The overall flow of our approach for structural diagnosis of NoC switches is depicted in Fig. 4.1. Firstly, using structural test data, the logic diagnosis reports that the test responses of a specific switch in the NoC can be explained by a single fault f. Secondly, the functional mapping identifies the switch functionalities and the ports that are disrupted under the fault f. Lastly, the ports to be avoided are looked up and the optimal switch reconfiguration is applied. The main three steps of our diagnosis approach (grey blocks of Fig. 4.1) are as follows:

1. Logic Diagnosis: The structural test data is analysed by an efficient logic diagnosis algorithm (explained in section 4.2) to identify the defect location within the random logic of the switch. During the manufacturing, logic diagnosis is accomplished for every chip which does not pass the production test. Furthermore, after production, any structural test with high structural fault coverage can be succeeded by a logic diagnosis for repair and reconfiguration purposes [Elm10]. The system may include an embedded diagnosis core for online repair and reconfiguration. Since an embedded diagnosis is a costly enhancement, logic diagnosis can be conducted at repair centres or workshops for customer support services.

2. Functional Mapping: For each possible function in the switch, it is determined if it may be disrupted by the identified defect (described in section 4.3). Functional mapping is performed once for every designed NoC switch. Inspired by the worst case condition of the Conditional Line Flip (CLF) model (i.e. the condition is always true), functional mapping is performed once independent of the fault type. The corrupted switch functions corresponding to every defect location are

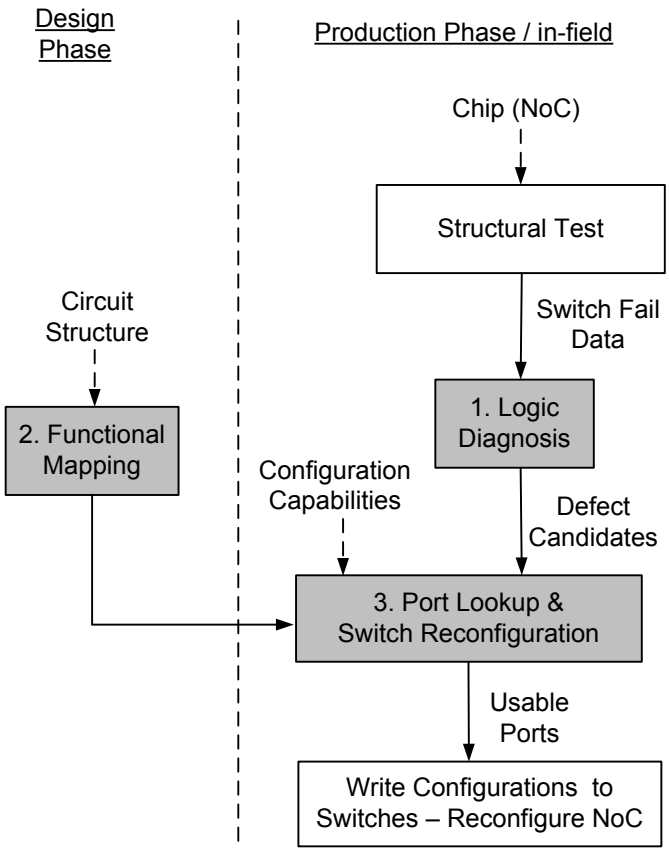


Figure 4.1: Overall flow

extracted. This mapping information is stored in a dictionary for fast lookup after logic diagnosis.

3. Port Lookup and Switch Reconfiguration: The defect candidate discovered by logic diagnosis is looked up in the dictionary and the affected functions and ports are picked up. An optimal set of ports associated with the disrupted functions is selected to be disabled (details in section 4.4). Accordingly, the switch and the network are reconfigured. The new reconfiguration guarantees that all fault effects are masked.

Example 4.1.1. Figure 4.2 presents an example of using a partially faulty switch in the NoC. In this 2D mesh, the outcome of our method indicates that defects only influence the southern outgoing port of the switch $(1, 0)$ and the northern incoming port of the

switch $(1, 1)$, and therefore both have to be deactivated. Consequently, if core A sends a packet to B, the data will be routed via either switches $(2, 0)$, $(2, 1)$ or $(0, 0)$, $(0, 1)$ using an appropriate fault tolerant routing algorithm, for example the one described in [Kohler10].

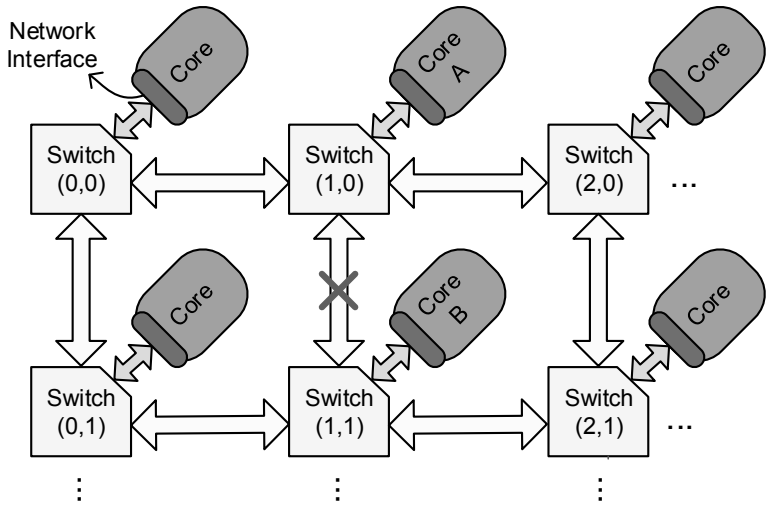


Figure 4.2: 2D mesh topology with a partially faulty switch

4.1.2 Requirements

The diagnosis approach here is not design-dependent and hence can be applied to arbitrary switches of any NoC topology. The only requirements are testability and reconfigurability. The structural test infrastructure is necessary for logic diagnosis. Moreover, the architecture must support individual deactivation of the switch ports for fine-grained graceful degradation.

Testability

Every switch of the network needs to be tested separately and a high structural fault coverage must be achieved in the test mode. As a consequence, each individual switch must be accessible by the ATE (Automatic Test Equipment) or via

Built-In Self-Test (BIST). Test data must be transported to the switch and test responses have to be propagated to the outside. Many efficient NoC test strategies [Amory05b, Grecu06b, Hosseinabady07, Raik07] already satisfy this requirement. In general, this condition must be fulfilled for a proper production test of an NoC.

Compared to the previous fault detection approaches for identifying switch portions to be deactivated [Lin09, Fick09b], the method here does not introduce any hardware overhead for test and diagnosis and utilizes the available test structures that have been already added for the production or in-field testing. Furthermore, it can utilize recent, powerful diagnosis techniques, which are able to track down defects to signals instead of defective units.

Reconfigurability

To efficiently utilize the test results for graceful degradation, a switch must allow individual deactivation of any incoming or outgoing port. This feature is already supported by some fault tolerant NoCs [Fick09b, Parikh13]. Even designs targeting only deactivation of complete switches [Zhang08b, Lin09] need to support the individual deactivation of ports to properly isolate a defective switch. In these designs, all neighbours of a defective switch are configured such that the outgoing port connected to the defective switch is deactivated. Moreover, all requests coming from an incoming port that is connected to a defective switch, have to be ignored.

Toward this end, a single bit is dedicated to every incoming and outgoing port, representing the faulty/non-faulty status of that port. A faulty incoming port is deactivated by configuring the router to ignore any data and requests coming from the respective port. A faulty outgoing port is deactivated by redirecting packets destined for the respective port to an alternative port. Moreover, the NoC must implement a fault tolerant routing scheme like [Chiu00, Kohler10, Ali05, Zhang08b] to diverge the traffic over alternative routes toward correct destinations. Since the same capabilities are needed to isolate a defective switch from the network, the more fine-grained port deactivation involves no additional overhead.

The following sections discuss in detail the steps of our diagnosis approach includ-

ing logic diagnosis, functional mapping, and reconfiguration.

4.2 Structural Logic Diagnosis

If a switch does not pass the structural test, well-established diagnosis techniques can be used to find the defect location. The application at hand however has slightly higher demands on the diagnosis results than other applications as explained below.

Structural logic diagnosis [Huisman05, Holst09, Bartenstein01, Huisman04] uses the available test fail data to locate the defects within the random logic of faulty switches. For many diagnosis applications, it is sufficient to locate one fault site in order to direct the physical failure analysis or to provide data for volume diagnosis. Here, as we need to have evidence that the remaining switch structures are defect-free, locating a single fault site is not sufficient. Indeed, additional fault sites may be present in other parts of the switch which are not part of the diagnosis result.

Toward this end, the logic diagnosis algorithm has to classify the faulty behaviour of the circuit to be either *completely* explainable by the identified defects or not. If the faulty signature of the circuit cannot be completely explained, some unknown interactions among multiple fault sites are present and the switch has to be completely disabled. On the other hand, if the identified defects can explain the faulty signature completely, there is no indication that the remaining circuit structure is defective.

Studies have shown that most production defects disrupt just a single internal signal directly [Huisman04] and diagnosis algorithms based on the single location at-a-time (SLAT) paradigm [Bartenstein01] can locate such defects with high precision. However, due to various physical and electrical interactions, numerous defect mechanisms appear in real silicon that can no longer be explained by a single fault model. Therefore, fault-model-independent diagnosis approaches are used to determine, whether a faulty behaviour of a circuit can be attributed to a single defective signal or gate within the circuit.

In this regard, the Conditional Line Flip (CLF) fault model [Wunderlich10] has been proved to be quite effective. Inspired by the CLF fault model, the algorithm from

[Holst09] identifies all faults affecting a single site in the switch. By using a test set with sufficiently high coverage, it reports whether the test responses can or cannot be explained by a single conditional line flip.

Even with a test set of high fault coverage, the diagnosis algorithm might only be able to narrow down the set of fault candidates to more than one possibilities due to structural equivalency. This, however, does not influence the overall method, since we have observed that in all these cases the equivalent candidates affect the same functions of the switch. Thus, the functions to disable are precisely identified. In the worst case, the algorithm may pessimistically discard some intact functions. How to map defects to affected functions is explained in the next section.

4.3 Functional Mapping

This section describes the mapping between structural faults and switch functionalities, which is done once during the NoC development. The result is stored in a dictionary and is looked up after completion of the first step and prior to the third step of our diagnosis approach.

4.3.1 Structural Fault Model

Structural logic diagnosis introduces a single Conditional Line Flip (CLF)[1] on the victim signal v as the defect candidate, which encompasses both fault site and fault condition. A defect can disrupt the victim signal in different ways. However, the victim signal will carry a faulty value at least in some situations. In the worst case, the victim signal will *always* generate a faulty value, which can be modelled as *unconditional line flip* $v \oplus [1]$. Fault simulation of this worst case condition may generate more failing responses than those which are observed in the switch containing a defect $v \oplus [cond]$ with an arbitrary condition. Nevertheless, every observed fail will be perfectly

[1] A CLF $v \oplus [cond]$ describes a victim signal v, which carries a faulty value (i.e. the opposite of the fault-free value), if the condition $[cond]$ is true. Refer to section 2.2 for more information.

explained by simulation of $v \oplus [1]$.

Accordingly, unconditional line flip is used to describe structural faults during functional mapping, thus making it fault-model-independent. In this way, the remaining functionality will not rely on the behaviour of the fault, which may change over time. If all fault effects of a line flip $v \oplus [1]$ can be explained by some switch functions, it is concluded that any defect $v \oplus [cond]$ is also explained by those switch functions. As commercial ATPG tools cannot directly work with unconditional line flip, a similar conclusion can be made if some switch functions can explain all fault effects of both a stuck-at 0 ($v \oplus [v]$) and a stuck-at 1 ($v \oplus [\overline{v}]$) at v.

4.3.2 Switch Functionality

The functionality of the NoC switch can be abstracted as a data transfer from an incoming port to an outgoing port.

Definition 4.3.1. *(switch ports)* For an n-port switch, we define P to be the ordered set of the incoming ports of the switch, i.e. $P = (p_1, p_2, \ldots, p_n)$. In addition, P' is the ordered set of the outgoing ports of the switch, i.e. $P' = (p'_1, p'_2, \ldots, p'_n)$. Accordingly, each port of the switch is represented by a pair of an incoming and an outgoing port as follows:

$$port_j = (p_j, p'_j) \qquad 1 \leq j \leq n\,,\; p_j \in P\,,\; p'_j \in P'$$

Definition 4.3.2. *(transport path)* For an n-port switch with P and P' being the ordered sets of incoming and outgoing ports as given in definition 4.3.1, a transport path τ_{ij} is a route from the incoming port $p_i \in P$ to the outgoing port $p'_j \in P'$ when the data transfer from p_i to p'_j is possible, and it is denoted as follows:

$$\tau_{ij} : p_i \triangleright p'_j \qquad 1 \leq i, j \leq n\,,\; i \neq j$$

Here, it is assumed that the 180° turns are not allowed.

A switch can be represented in a combinational form. Figure 4.3 gives an example for such a representation of a switch of the 2D mesh topology. The depicted switch has five incoming ports $P = (N, S, W, E, C)$, five outgoing ports

$P' = (N', S', W', E', C')$, and additional inputs and outputs from and to its state elements. Besides, the control input assignments for forcing the switch into any of its transport paths $x \triangleright y$ are known. Since the data can be forwarded from every incoming port to every other outgoing port, such a switch has $5 \times 4 = 20$ distinct transport paths.

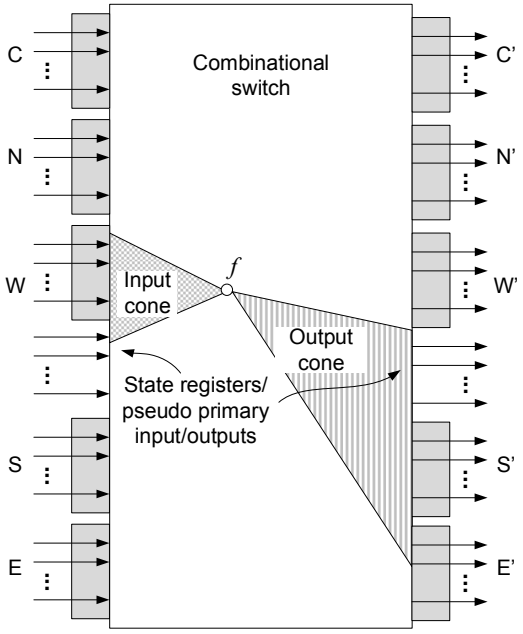


Figure 4.3: Combinational representation of the five port switch with a fault f

The mapping of every structural fault to the set of switch transport paths that it may disrupt, is performed in two steps. The first step is a topological preprocessing and the second is the functional reasoning.

4.3.3 Topological Preprocessing

A fault f can disrupt signals within the combinational circuit in different ways. Let v be the victim signal associated with f. If there is no topological path from the victim signal v, neither to the outgoing port $o \in P'$ nor to the router states, the fault f does not affect the transport paths $x \triangleright o$ ($\forall x, x \in P$). Thus, only the remaining paths, $x \triangleright o'$, for every $o' \neq o$ have to be analysed.

This simple observation provides a good approximation of unaffected transport paths for all faults close to the outgoing ports of the switch. However, most signals near the input ports have structural paths to many outputs and the router state. Hence, the topological preprocessing is pessimistic and a complementary analysis is needed to obtain a better approximation for unaffected switch functionalities.

4.3.4 Functional Reasoning

Functional reasoning exactly determines the switch transport paths and the outgoing ports that are affected by a fault. This is achieved by means of a constrained ATPG algorithm. The fault f is injected and the control inputs are constrained to enable a certain transport path under which the switch needs to be checked. In every check, only one outgoing port remains unmasked. Those outgoing ports which are not of interest for the current check are masked out by the ATPG constraints as well. This allows us to find a single port or a transport path which is affected by the fault and prevents pessimistic deactivation. If the ATPG is able to generate a test pattern activating and propagating the fault under the given constraints, it proves that the fault f in combination with the chosen path may affect the unmasked outgoing port.

Functional Implications of Structural Faults

A structural fault may disrupt the switch functionality in two ways: firstly, when a certain transport path is selected, the fault may propagate to the outgoing port of the path and hence the transported data is corrupted. Secondly, the fault may propagate to an idle outgoing port, generating spurious traffic. Depending on the selected path and the unmasked outgoing port, it is either sufficient to disable a certain transport path or the outgoing port has to be disabled. The distinction between these two cases is done by categorizing the checks into two classes: *output conditions* and *path conditions*. As a consequence, the complete analysis for every fault f may have the following implications for reconfiguration:

1. Path condition:

For each transport path $x \triangleright y$ not ruled out by the topological preprocessing, we mask all the outgoing ports $o \neq y$ and assign the control signals to select $x \triangleright y$. If the ATPG is able to propagate f to y under this constraint, $x \triangleright y$ must be avoided.

As an example, consider Fig. 4.4 and assume we want to check the transport path $C \triangleright N'$. The path is selected by assigning corresponding control signals. Then, all outputs except N' are masked. If the fault f propagates to the outgoing port N', it will disrupt the selected path $C \triangleright N'$, so it must be avoided.

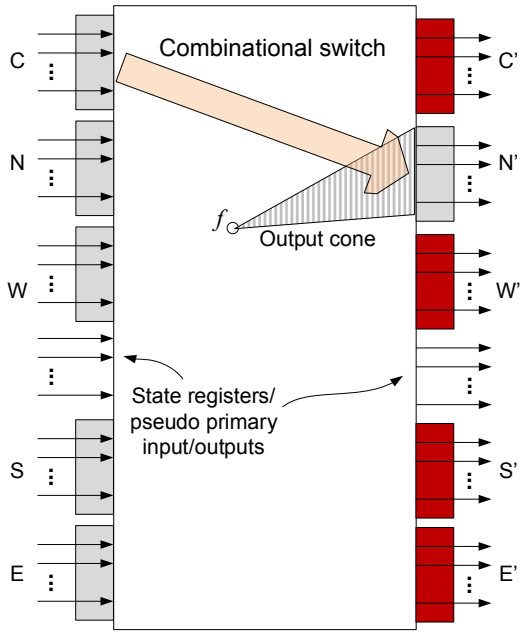


Figure 4.4: Path condition

2. Output condition:

   For each outgoing port o and each control signal assignment selecting a transport path $x \triangleright y$ for $y \neq o$, we mask all the ports $o' \neq o$. If the ATPG is successful to propagate the fault f to o, the outgoing port o may generate spurious traffic even if it is not selected by the control logic. Thus, it has to be discarded.

   Figure 4.5 gives an example to check the output condition for the outgoing port W', where all outgoing ports except W' are masked. To ensure that a fault f never generates spurious traffic on W' when it is not the outgoing port of any

selected transport path, the output condition must be checked for all transport paths with outgoing port $o \neq W'$. In Fig. 4.5, the output condition for W' is checked when the path $C \triangleright N'$ is selected.

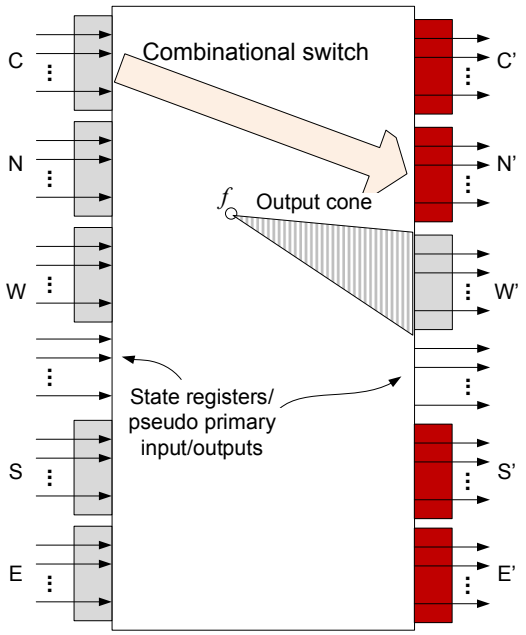


Figure 4.5: Output condition

3. Switch condition:

   The switch is disabled completely if:

   (a) logic diagnosis cannot explain the faulty behaviour by particular conditional line flips, or

   (b) the ATPG is able to propagate the error signal to the router states.

Since the switch is a rather small circuit, for any fault in the circuit, the ATPG should always be able to generate the test or to prove redundancy. However, if the ATPG fails due to a timeout and therefore aborts a fault, the corresponding paths or outputs have to be removed as well.

Completeness of Functional Implications

All fault effects of any testable fault in the switch not ruled out by the switch condition, can be fully masked by avoiding the transport paths and the outgoing ports which have been reported by the path and the output condition, respectively.

A fault effect may propagate to the state registers or to the switch outgoing ports. The former is the case when the switch must be deactivated under the switch condition. Let us now consider an outgoing port p'_j, not ruled out by the output condition and suppose we observe erroneous data on p'_j under a certain assignment of the inputs and control signals.

If the control assignment has selected the transport path $\tau_{ij} : p_i \triangleright p'_j$ under which the erroneous data is observed, there exists an input assignment that propagates the fault to p'_j when τ_{ij} is selected. That is, the fault is testable under the given constraints and the ATPG should have found the solution, i.e. the transport path is ruled out by the path condition and thus it should have been avoided.

If the control assignment has selected any transport path τ_{ik}, $k \neq j$, it means that there exists an input assignment that propagates the fault to the outgoing port p'_j when τ_{ij} is not selected. In other words, the fault is testable under the given constraints, i.e. the outgoing port p'_j must have been ruled out by the ATPG because of the output condition.

4.4 Port Lookup and Switch Reconfiguration

Functional mapping determines the transport paths and the ports which are damaged in the presence of a structural fault. However, for the port-accurate diagnosis, the information about the corrupted transport paths has to be mapped to the switch ports.

A defective transport path of a switch, ruled out by the path condition, can be avoided by discarding its corresponding incoming and outgoing ports. However, discarding a port disables some other paths. A structural fault may disrupt several transport paths of the switch at the same time. This will soon end up to deactivation of the

entire switch. Therefore, it is necessary to select a minimum number of discarded ports so that all defective paths are avoided.

Given P and P' as the sets of the incoming and outgoing ports of an n-port switch, consider the function $\gamma : P \times P' \rightarrow \{0, 1\}$ to be $\gamma(p_i, p'_j) = \pi_{ij}$ with $i, j = 1, \ldots, n$, where π_{ij} is defined as:

$$\pi_{ij} = \begin{cases} 1 & \tau_{ij} \text{ is defective, or } p'_j \text{ is defective.} \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

In the above equation, τ_{ij} refers to a switch transport path (see definition 4.3.2) ruled out by the path condition and p'_j is an outgoing port ruled out by the output condition. For each structural fault f the matrix $\Pi^f_{n\times n} = \left[\pi_{ij}\right]$ is constructed using the outcome of the functional mapping. It presents the defective paths and ports that must be avoided in the presence of a given structural fault f in order to confine all the fault effects.

To select a minimum number of discarded ports, a graph is constructed that contains a vertex for every incoming and outgoing port. An edge is added from the incoming port p_i to the outgoing port p'_j when $\pi_{ij} = 1$. In other words, the edges of the graph correspond to those transport paths that have been ruled out by the path condition.

The minimal set of the ports which confines all fault effects can be found by means of heuristics used in the classical vertex cover problems. Firstly, all output ports disabled by the output condition are marked as a part of the vertex cover in the graph. Using simple heuristics, additional vertices are then marked until all edges are covered. The marked vertices are the ports to be disabled in order to confine the effects of the fault f.

Example 4.4.1. We consider the five port switch of the 2D mesh topology with $P = (p_1, \ldots, p_5)$ and $P' = (p'_1, \ldots, p'_5)$, where the members of P and P' correspond, respectively, to the ports North, South, West, East and Core of the 2D mesh switch. Now, let us assume that under a given fault f, the transport paths $\tau_{15} : N \triangleright C'$, $\tau_{12} : N \triangleright S'$ and $\tau_{34} : W \triangleright E'$ have been ruled out by the path condition. According to equation (4.1), this leads to $\pi_{15} = 1$, $\pi_{12} = 1$ and $\pi_{34} = 1$.

Let us further assume the output port condition implies that the eastern outgoing port

E', i.e. the fourth port, is ruled out as well. Thus, inspired by equation (4.1), we have $\pi_{i4} = 1$, for $i = 1, \ldots, 5$. Matrix $\Pi^f_{5\times5}$ is represented as follows:

$$\Pi^f_{5\times5} = \begin{array}{c} \\ N \\ S \\ W \\ E \\ C \end{array} \begin{array}{c} \begin{array}{ccccc} N' & S' & W' & E' & C' \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

Example 4.4.2. Figure 4.6(a) represents the constructed graph for the five port switch using the matrix $\Pi^f_{5\times5}$, referred to earlier in example 4.4.1. We label the vertices corresponding to incoming ports $p_1, \ldots, p_5$ with N, S, W, E, and C, respectively. Furthermore, the vertices corresponding to outgoing ports $p'_1, \ldots, p'_5$ are labelled with N', S', W', E', and C'. Moreover, an edge is added between a pair of incoming and outgoing ports when the corresponding element in the matrix $\Pi^f_{5\times5}$ equals one.

Marked vertices of Fig. 4.6(b) are the ports that have to be disabled in order to confine all fault effects. The outgoing port E' has been ruled out by the output port condition (i.e. all elements in the fourth column of the matrix $\Pi^f_{5\times5}$ are one) and thus the vertex E' is marked to be disabled. By disabling E', the transport path $W \triangleright E'$ has been already covered. To avoid the paths $N \triangleright C'$ and $N \triangleright S'$, the efficient decision is to mark the input port N.

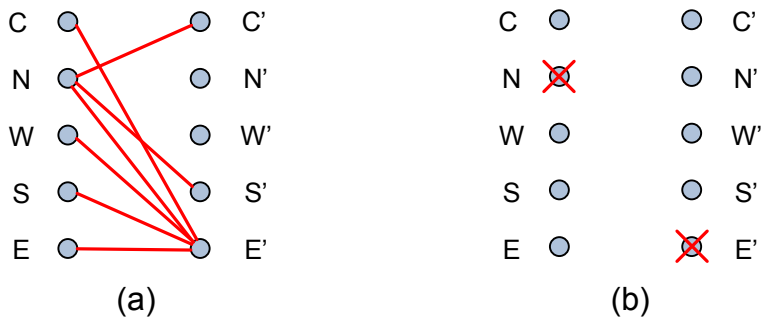


Figure 4.6: Minimal vertex cover for a faulty switch

4.5 Experimental Results

In this section, the proposed method is applied to retain fault-free functionalities of two typical switches designed for 2D mesh topology. The benefit of retaining the fault-free ports of defective switches for system use is discussed in Chapter 7.

4.5.1 Switch Characteristics

According to the specification of a typical NoC switch given in section 2.1.1, a switch for the 2D NoC mesh topology has been designed in VHDL. The router of the designed switch supports a fault tolerant routing scheme similar to [Zhang08b]. The switch has been synthesized using Synopsys Design Compiler and mapped to the lsi10k library in which the cell area of a two-input NAND gate is one area unit.

Since the memory elements are usually equipped with the advanced BIST infrastructure, there is no need to consider the FIFO storage elements in the diagnosis process. Every port has its own storage space and it is already known which port must be deactivated when a fault in the corresponding memory elements occurs. Here, the FIFO controller including head and tail counters and synchronization circuitry are considered as a part of the random logic in the switch. We have used a commercial ATPG tool and the experiments have been conducted based on the stuck-at fault model.

4.5.2 Remaining Functionality

The experimental result is obtained for two NoC switches. The first switch is specialized for transmitting packets with 12-bit flits and the second one supports 32-bit flit sizes as in the Intel Polaris [Vangal08]. Both of the switches have the same number of ports (5 incoming and 5 outgoing ports), and the same FIFO sizes. Table 4.1 summarizes the synthesis results for the switches and the routers, as we have synthesized the router logic separately. The router logic is identical for both switches. As shown in Table 4.1, both routers have the same number of flip-flops corresponding to the router states. The small difference between the total cell area of the routers is related

to different sizes of the head flits.

Table 4.1: Area report of the 12-bit and 32-bit switch

	12-bit switch	32-bit switch
number of I/O ports	323	723
number of combinational cells	2176	2842
number of flip-flips	2683	3583
total cells	4859	6425
Router:		
number of combinational cells	1065	1129
number of flip-flops	448	448
total cells	1513	1577

The number of fault sites in the 12-bit and 32-bit switches are 3301 and 4976, respectively. Using a commercial ATPG tool, the set of structural test patterns with 100% fault coverage is extracted. We applied the structural diagnosis algorithm from [Holst09] with the generated ATPG patterns. According to the diagnosis result, all faults in the switches are diagnosable. Besides, for the vast majority of faults, the diagnosis algorithm is able to determine a single fault site as the fault candidate. For the rest of faults, diagnosis finds a set of suspected candidates. However, the functional mapping indicates that all candidates have the same functional implications.

Both switches have been analysed using the functional mapping method presented earlier, and the dictionary has been created. We have conducted the port lookup to map each fault site to the specific incoming or outgoing ports to be deactivated. The fault site is an internal signal or gate within the random logic pinpointed by the structural diagnosis algorithm.

Table 4.2 shows the statistics of the generated dictionaries. We observe that more than 56% of the fault sites of the 12-bit switch is dedicated to a single incoming or outgoing port. In other words, a defect within this portion of the switch can be tolerated by disabling only one input or output port.

For the larger switch with wider flits, the method is even more effective. In the 32-bit switch, a larger portion (more than 72%) of the total fault sites is dedicated to a single incoming or outgoing port. The reason is that in the switch with wider flits,

Table 4.2: The portion of fault sites in the switch dedicated to each port and the router

	12-bit switch Stuck-at faults		32-bit switch Stuck-at faults	
	count	%	count	%
input C	228	6.91%	295	5.93%
output C	152	4.60%	445	8.94%
input S	221	6.69%	268	5.39%
output S	152	4.60%	448	9.00%
input W	221	6.69%	268	5.39%
output W	152	4.60%	448	9.00%
input N	224	6.79%	271	5.45%
output N	155	4.70%	448	9.00%
input E	221	6.69%	268	5.39%
output E	151	4.57%	445	8.94%
Ports portion sum	1877	56.86%	3604	72.42%
Router	1424	43.13%	1372	27.57%
Sum	3301	100%	4976	100%

the size of datapath elements is larger. This increases the probability of having faults in the datapath elements of the circuit rather than in the shared parts as in the router. Faults in the controlling logic of the switch may have a global effect on all switch ports. However, faults in the datapath elements (e.g. in the multiplexer outputs of the crossbar switch) may disrupt the operation of a single port. In the switch with wider datapath elements a larger portion of faults are dedicated to the switch ports and therefore can be masked by deactivating only one port instead of the entire switch.

4.6 Summary

This chapter presented the novel port-accurate diagnosis approach that determines the intact ports of defective switches in NoCs. Instead of disabling defective switches completely, the unaffected ports can be retained to maintain performability. The approach

has a great advantage especially for the larger switches with wider flits. Utilizing the proposed algorithm, which accurately determines the defective ports, avoids switch deactivation in many faulty cases.

5

Structural Software-Based Self-Test

This chapter explains the structural Software-Based Self-Test (SBST) method for NoCs. Test patterns targeting structural faults are turned into valid NoC packets. The network diagnosis method here is switch-accurate, that is all the faults in the network are mapped to faulty switches. The test technique provides a high structural fault coverage that makes it suitable for both manufacturing test and in-field testing. Besides, structural test data can be used for structural diagnosis of NoC switches.

5.1 Overview and Preliminaries

In Network-on-Chips (NoCs), test infrastructure like scan-design are widely employed to apply test patterns and reach acceptable test time (see 3.3.1). Structural testing targets faults of a predefined structural fault model like stuck-at faults and allows measuring the fault coverage as a quality metric. Depending on the expected product quality, more complex fault models like transition delay faults, bridging faults or cross-talk faults may be targeted as well. In contrast, functional testing targets certain functionalities of the NoC (see 3.3.2). Although functional testing provides some confidence about the correct functionality of certain parts of the network, a high structural fault coverage is not explicitly targeted.

For microprocessors, the benefits of structural and functional testing are com-

bined by a so-called structural Software-Based Self-Test (SBST) [Chen01, Corno03, Paschalis05, Kranitis05, Zhou06]. ATPG (Automatic Test Pattern Generation) provides deterministic, structural test patterns which are transformed into arguments of a sequence of valid instructions. In a similar way, the method here transforms deterministic test patterns into packets of an NoC to conduct SBST for NoCs. Structural faults in the switches and links are targeted and tested by valid NoC packets without the need for dedicated test infrastructure.

5.1.1 Target Architecture for SBST

The Switch Under Test (SUT) is tested by applying a set of test patterns to its incoming ports and observing the test responses at the outgoing ports. Since the SBST patterns form valid NoC packets, they can be fed to the NoC externally as long as they pass the SUT, without requiring to put the system in a non-functional test mode. Here, we assume that test packets are applied by the software running on the Processing Elements (PEs) attached to the NoC. The resulting test responses are captured and evaluated by the test programs in the ambient PEs. Figure 5.1 illustrates an example for the 2D mesh topology, in which every switch is connected to four neighbouring switches and a PE is attached to each switch. Depending on the network topology and the switch location, the SBST pattern generation is confined such that only available neighbouring PEs contribute in testing. For example, in a 2D mesh a switch at the boundary has three neighbours, consequently its test patterns contain input values for only three incoming ports of the switch.

The SBST process starts when all PEs surrounding the Switch Under Test (SUT) have sufficient resources to run the test program. A local signal (such as the Acknowledge/Request signal used for the link flow control) can be reused to synchronize the launch of the test programs running on the PEs involved in testing the SUT. The switches and PEs give the highest priority to test packets and bypass their caches. Because surrounding switches are identical, the SUT access time through all the incoming links is deterministic. Moreover, once the test begins, normal packets are not routed through the SUT. The entire NoC can be tested by consecutively testing all contained switches.

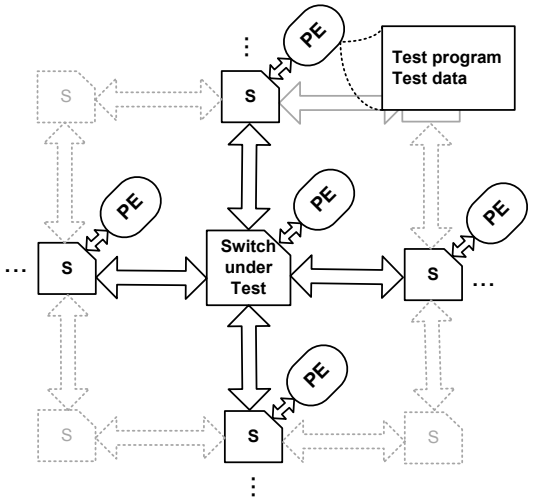


Figure 5.1: Target architecture for SBST of a switch

5.1.2 Pattern Generation

The most challenging part of SBST for NoCs is the generation of efficient test patterns in the packet format with high structural fault coverage. Since we have only access to the functional input and output ports of the switches, functional patterns are used in SBST. Therefore, in contrast to scan-based testing, direct controllability and observability of the states of the switch (i.e. pseudo primary inputs and outputs) is not possible. Instead of that, each pattern includes a sequence of assignments for functional inputs. Achieving a high structural fault coverage with such patterns poses some challenges similar to those of sequential test.

Sequential Tests

Figure 5.2 shows the general structure of sequential circuits, comprising a combinational logic and some flip-flops (FF). The flip-flops constitute the state registers of the design. In sequential tests, a sequence of vectors provide the excitation and propagation conditions for the faults. Instead of initialising the state registers for test, functional input assignments modify the circuit states. Therefore, the test becomes sensitive to the order in which the input stimuli is applied [Miczo03]. Test pattern generation

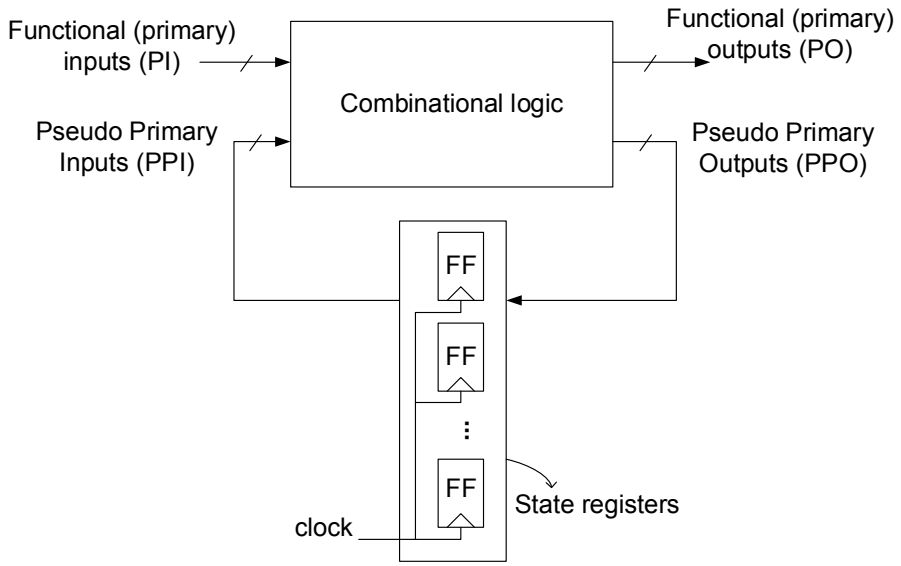


Figure 5.2: General structure of sequential circuits

using functional inputs has been attractive for designers and researchers as it can mitigate the full-scan overheads of area and test application time. However, sequential ATPG algorithms cannot provide the same performance as the combinational ATPG with high fault coverage and 100% fault efficiency [Kim01].

By replicating combinational logic of the circuit into several time frames, known as *time-frame expansion*, test generation for sequential circuits becomes similar to that for combinational circuits. Figure 5.3 demonstrates an example of time-frame expansion in which the sequential circuit is unrolled for several time-frames. Each time-frame includes a copy of the combinational logic. The pseudo primary inputs are fed by pseudo primary outputs coming from the previous time-frame.

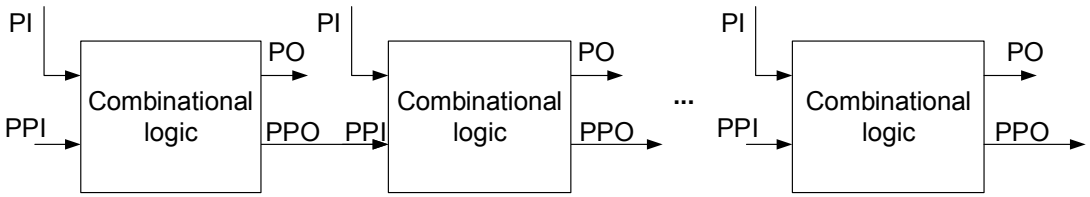


Figure 5.3: Time-frame expansion

Time-frame expansion constitutes a combinational representation of sequential circuits. However, to limit the test length, an upper bound for the number of time-frames is required, which is known as *sequential depth*. The partial scan technique proposed in [Kunzmann90] guarantees that every acyclic sequential circuit, i.e. it has no feedback among its flip-flops, has a well-defined sequential depth and is initializable [Bushnell00]. Reference [Chakradhar95] shows how to make a circuit acyclic by scanning a minimum set of flip-flops. To determine the sequential depth of the circuit, a directed graph called s-graph [Bushnell00] must be extracted. The s-graph represents dependencies among the circuit flip-flops. It contains a vertex for every flip-flop in the circuit. Moreover, an edge is established between a pair of vertices, namely A and B, if there exists a combinational path between the corresponding flip-flops in the circuit, from A to B. Figure 5.4 represents a sample sequential circuit and its s-graph. The number of vertices in the longest path of the s-graph is the sequential depth of the circuit. Accordingly, the sequential depth of the circuit in Fig. 5.4 is three.

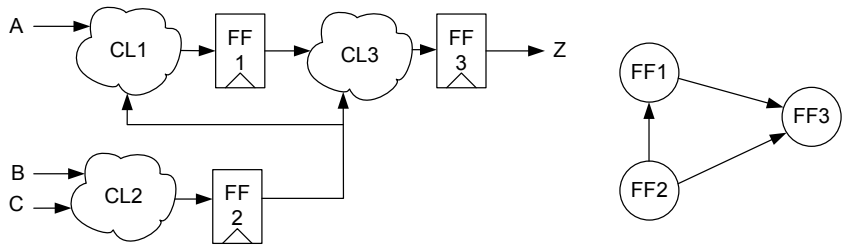


Figure 5.4: S-graph of a sequential circuit (CL: combinational logic)

The time-frame expansion with the bounded sequential depth reduces the complexity of the sequential ATPG. However, since a single fault may appear in more than one time frame, the pattern generation algorithm requires detection of multiple faults, which is not handled by existing combinational ATPG programs.

5.2 SBST Pattern Generation for NoCs

The principle of SBST pattern generation is depicted in Fig. 5.5. Similar to the conventional pattern generation structure, it contains the *original circuit* block (corresponding to the sequential switch) and a *faulty copy*, which is a copy of the original circuit with

an injected *fault* on the victim signal, carrying the faulty value. A test pattern now is an input assignment that activates the fault and makes it observable at the circuit functional outputs so that the *comparators* rise a mismatch. As the input assignment must be in the NoC packet format, the inputs are confined to fulfil the requirements defined by *NoC-packet constraints*.

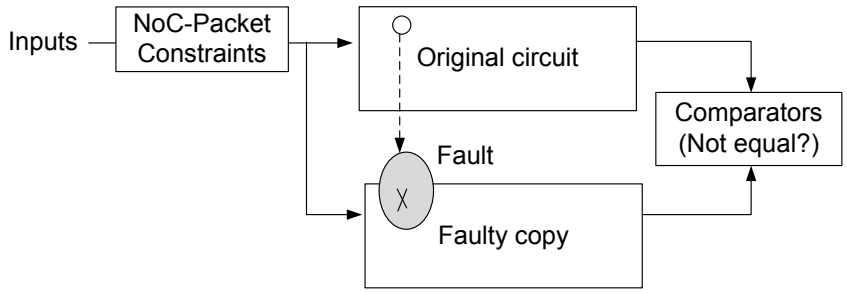


Figure 5.5: Structure of the SBST pattern generation

Accordingly, the SBST pattern generation reflects the following aspects:

1. Original Circuit: The combinational logic of the switch under test and its interconnect links are described. Furthermore, the model reflects the sequential behaviour of the switch by time-frame expansion.

2. Faulty Copy: It is constructed similar to the original circuit reflecting the combinational logic and the sequential behaviour.

3. Fault Model: Classic fault models such as stuck-at faults are not sufficient to reason about arbitrary defects in the recent process technologies. Hence, the Conditional Line Flip (CLF) calculus (see section 2.2) is used as a generalized fault model to describe arbitrary defect mechanisms in the switch logic and the links. Furthermore, the model supports fault excitation at multiple cycles individually.

4. Comparators: In the SBST method, test responses are captured from functional outputs, similar to sequential tests. Therefore, the comparators are made over the functional outputs of the switch and check the inequality among the outputs in the original circuit and the faulty copy.

5. NoC-Packet Model: According to the described SBST properties, in order to utilize the packet-based communication platform of the NoC, only valid packets are accepted as test patterns. The model defines suitable constraints to fulfil this requirement.

We have formulated the SBST pattern generation as a Boolean SATisfiability (SAT) problem, which offers high flexibility to model the constraints of the SBST pattern generation. The rest of this chapter will discuss in detail the construction of each block of the SBST pattern generation in the SAT instance. Preliminaries of the Boolean satisfiability have been summarized in section 2.5.

5.2.1 Circuit Modelling and Sequential Mapping

The NoC switch structure is represented in conjunctive normal form (CNF). The sequential switch structure is modelled by time-frame expansion of the combinational core of the switch. The combinational core of the circuit is extracted by removing the flip-flops and replacing the input/output signals of the flip-flops by pseudo primary output/input signals, respectively.

To represent the combinational circuit structure into CNF, each signal is represented by a variable and the gates are modelled by converting the Boolean functions into CNF as presented in section 2.5. Let I_g and O_g be the sets of input and output literals of a gate g. Furthermore, let C be the set of logic gates in the combinational netlist and L be the set of literals corresponding to the signals of the circuit so that:

$$\forall\, g \in C: \quad I_g \subseteq L\,,\, O_g \subseteq L \tag{5.1}$$

According to definition 2.5.2, the characteristic formula for the entire circuit is extracted by taking the conjunction of the CNF formulas of all gates in the netlist. It is represented by Φ_{C} as follows:

$$\Phi_{\mathrm{C}} = \bigwedge_{g \in C} \mathrm{CNF}(g, I_g, O_g) \tag{5.2}$$

The number of clauses in the resulting formula is linear to the number of modelled gates at the cost of introducing new literals for each signal. For T being the sequential depth of the switch, T copies of the combinational switch are required to model the sequential behaviour of the switch. Each combinational switch is modelled in CNF as explained above while the signals of each combinational instance get their unique variable id.

The sequential instance Φ_{S} is formulated as given in Eq. (5.3). It comprises the conjunction of T copies of the combinational instance, Φ_{C}, and some additional clauses to make equivalent literals for the pseudo primary signals. In this formula, $\Phi_{\mathrm{C},t}$ indicates to the copy t of the combinational instance, PPI_t refers to the ordered set of literals for pseudo primary inputs of copy t, and PPO_{t-1} refers to the ordered set of literals for pseudo primary outputs of copy $(t-1)$. Each pair of literals (p_i, q_i), $p_i \in PPI_t$, $q_i \in PPO_{t-1}$ represents the equivalent pseudo primary signals of two consecutive copies.

$$\Phi_{\mathrm{S}} = \bigwedge_{t=1}^{T} \Phi_{\mathrm{C},t} \wedge \bigwedge_{t=2}^{T} \mathrm{CNF}(PPI_t \overset{*}{\Leftrightarrow} PPO_{t-1}) \tag{5.3}$$

The last term of Eq. 5.3 implements the equivalent literal sets (see definition A.1.3 in appendix A.1) for all copies. It causes the literals of pseudo primary inputs of each copy to get the same values as the respective literals of pseudo primary outputs of the previous copy. It is noted that the clauses for equivalent literals can be omitted if the literals for pseudo primary outputs are directly used as the pseudo primary input literals while constructing the next copy.

Example 5.2.1. We construct the SAT instance for a small sequential circuit. The gate-level representation for the least significant bit (LSB) of a synchronous binary counter is shown in Fig. 5.6. A flip-flop (FF) holds the current state. The counter counts up whenever the primary input *z* is set to one. The primary output *x* shows the current value of the least significant bit.

Figure 5.7 represents the time-frame expansion of the circuit for two time-frames. The label on each signal line represents the corresponding variable name in the SAT instance. As *x* and *h* carry the same value, only one literal is used in the CNF formula.

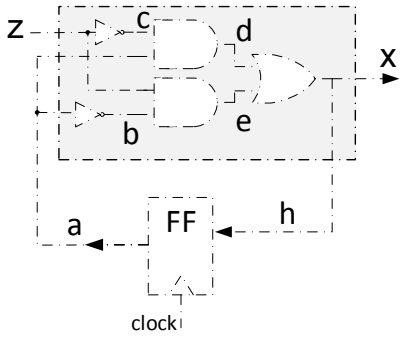


Figure 5.6: Logic of a binary counter for the least significant bit (LSB)

According to Eq. 5.2, the CNF formula for the combinational core in the first time frame is constructed as the conjunction of the CNF formula of the gates in the combinational netlist as follows:

$$\begin{aligned}\Phi_{\mathrm{C},1} \;=\; & \mathrm{CNF}(\mathrm{NOT}, \{z_1\}, \{c_1\}) \wedge \mathrm{CNF}(\mathrm{NOT}, \{a_1\}, \{b_1\}) \wedge \\ & \mathrm{CNF}(\mathrm{AND}, \{c_1, a_1\}, \{d_1\}) \wedge \mathrm{CNF}(\mathrm{AND}, \{z_1, b_1\}, \{e_1\}) \wedge \\ & \mathrm{CNF}(\mathrm{OR}, \{d_1, e_1\}, \{h_1\})\end{aligned}$$

The CNF formula for the second copy, $\Phi_{\mathrm{C},2}$, is created similarly by using the respective literals of the second copy. A buffer gate (coloured in grey in Fig. 5.7) is inserted to show the equivalent pseudo primary signals, h_1 and a_2. The resulting CNF formula for the given sequential circuit will be as follows:

$$\Phi_{\mathrm{S}} = \Phi_{\mathrm{C},1} \wedge \Phi_{\mathrm{C},2} \wedge \mathrm{CNF}(h_1 \;\Leftrightarrow a_2)$$

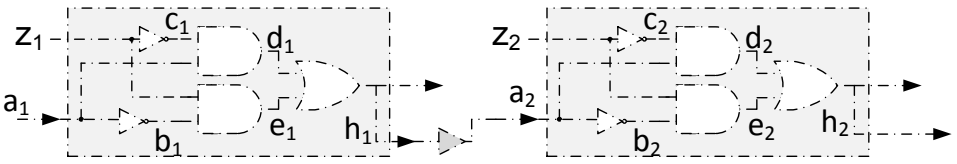


Figure 5.7: Time-frame expansion for two time-frames

5.2.2 Faulty Copy

The faulty circuit can be constructed by making a copy of the CNF formula of the original circuit and renaming the variables. However, the logic values of the signals in the faulty copy are the same as in the original circuit, except for those signals located in the output cone of the fault site. The output cone of a signal includes the gates that lie on a path between the signal and a circuit output.

Therefore, to simplify the SAT instance, only the clauses of those gates in the output cone of the fault site are added to the faulty copy as illustrated in Fig. 5.8. In each time-frame, the literal of the fault site and the output literals of the gates in its output cone are renamed. Other literals in the faulty instance are deduced from the literals in the original circuit instance.

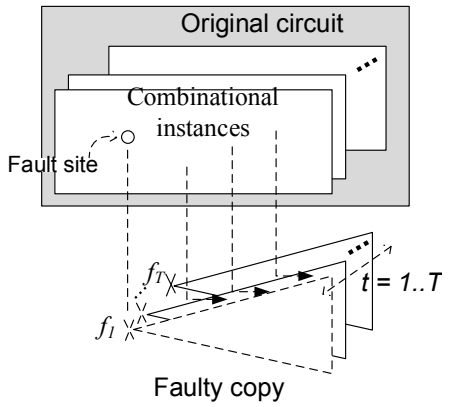


Figure 5.8: Faulty copy structure in the SAT instance

As the switch is a sequential circuit, some faults may firstly propagate to the internal states and after a few cycles become observable at primary outputs. If the fault propagates to pseudo primary outputs, the faulty circuit models additionally the gates in the output cone of the equivalent pseudo primary inputs as it is illustrated in Fig. 5.9. The dashed triangles indicate to the output cone of the fault site. The second time-frame includes additionally the output cone of those pseudo primary inputs to which the fault has propagated at the first time-frame. The cones may have some overlaps. In order to keep the SAT instance smaller, each gate is included only once in the model. For a fault location f, the resulting faulty instance is named Φ_{S}^{f}.

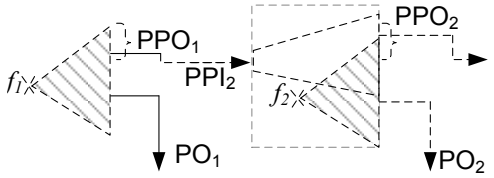


Figure 5.9: Faulty copy: the output cone of the fault and the cone of some pseudo primary inputs to which the fault propagates

Example 5.2.2. We construct the faulty instance for the sample circuit of example 5.2.1 considering the fault site on signal d. Figure. 5.10 illustrates the time-frame expansion of the original circuit for two time-frames. The gates that are added to the faulty instance are coloured in grey and the new literals are marked with a prime symbol. In the first time-frame, the corresponding literal to the fault site d is renamed to d'_1. The OR gate is located in the output cone of d and therefore is added to the faulty instance. Its output signal h gets a new literal name h'_1. As signal e is not in the output cone of the fault, literal e_1 is deduced from the original circuit. The combinational faulty instance at the first time-frame will be:

$$\Phi^{f}_{\mathrm{C},1} = \mathrm{CNF}(\mathrm{OR}, \{d'_1, e_1\}, \{h'_1\})$$

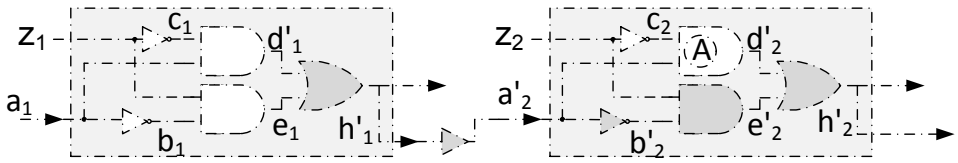


Figure 5.10: The counter LSB logic for two time-frames. The gates in grey are added to the faulty instance.

In the second time-frame, the fault site and its output cone are modelled similar to the first time-frame. Because the fault has propagated to the pseudo primary output h, its effect must be considered at the second time-frame. In the second copy, the pseudo primary input a gets a new literal a'_2 and the gates in its output cone are also added to the faulty instance (gates coloured in grey in the second time-frame). The

combinational faulty instance at the second time-frame will be:

$$\begin{aligned}\Phi^{f}_{\mathrm{C},2} &= \mathrm{CNF}(\mathrm{NOT}, \{a'_2\}, \{b'_2\}) \wedge \mathrm{CNF}(\mathrm{AND}, \{z_2, b'_2\}, \{e'_2\}) \wedge \\ &\quad \mathrm{CNF}(\mathrm{OR}, \{d'_2, e'_2\}, \{h'_2\})\end{aligned}$$

Since c_2 is neither in the output cone of the fault site nor in the output cone of a_2, its literal is deduced from the original circuit instance. Moreover, gate (A) in Fig. 5.10 is not added to the faulty instance, because d'_2 is the fault site and already carries the faulty value. As the fault has propagated to a pseudo primary output, the faulty instance includes also the clauses which implement the equivalent pseudo primary signals. The resulting faulty instance is as follows:

$$\Phi^{f}_{\mathrm{S}} = \Phi^{f}_{\mathrm{C},1} \wedge \Phi^{f}_{\mathrm{C},2} \wedge \mathrm{CNF}(h'_1 \Leftrightarrow a'_2)$$

5.2.3 Fault Description in the SAT Instance

To develop a generalized SBST method targeting arbitrary defects in switches and links, the Conditional Line Flip (CLF) fault model is used. In order to generate test patterns for a fault in the switch, the fault must be activated. It means, the CLF condition becomes true in at least one time-frame and the victim signal carries different values in the original and faulty circuit.

In a single time-frame t, the CLF f is represented by the victim line denoted by variable l_t in the original circuit, and l'_t in the faulty copy. A new variable c_t denotes the condition of the CLF and relates the victim line literals in the original and the faulty circuit so that: $l'_t = (l_t \oplus c_t)$ (see CLF definition in section 2.2). Figure 5.11 visualizes the CLF fault modelling in the SAT instance for a single time-frame.

When the circuit is faulty, the fault is present in all time-frames. The CLF clauses are defined for all time-frames and must be independently satisfiable. Therefore, the fault model in conjunctive normal form (CNF), named Φ_{CLF}, is defined as the conjunc-

tion of the CLF clauses in all time-frames as follows:

$$\Phi_{\mathrm{CLF}} = \bigwedge_{t=1}^{T} \mathrm{CNF}(\mathrm{XOR}, \{l_t, c_t\}, \{l'_t\}) \tag{5.4}$$

Since the condition literals denoted by c_t are free literals in the model, the SAT solver may assign values to them such that the fault can be activated in different cycles independently. Therefore, it can model the defects that their nature changes over time. With respect to the targeted faults, appropriate functions [Wunderlich10] can be defined as a condition, which activates a fault only in certain cycles.

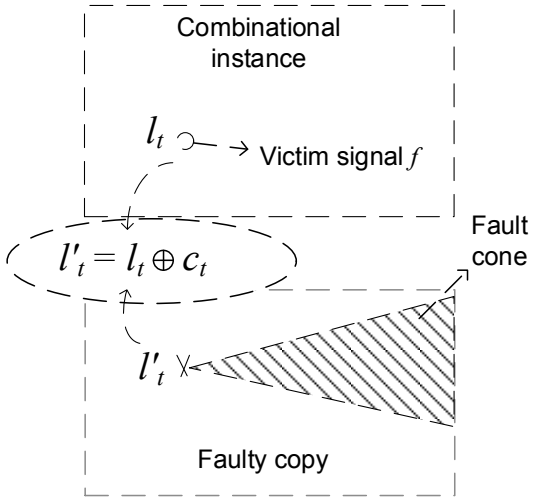


Figure 5.11: Modelling a CLF in a single time-frame

5.2.4 Comparators Modelling

The excited fault needs to produce an error on the primary outputs of the faulty copy, i.e. the circuit responses must differ in the original and faulty copy. Therefore, all the output literals of the original circuit are compared against those of the faulty circuit. As denoted in Fig. 5.12, this comparison is performed by a set of XOR gates which compare pairs of primary outputs in the original circuit and the faulty copy. For n being the number of primary outputs to which the fault propagates, the XOR gate

comparators at the time-frame t are defined as follows:

$$\Phi_{\mathrm{cmp},t} = \bigwedge_{j=1}^{n} \mathrm{CNF}(\mathrm{XOR}, \{o_{tj}, o'_{tj}\}, \{m_{tj}\}) \tag{5.5}$$

In the above formula, o_{tj} and o'_{tj} stand for the j^{th} output bit in the original and faulty circuit, respectively. In addition, m_{tj} is a new variable carrying the output of the corresponding XOR gate.

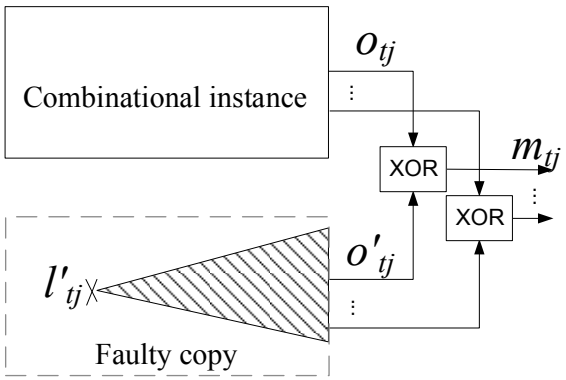


Figure 5.12: Comparators modelling

The test pattern generation must find an input assignment so that the responses of the good and faulty copy mismatch. As it is sufficient to detect a CLF in one time-frame, it is sufficient to have one XOR gate out of all time-frames to be evaluated to true. This is implemented by the disjunction over the output variables of all XOR gates in the entire time-frames. The resulting comparators instance comprises the conjunction of the XOR gate clauses of all time-frames and one additional clause that obliges a mismatch in the good and faulty copy for at least one output bit as follows:

$$\Phi_{\mathrm{obs}} = \bigwedge_{t=1}^{T} \Phi_{\mathrm{cmp},t} \wedge \bigvee_{t=1}^{T} \bigvee_{j=1}^{n} m_{tj} \tag{5.6}$$

Example 5.2.3. We construct the comparators instance for the counter's LSB logic of example 5.2.1 for two time-frames. The circuit has one primary output denoted by literals h_1 and h_2 in the first and second time-frames, respectively. These literals are denoted by h'_1 and h'_2 in the faulty copy as shown in example 5.2.2. In each time-frame

one XOR gate is required for comparison. Two new literals m_1 and m_2 show the output literals of the XOR gates in the first and second time-frames:

$$\Phi_{\text{cmp},1} = \text{CNF}(\text{XOR}, \{h_1, h_1'\}, \{m_1\})$$
$$\Phi_{\text{cmp},2} = \text{CNF}(\text{XOR}, \{h_2, h_2'\}, \{m_2\})$$

The comparators instance will be:

$$\begin{aligned}\Phi_{\text{obs}} \;=\; & \text{CNF}(\text{XOR}, \{h_1, h_1'\}, \{m_1\}) \wedge \\ & \text{CNF}(\text{XOR}, \{h_2, h_2'\}, \{m_2\}) \wedge \\ & (m_1 \vee m_2)\end{aligned}$$

5.2.5 Modelling of Valid NoC Packets

The original circuit model in the SBST pattern generation instance is constructed by time-frame expansion of the switch for T consecutive cycles. The instance contains literals for the primary inputs of T combinational instances, namely $PI_1, PI_2, \ldots, PI_T$. A SBST pattern defines an assignment for the primary input literals PI_1 to PI_T to be applied to the switch inputs at cycles 1 to T, respectively.

To ensure test patterns are valid NoC packets, the appropriate clauses which define the packet characteristic must be included in the SAT instance. Here, we consider the packet format presented in section 2.1.1 in which a packet starts with the head flit that is followed by a number of data flits and ends with a tail flit.

Figure 5.13 demonstrates a packet including head, tail and data flits which is applied to data inputs of the switch port i. In order to generate the test pattern in the packet format, the input sequence must contain head and tail flits in a manner that every head flit is followed by a tail flit:

$$\forall\, t,\ 1 \leq t \leq T\ :\quad (D_{i,t} \text{ is head}) \leftrightarrow (D_{i,t+fpp-1} \text{ is tail}) \tag{5.7}$$

In the above formula, fpp is the number of flits per packet, and $D_{i,t}$ refers to the switch data inputs of port i at cycle t of the test pattern. Since head and tail flits are

mandatory for a packet, fpp is at least 2. However, for $fpp > 2$ the intermediate flits must be defined as data flits:

$$\begin{aligned} &\forall\, t,\ 1 \leq t \leq T\,, \\ \forall\, j,\ t < j < (t + fpp - 1) \quad &: \qquad (D_{i,t} \text{ is head}) \leftrightarrow (D_{i,j} \text{ is data}) \end{aligned} \tag{5.8}$$

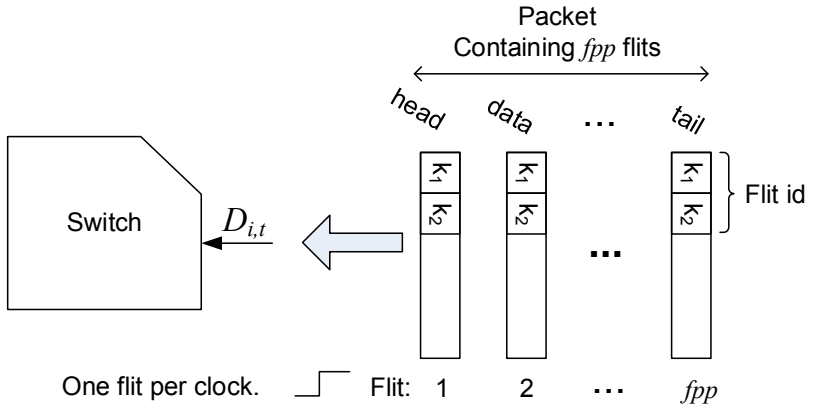


Figure 5.13: Flits of a packet are applied to data inputs of the switch port i.

To model the test packets independent of the packet length, the sequence of the flits are taken into account rather than the number of flits per packet. Figure 5.14 illustrates the finite state machine (FSM) to define the flit sequence of the valid NoC packet so that for every $fpp > 2$ the formulas (5.7) and (5.8) become true. In the state machine, the idle state refers to the cycles in which no flit arrives.

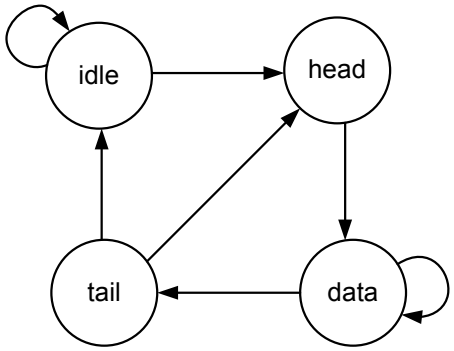


Figure 5.14: Finite state machine for the flit sequence of valid packets

A few control bits (flit id) determine the flit type. According to the FSM of

Fig. 5.14, the Boolean function for each control bit is extracted. A switch has several ports. The Boolean functions are defined for all switch ports and converted to the clauses. The resulting instance is called $\Phi_{\text{valid packet}}$ and is added to the pattern generation instance. Introducing these clauses guarantees that the SAT instance is only satisfiable if test patterns are in the form of the NoC packets.

Adding valid packet clauses helps the SAT to find test patterns for the faults in the controlling logic which are difficult to test. The NoC packets invoke the control flow of the switch similar to instructions in the SBST of processors. Valid packet clauses confine the input space to those which specifically can sensitize the faults in the controlling parts. For example, a test pattern may contain several NoC packets which arrive at different switch ports at the same time, requesting the same outgoing port. This activates the scheduler of the switch to decide which packet has the priority. Moreover, as formula (5.7) does not impose any constraint on the time in which the head flit arrives, the packet may arrive at any cycle between 1 and T. In this case, if the outgoing port has been already assigned to a packet, the new incoming packet must not overwrite this setting.

5.3 Pattern Generation Flow

Inspired by the above discussion, the SAT instance for generating SBST patterns for NoCs is constructed as the conjunction of the clauses of the sequential switch model, Φ_{S}, the faulty copy, Φ_{S}^{f}, the clauses describing a CLF fault, Φ_{CLF}, the comparators model, Φ_{obs}, and the clauses to ensure only valid packets are accepted as a solution:

$$\Phi_{\text{SBST}} \;=\; \Phi_{\text{S}} \wedge \Phi_{\text{S}}^{f} \wedge \Phi_{\text{CLF}} \wedge \Phi_{\text{obs}} \wedge \Phi_{\substack{\text{valid}\\ \text{packet}}}. \tag{5.9}$$

To generate SBST patterns for all CLFs in the switch, as shown in Fig. 5.15, the SBST pattern generation selects the first undetected CLF from the fault list and builds Φ_{SBST}. To shorten the test time, the process starts with a small T of three, two cycles for head and tail flits to arrive and one cycle for the internal router process. If Φ_{SBST} is satisfiable, the values of the input literals are stored as a test pattern. Then, the

test pattern is applied to the SAT instance as a constraint. The process searches for additional faults detected by this pattern and prunes them from the CLF fault list. The process continues until test patterns are found for all faults which are detectable for the given time-frame. To increase the coverage, the faults that are not testable under the current T are reprocessed by iteratively increasing the number of time-frames to $T + 1$ and repeating the SBST pattern generation process. This continues up to the sequential depth of the circuit plus one, which is the upper bound of the test sequence for any detectable fault in the circuit [Bushnell00].

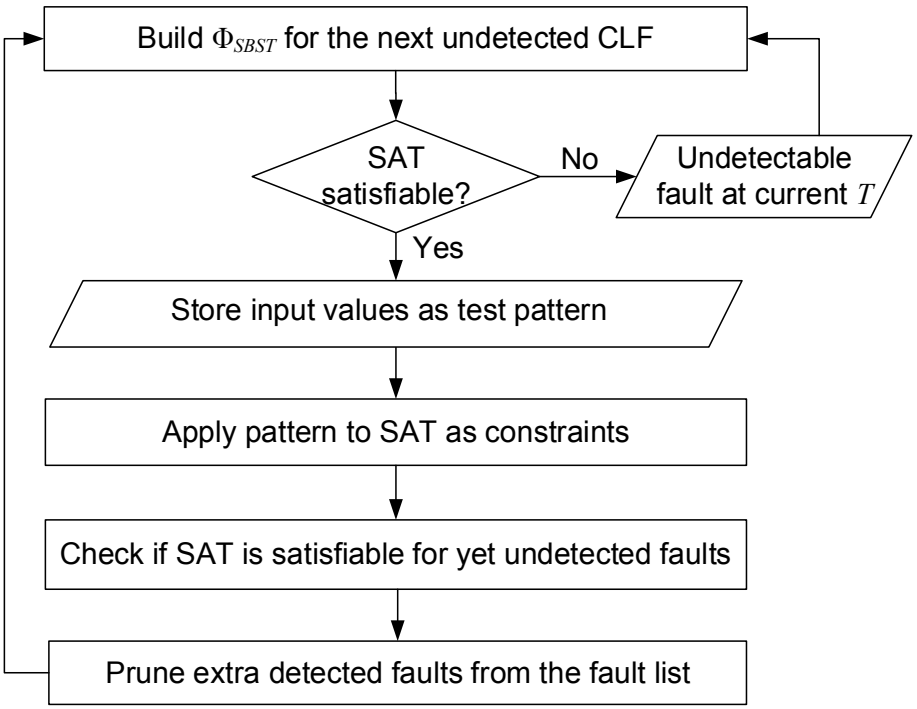


Figure 5.15: The flow of SBST pattern generation

5.4 Test Cost

For the test volume, the size of the test pattern set in bits is computed. It reflects the memory space which is required to store the test data. For a sequential depth T, test patterns of SBST contain the value of the primary inputs in T consecutive cycles as the

input pattern and the value of the primary outputs as the test response. Thus, the test volume of the SBST test data is computed as follows:

$$\text{Test volume}_{\text{SBST}} = T \times (|PIs| + |POs|) \times n\,, \tag{5.10}$$

in which $|PIs|$ stands for the number of primary inputs, $|POs|$ the number of primary outputs, and n refers to the number of test patterns. It is noted that storage of the test patterns in the presented SBST method is not a problem, because only a small portion of memory space in processing elements has to be dedicated to that.

Every SBST test pattern is generated with regard to T cycles of the switch operation. Thus, the number of clock cycles required to apply input patterns and get responses is equal to T. Moreover, one additional cycle is needed to reset the flip-flop states before applying the next test pattern, as it is assumed by the pattern generation procedure. Accordingly, the test time for applying n test patterns is calculated as follows:

$$\text{Test time}_{\text{SBST}} = n \times (T + 1)\,. \tag{5.11}$$

5.5 Experimental Results

5.5.1 Switch Characteristics

According to the specification of the typical NoC switch given in section 2.1.1, a switch is designed in VHDL. The switch design is parametrised to be adjusted based on the number of switch ports and the flit width. The router, the crossbar and additional control logic to generate and process the handshake signals of the ports are random logic and are synthesized from the VHDL model and mapped to the target technology library.

The SBST pattern generation method is independent of the internal architecture of the switch and the topology. Only the number of switch ports which are available for testing must be known. Besides, in order to introduce valid packet literals, the flit width and the flit format is required.

We evaluate the efficiency of the presented SBST method on the typical switch adjusted for the mesh topology, consisting of five incoming and outgoing ports. The switch is synthesized using Synopsys Design Compiler. The target library lsi10k is constrained to basic gate primitives. Since the memory elements are usually equipped with advanced memory BIST, there is no need to consider the incoming buffers in the SBST pattern generation process.

Table 5.1 summarizes the synthesis statistics of the switch for a flit width of 8 bits. The first four columns report the circuit name, the number of primary inputs (PIs), primary outputs (POs), and the number of flipflops (FFs), which is equal to the number of pseudo primary inputs/outputs. The remainder of the table reports the number of gates and the size in cell area units where the cell area of a two-input NAND gate in the library is one area unit.

Table 5.1: NoC switch characteristics

Name	# PIs	# POs	# FFs	# gates	Cell area
Switch	58	50	497	3618	8130

5.5.2 Stuck-at Faults

Table 5.2 presents the result for the complete SBST process which starts with an initial time-frame expansion for three cycles. The second row in the table shows the number of faults which are testable for each T. The third row shows the number of undetectable faults and the last row reports the overall fault coverage.

The sequential depth of the switch is eight, hence it is sufficient to examine at most nine time-frames during the pattern generation process. For the initial time-frame

Table 5.2: Pattern generation for stuck-at faults

	$T = 3$	$T = 4$	$T = 5$	total
Detected faults	10491	121	1	10613
Undetectable faults	503	380	379	379
Fault coverage	95.42%	96.52%	96.53%	96.53%

expansion, $T = 3$, the SBST process is able to generate test patterns for more than 95% of the faults. However, for $T = 4$ and $T = 5$, test patterns are found for 122 additional faults. For larger values of T no additional patterns were found. As the switch has been unrolled up to its sequential depth, the faults which remain undetectable have no functional influence during the normal operation of the switch.

The last column of Table 5.2 sums up the detailed result and presents ultimate statistics of the SBST pattern generation for stuck-at faults in the switch. Among 10994 collapsed faults in the switch, 10613 faults are testable and for the rest, the SAT model proves that there exists no valid NoC packet such that the fault is propagated to the functional switch outputs.

For comparison, a commercial sequential ATPG tool without any constraints was applied. It reports a fault coverage of 83.29% and a fault efficiency of 86.33%, while 2.39% of the faults are undetectable. All faults classified as undetectable by the ATPG are functionally redundant. The fault coverage for stuck-at faults obtained by a scan-based test strategy of a commercial ATPG tool was computed as well. There, 506 faults out of 14355 collapsed faults in the switch are recognized *Unused* by the tool, leading to the fault coverage of 96.47% and the fault efficiency of 100%. As it is shown in the last column of Table 5.2, the fault coverage of the presented SBST approach is roughly the same (96.53%). However, these numbers cannot be directly compared as the number of faults in the scan-based model and the unmodified switch do not match exactly.

Test Cost

Table 5.3 reports some statistics about the test cost of the generated test patterns. Firstly, the number of test patterns generated in each step of the SBST process is listed. The overall test volume is a summation of the test volume of each step, which is computed according to Eq. (5.10). The next row reports the test time. The test time of each step is computed as given in Eq. (5.11).

To shed some light on the test cost of the SBST method compared to the conventional scan-based test approaches similar to [Amory05b], we applied commercial

Table 5.3: Test cost of SBST

	$T = 3$	$T = 4$	$T = 5$	total
Test patterns	504	32	1	537
Test volume (bits)	163296	13824	540	177660
Test time (cycles)	2016	160	6	2182

ATPG with test compression targeting stuck-at faults in the switch and extracted the result of the test pattern generation. Any scan test pattern consists of the value of both primary inputs and the scan registers as the input vector, and the value of primary outputs and the scan registers as the test response. The test volume of the scan test data is computed as:

$$\text{Test volume}_{\text{scan}} = (2 \times |\textit{scan cells}| + |\textit{PIs}| + |\textit{POs}|) \times n\,, \tag{5.12}$$

in which $|\textit{scan cells}|$ stands for the number of scan cells (i.e. flipflops) and n refers to the number of test patterns. $|\textit{PIs}|$ and $|\textit{POs}|$ stand for the number of primary inputs and primary outputs, respectively.

The test time for testing the switch using the scan method including shift-in/shift-out and load/capture cycles is calculated as follows:

$$\text{Test time}_{\text{scan}} = (|\textit{scan cells}| + 1) \times n \;+\; |\textit{scan cells}|\,. \tag{5.13}$$

Because the shift-in and shift-out phases of two successive test patterns are conducted simultaneously, the formula counts the number of scan cells only once for shifting phase of each test pattern and needs one extra cycle for load/capture. To shift out the test response of the last test pattern, extra shift cycles equal to the number of scan cells are required at last. The scan test time however may reduce by introducing multiple scan chains to shift the scan data in parallel.

The ATPG tool generates 135 test patterns for the synthesized switch. We substitute in (5.12), (5.13), the number of PIs, POs, the number of flip flops as scan cells (reported in Table 5.1), and the number of test patterns as reported by ATPG. Accordingly, the test data volume of the scan test is 148770 bits and the test time is 67727 cycles. The

test time may vary depending on the number of scan chains and available test pins in the switch. To achieve a test time comparable to the presented SBST test time, at least 30 scan chains are needed, which imposes extra hardware for test. The SBST approach however brings a high fault coverage without introducing hardware overhead, which is 30x faster than the full-scan technique.

5.5.3 Other Fault Types

Since the SBST approach relies on the Conditional Line Flip (CLF) as the fault model, it can generate test patterns targeting various kinds of structural faults. Here, we investigate the efficiency of the SBST method for the bridging fault and the crosstalk.

For the bridging faults, we consider the 4-way bridge in which two signal lines a and b swap the values. The respective CLF formulas have been denoted in section 2.2. Since in each bridging fault two signals are affected by the fault, the fault model (Φ_{CLF}) of the SBST pattern generation instance needs to create two sets of condition literals describing the bridge. Furthermore, the faulty instance Φ_{S}^{f} includes the output cone literals corresponding to both victim lines. The pairs of suspected signals for the bridging fault are selected according to [Tran06a].

The second column of Table 5.4 presents the result of SBST pattern generation for bridging faults in the switch. The result indicates that the SBST pattern generation with the initial time-frame expansion for three cycles is able to achieve 98.88% fault coverage with 159 test patterns, resulting in very short test time.

Table 5.4: SBST pattern generation statistics

	Bridging faults $T = 3$	Crosstalk $T = 3$
Faults	10834	70
Detected faults	10713	70
Undetectable faults	121 (1.11%)	0
Fault coverage (%)	98.88	100
Test patterns	159	22
Test volume (bits)	51516	7128
Test time (cycles)	636	88

To investigate the crosstalk effect on communication links, pairs of neighbouring signal lines on each input port of the switch are considered as aggressor and victim lines. The respective CLF formula for the crosstalk has been denoted in section 2.2. The condition clause in the CLF representation is a function of the victim and the aggressor line values. In addition, it requires the last value of the aggressor line. This value comes from the aggressor literal in the previous copy of the unrolled switch. For the 8-bit switch, 14 crosstalk faults are injected in each incoming port. The last column of Table 5.4 illustrates the SBST pattern generation result which achieves 100% fault coverage at sequential depth three. Thus, testing all crosstalk faults in communication links of the switch requires 22 test patterns which are applied in 88 cycles.

The process time for test pattern generation may vary depending on the solver capabilities. In the experiments, we have used Sat4j [sat], a Java library for Boolean satisfiability. The method generates test patterns for stuck-at faults and bridging faults in less than six hours at $T = 3$. For larger T, as the size of the SAT instance increases, the average process time per fault increases as well. However, since many faults are already covered at smaller T, the overall process time reduces. The fault type has an impact on the processing time as well. For example, in the crosstalk and bridging fault modelling, an interdependency between two signal lines is defined as the activation condition of the fault. This makes the instance more complex as compared with the stuck-at fault.

5.6 Summary

The presented structural Software-Based Self-Test targets structural faults in the NoC switches and links. In order to apply test patterns, the processing elements surrounding the switch under test are reused for test generation and evaluation as well as test access. The SBST pattern generation is mapped to a Boolean satisfiability problem, and only valid NoC packets are accepted as satisfying assignments by introducing additional formulas to the SAT instance.

The proposed Software-Based Self-Test method is capable to generate test patterns targeting various kinds of structural faults in the switches and the links. Conducted

experiments for stuck-at, bridging and crosstalk faults confirm the efficiency of the method with high fault coverage and low test time. By accessing only the primary input and outputs of the switch, the test time decreases without imposing any hardware overhead. Besides, as test patterns are in the form of NoC packets, the test process can be performed in-field without putting the system in a non-functional test mode. This, in turn, eliminates the requirement to restore the system states after testing. High structural fault coverage and low testing overhead makes the method attractive for both manufacturing and in-field testing as well as structural diagnosis.

6

Concurrent Error Detection in NoCs

Switch-accurate network diagnosis can be conducted at start-up or on-demand by the proposed Software-Based Self-Test method as discussed in Chapter 5. Test patterns are applied to individual switches, and those which fail the test are marked as faulty. However, to shield the NoC against faults during the run time, concurrent error detection is necessary.

This chapter introduces a hybrid method to synthesize area-efficient fault-secure NoC switches. The fault-secure structure is able to detect all errors resulting from any single-point combinational or transition fault in switches or interconnect links.

6.1 Overview and Preliminaries

Concurrent error detection (CED) techniques are used in safety-critical applications to detect permanent and transient faults during the normal operation of the circuit [Nicolaidis98]. Conventionally, Duplication With Comparison (DWC) has been used to detect any single and multi-bit errors. In this method, the outputs of the original and duplicated circuit (not necessarily the same implementation) are compared with each other to signal an error when the outputs disagree. In order to decrease duplication overhead, Error Detecting Codes (EDC) are employed (see 3.3.3). The general structure for CED using EDC is illustrated in Fig. 6.1. The *checker* computes the check bits

over the circuit output bits and compares them against those generated by the *prediction logic* and throws an error in case of a mismatch. The circuit outputs and the check bits form the *codeword*.

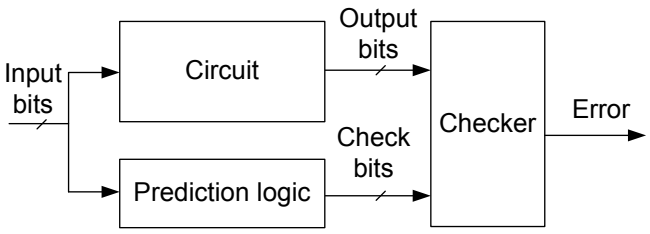


Figure 6.1: General structure of Concurrent Error Detection (CED)

As discussed in section 1.3.2, a circuit is defined Totally Self-Checking (TSC) when it is both self-testing and fault-secure. In this case, the checker itself needs to be self-testing [Lala01d]. A single output checker is insufficient for this purpose, because a stuck-at zero at the checker output is never detected. Moreover, faults within the circuit cannot be reported by the error indicator any more. To make self-testing checkers, dual-rail checkers are used to encode the error indication signal. A dual-rail checker with a two-input pair consists of two outputs as the error indicator. The values of error indicator outputs are complementary in case of any mismatch between the input pairs. A k-input pair dual-rail checker is built up by cascading $(k-1)$ two-input pair dual-rail checkers [Lala01d].

6.1.1 Fault-Secureness

For a set of faults F, a circuit is fault-secure *iff*:

$$\forall f \in F\,,\, \forall i \in I^n : \quad C(i) = C^f(i) \;\vee\; \Pi(C(i)) \neq \Pi(C^f(i))\,, \tag{6.1}$$

where I^n is the set of possible input vectors of n bits, $C(i)$ denotes the circuit response for input vector i in the fault-free case, and $C^f(i)$ is the response under fault f. Moreover, Π is a function that computes the check bits over the circuit response for the selected error detecting code. The formula expresses that for *any* input vector, the fault either is not propagated to the circuit outputs, i.e. $C(i) = C^f(i)$, or it is detected by

the check bits, i.e. $\Pi(C(i)) \neq \Pi(C^f(i))$. If a fault propagates to the circuit outputs but it is not detectable by the check bits, *Silent Data Corruption (SDC)* occurs.

For a non-redundant fault-secure circuit, the self-testing property is implicit, because there exists at least one input vector for every testable fault that makes it observable at the circuit outputs. Moreover, the fault-secure property ensures generating a non-codeword output for that.

6.2 Fault-Secure Synthesis of NoC Switches

The fault-secure synthesis relies on error detecting codes at flit-level to reduce the incurred hardware overhead, and introduces extra parity bits only to cover those faults which violate fault-secureness.

6.2.1 Error Detection at Flit-Level

In the fault-secure switch, each flit is converted to a codeword by appending a number of check bits to its content. For instance, when the parity is selected as the error detecting code (EDC), the parity bit is added to the flit content as the check bit. According to the predefined EDC, check bits are produced by the sender once a packet is injected into the network.

As depicted in Fig. 6.2 flit checkers (*Flit chk*) are positioned at the outgoing ports of the switches. The flit checker comprises an encoder, implementing Π as EDC function, and a comparator. Upon passing a flit through the flit checker, the encoder recomputes the check bits which are compared against those stored in the flit. A mismatch causes the assertion of the error signal. The flit checkers are necessary while pursuing fault-secureness. An end-to-end error control scheme similar to [Rossi07] can check the correctness of the transported data once in the destination, and therefore omits the flit checkers. However, such schemes do not assure fault-secureness of the switches, because a fault effect could be masked while passing through other switches.

As the flit checkers are positioned only in the outgoing ports, faults in a link are

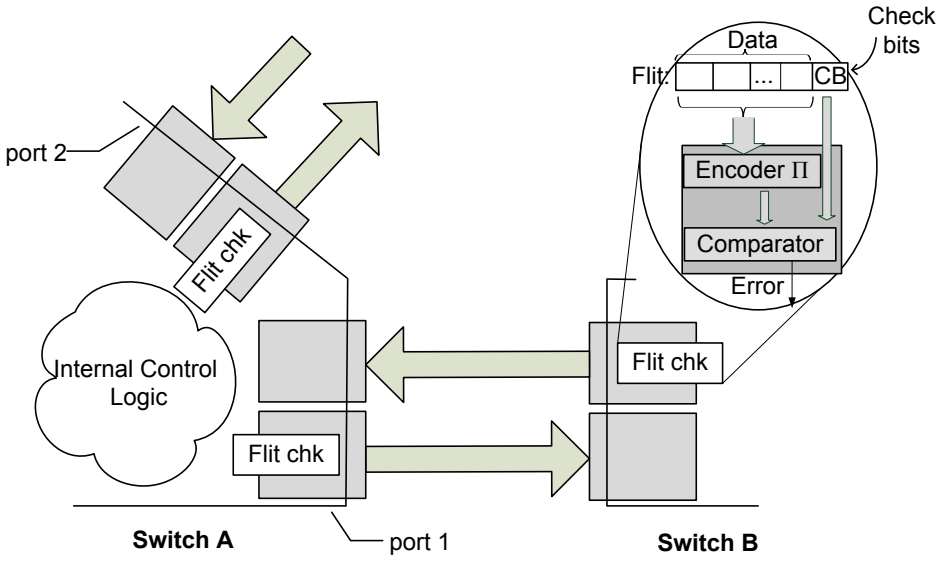


Figure 6.2: Flit checkers: position and structure

recognised upon passing a data through the downstream switch. This is sufficient for the switch-accurate diagnosis. However, since the flit checkers are constructed over a subset of switch outputs, i.e. data outputs, they are not sufficient to assure the fault-secureness of the switch [Augustin10]. Depending on the EDC, a subset of faults in the incoming data links and in the switch datapath elements are detectable by the flit checkers. Some faults may propagate either to the internal states of the switch or to other outputs rather than the data outputs. Moreover, certain faults may cause an error on the data outputs that is not detectable by the error detecting code. This is elaborated in the following example.

Example 6.2.1. Let us consider a 2-to-1 multiplexer as part of the crossbar in the switch datapath and assume a single parity bit is added to the flit as the check bit. In the 2-to-1 multiplexer of Fig. 6.3, $z = (\bar{s} \wedge x) \vee (s \wedge y)$. Therefore, for the function Π that computes the parity check bit of the flit, we have $\Pi(z) = \Pi(x)$ when s is zero, and $\Pi(z) = \Pi(y)$ when s is one.

Assume that there exists a stuck-at-1 fault at line a of the multiplexer as shown in Fig. 6.3. In this case, the multiplexer function for every bit i changes to: $z_i = x_i \vee (s \wedge y_i)$. Thus, when s is zero, z carries the value of x, and the fault is not detectable. However, when $s = 1$, z carries a bit-wise OR of the data on the lines x and y ($z = x \vee$

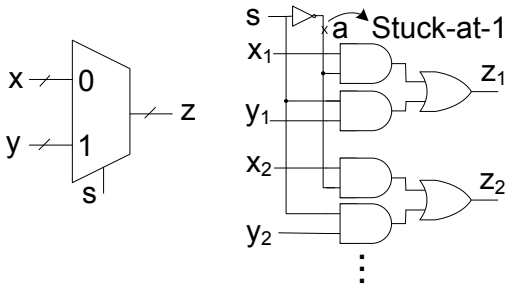


Figure 6.3: 2-to-1 multiplexer: Stuck-at-1 fault on line a violates fault-secureness

y), and therefore $\Pi(z) = \Pi(x \vee y)$. In this case, erroneous data on z is not necessarily detected by the parity bit. One may specify certain values for x (e.g. $x = 0$) such that the fault becomes detectable. Nonetheless, since the fault effect is unpredictable, it is hard (impossible) to design an error detection mechanism based on error detecting codes, even at the higher network layers, such that the fault effect is detected for *all* input vectors, i.e. fault-secureness cannot be guaranteed.

6.2.2 Synthesis Flow

Figure 6.4 depicts the overall flow of the synthesis scheme. Initially, the switch including the flit checkers is analysed to identify which faults propagate to data outputs and are detectable by flit checkers under *any* given input vector (details in section 6.3). These faults are named *easy-to-detect faults*, or briefly *easy faults*. The rest of the faults are categorized as *hard-to-detect faults*, or briefly *hard faults*.

With respect to the hard faults, a sub-block of the circuit is extracted, which is called *critical region*. It includes only the parts that influence the propagation of errors produced by the hard faults. A CED structure is synthesized to make the critical region fault-secure (details in section 6.4). The method concentrates on only the hard faults in order to reduce the area overhead.

One error signal is generated per switch by the disjunction over the error signals of the flit checkers and the error signal of the fault-secure critical region. The error signal is sent to the neighbouring switches and the local core via a dedicated error port in order to invoke an error recovery mechanism. The resulting switch, including

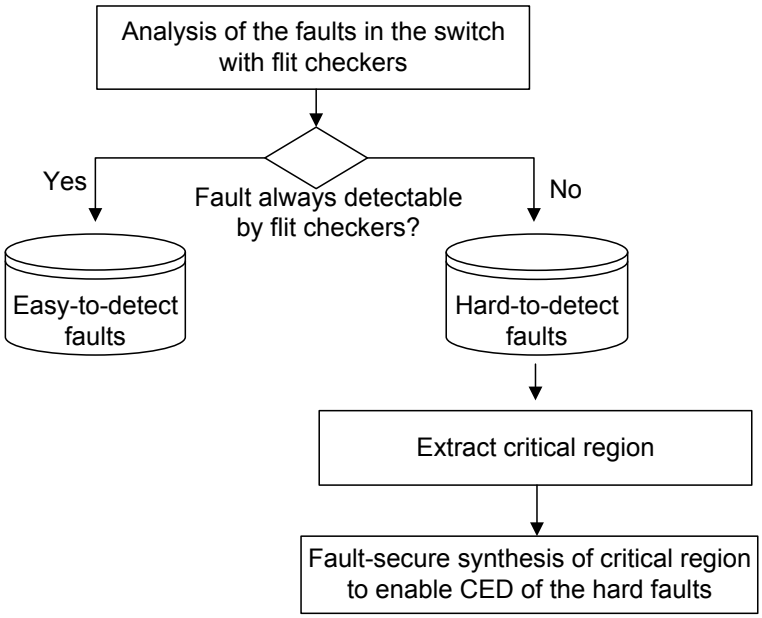


Figure 6.4: Overall flow of the synthesis scheme

flit checkers and CED circuitry of the critical region is fault-secure. According to the definition, if the original switch is non-redundant, it is also totally self-checking.

We use the Boolean satisfiability solver (see the preliminaries in section 2.5) to identify the hard faults and further instruct the construction of an error detecting code for fault-secure synthesis.

6.3 Identification of Hard Faults

The switch with flit checkers is fault-secure, if the effect of the faults that propagate to the switch outputs is never masked at flit checkers under any input vector, i.e. the faults never cause a Silent Data Corruption (SDC). Proving that a fault f never causes SDC is an NP-complete problem. Exhaustive fault simulation of all faults and all input vectors (like the error detectability table in [Almukhaizim06]) is not feasible due to the huge input vector space, I^n, where:

$$n = \text{num. of switch ports} \times (\text{flit size} + \text{num. of handshake signals}). \tag{6.2}$$

However, to identify the faults that cause silent data corruption (i.e. hard faults), it is sufficient to find one input vector ($i \in I^n$) for which the fault effect propagates to the switch outputs and is masked at flit checkers. Formally, in a switch with m ports, fault f causes silent data corruption *iff*:

$$\exists i \in I^n : C(i) \neq C^f(i) \wedge \bigwedge_{0 \leq j < m} \Pi(D_j(i)) = \Pi(D_j^f(i)) \,, \tag{6.3}$$

where $C(i)$ is the switch response for the input vector i, and $C^f(i)$ is the switch response under fault f. Moreover, $D_j(i)$ denotes the data outputs of port j in the fault-free case, and $D_j^f(i)$ stands for the data outputs under fault f, while Π computes the check bits over the data bits of each port.

A directed search for one input vector that causes silent data corruption under fault f, or the proof that such a vector does not exist, can be mapped to a test pattern generation (ATPG) problem [Hunger08]. In [Fey08], it is shown that such a check can be efficiently conducted using Boolean satisfiability (SAT) solvers. We map the problem of identifying hard faults onto a SAT instance as illustrated in Fig. 6.5. The outputs of the combinational switch S and the faulty copy S^f, including primary and pseudo primary outputs, are compared bit-wise (block ①). The encoders Π compute the check bits over the data outputs D_0 to D_m, which are then compared to those in the faulty instance (block ②). The instance is satisfiable if and only if statement (6.3) is true.

The corresponding Boolean formula for the SAT instance of Fig. 6.5 in Conjunctive Normal Form (CNF) is defined as follows:

$$\begin{aligned} \Phi^f \;=\; & \mathrm{CNF}(S) \wedge \mathrm{CNF}(S^f) \wedge \mathrm{CNF}(C \overset{*}{\nLeftrightarrow} C^f) \wedge \\ & \mathrm{CNF}(\Pi) \wedge \mathrm{CNF}(\Pi^f) \wedge \bigwedge_{0 \leq j < m} \mathrm{CNF}(\Pi_{D_j} \overset{*}{\Leftrightarrow} \Pi_{D_j}^f) \,. \end{aligned} \tag{6.4}$$

The instance Φ^f comprises the characteristic function of the gates in the switch $\mathrm{CNF}(S)$ and the faulty copy $\mathrm{CNF}(S^f)$. According to definition 2.5.2, $\mathrm{CNF}(S)$ is the conjunction of the CNF formulas of all logic gates in S, while $\mathrm{CNF}(S^f)$ models only the gates in the output cone of fault f. The term $\mathrm{CNF}(C \overset{*}{\nLeftrightarrow} C^f)$ stands for the CNF formula of nonequivalent literal sets (see definition A.1.6 in appendix A.1) and

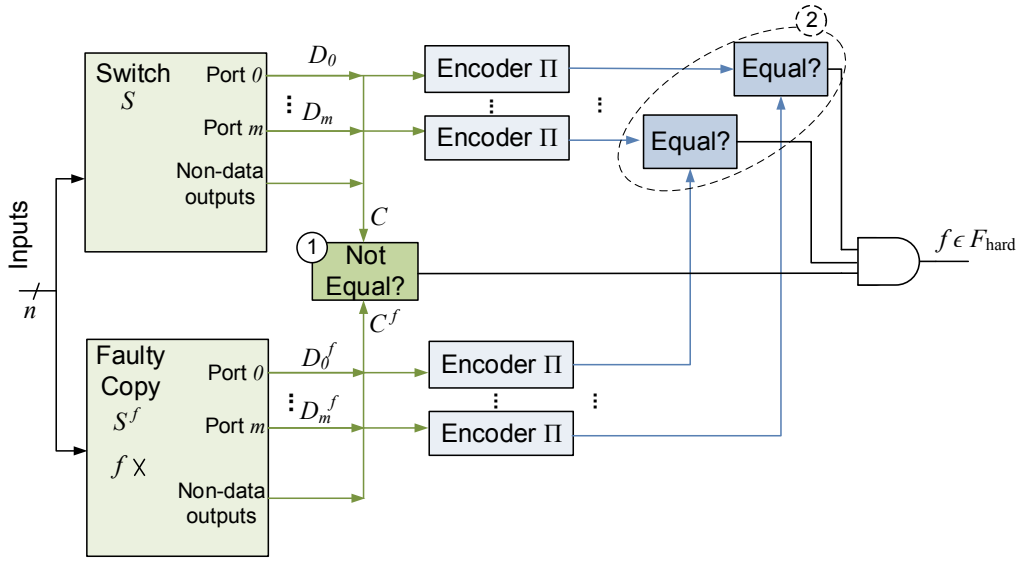


Figure 6.5: Schematic of the SAT instance to identify hard faults

ensures that at least one output of the switch in the good copy differs from the faulty copy. $\mathrm{CNF}(\Pi)$ and $\mathrm{CNF}(\Pi^f)$ denote the CNF formulas for the encoder structures over the data outputs while Π_{D_j} and $\Pi^f_{D_j}$ represent the set of literals for the encoder outputs of each switch port j in the good and faulty copy, respectively. $\mathrm{CNF}(\Pi_{D_j} \overset{*}{\Leftrightarrow} \Pi^f_{D_j})$ denotes the CNF formula of two equivalent literal sets (see definition A.1.3 in appendix A.1) and ensures that the encoder outputs are identical.

For each fault, a new instance Φ^f is constructed. Φ^f is satisfiable if there exists one input vector such that the switch outputs differ in the good and faulty copy while the encoder outputs are equal. In this case, the fault is a hard fault which is not detectable by the flit checkers for (at least) the satisfying input assignment found by the solver.

Once the hard faults are identified, the synthesis scheme searches for extra parity trees, which is explained in the following section.

6.4 Fault-Secure Synthesis for Hard Faults

The support of a signal, $\mathrm{support}(f)$, is the union of the gates in the input cones of all outputs reachable from f [Hamzaoglu99]. We define the critical region C of the switch

as the part of the logic that may influence the propagation of hard faults ($f \in F_{\text{hard}}$) toward the circuit outputs:

$$C := \bigcup_{f \in F_{\text{hard}}} \text{support}(f). \tag{6.5}$$

The fault-secure synthesis of the critical region must ensure that hard faults do not violate fault-secureness by Silent Data Corruption (SDC). One conventional approach to achieve this is the duplication with comparison. However, as discussed previously in section 3.3.3, fault-secure synthesis using error detecting codes may reduce the incurred area overhead.

Here, we present a novel approach for synthesizing the prediction and checker logic based on multi-bit parity code. Since the proposed method reduces the number of required parity groups, the complexity of the parity prediction and checker logic decreases. Furthermore, by taking advantage of the satisfiability solver in searching for silent data corruptions, the method is applicable even for larger circuits.

The rest of this section explains the synthesis flow for the critical region of the switch. It starts with a topological analysis to construct initial parity groups. A SAT instance is then created to identify the faults in F_{hard} that still cause SDC. Lastly, the initial parity groups are split iteratively until all cases of SDC are resolved.

6.4.1 Topological Analysis

The topological analysis of the critical region of the switch considers the hard faults only. Moreover, since by reducing the number of parity check bits, the area overhead of the parity prediction and checker logic decreases, , the initial goal is to minimize the number of parity groups. Toward this end, the initial parity groups are constructed such that each hard fault effect is not masked in at least one parity group.

Definition 6.4.1. *(dependent and independent outputs)* Two circuit outputs are *dependent* if logic is shared in their input cones, and *independent* otherwise.

Definition 6.4.2. *(output dependency graph)* Graph $G = (V, E)$ is constructed by introducing a vertex $v \in V$ for every circuit output. An edge $e \in E$ is placed between

pairs of outputs which are dependent. The graph illustrates all dependencies among the circuit outputs.

Definition 6.4.3. *(covered gate)* A gate g is *covered* by a group or subgraph G_i, when it is in the input cone of only one output in that group. In this case, any combinational fault at g is propagated to at most one output and therefore it is not masked in the parity tree over the outputs in G_i. Consequently, $\mathrm{Cov}(g)$ is defined as follows:

$$\mathrm{Cov}(g) = \begin{cases} 0 & g \text{ is not covered in any group.} \\ 1 & g \text{ is covered in at least one group } G_i. \end{cases}$$

Definition 6.4.4. *(partitioning P_k)* A partitioning $P_k = \{G_i = (V_i, E_i) \mid 1 \leq i \leq k\}$ of an output dependency graph G with subgraphs G_i and $\bigcup_i V_i = V$, corresponds to k parity groups over the outputs so that logic sharing of outputs within a single group could be allowed as long as the fault effects are propagated to an odd number of outputs covered by another parity group.

Definition 6.4.5. *(coverage of a partitioning P_k)* In a circuit with l logic gates of hard fault sites, the coverage of a partitioning P_k is defined as the number of covered gates as follows:

$$\mathrm{Coverage}(P_k) = \sum_{i=1}^{l} \mathrm{Cov}(g_i). \tag{6.6}$$

In graph theory, an independent set is defined as a set of vertices in a graph, no two of which are connected with an edge [Golumbic04]. Moreover, an independent set is maximal, when it is not a subset of any other independent set. According to the above definitions, the outputs that can be placed in the same parity group without any risk of masking the errors, are independent and share no logic. Moreover, they form an independent set of vertices in the output dependency graph G.

In a circuit with l logic gates of the hard fault sites, topological analysis searches for a partitioning P_k such that $\mathrm{Coverage}(P_k) = l$ with minimized k. The procedure consists of: (A) computing a pre-partitioning constructed from independent sets of outputs in G, and (B) iteratively creating additional groups and distributing yet unpartitioned vertices among them.

Pre-Partitioning of Circuit Outputs

If some outputs are dependent, the partitioning needs at least two groups of outputs: $P_2 = \{G_1, G_2\}$, where G_1 and G_2 are two independent sets of outputs in G which maximize $\mathrm{Coverage}(P_2)$.

Firstly, the maximal independent sets for every pair of the independent outputs are computed. Compared to the problem of listing all maximal independent sets [Tomita06], this constraint limits the runtime complexity to $\mathcal{O}(|V|^3)$.

For $\mathrm{Cone}(v)$ being the set of gates in the input cone of output v, and G_i being an independent set of outputs, we compute the set of gates in the input cones of all outputs in the group G_i as follows:

$$\mathrm{Cone}(G_i) = \bigcup_{v \in V_i} \mathrm{Cone}(v). \tag{6.7}$$

As the outputs in G_i are independent, the number of covered gates in G_i is equal to $|\mathrm{Cone}(G_i)|$. The independent set with maximum $|\mathrm{Cone}(G_i)|$ is assigned to the first group of outputs.

The second group is similarly chosen from the list of independent sets of outputs, excluding the outputs in the first group. This increases the flexibility for adding the remaining unpartitioned vertices to the available groups.

Example 6.4.1. Figure 6.6 depicts a circuit with four outputs, v_1 to v_4, and their input cones shaped by triangle. The numbers written in each polygon indicate the number of gates in that part. Accordingly, output v_1 has seven gates in its input cone, i.e. $|\mathrm{Cone}(v_1)| = 7$. Graph G summarizes the logic sharing information among the vertices. Next to each vertex, the number of gates in its input cone is depicted. The maximal independent sets of outputs are as follows:

$$\begin{aligned} G_1 &= \{v_1, v_4\} \quad |\mathrm{Cone}(G_1)| = 13 \\ G_2 &= \{v_2, v_4\} \quad |\mathrm{Cone}(G_2)| = 15 \\ G_3 &= \{v_3\} \qquad |\mathrm{Cone}(G_3)| = 8 \end{aligned}$$

The pre-partitioning chooses G_2 with the maximum covered gates as the first group of

outputs. Excluding the outputs in G_2, the number of covered gates will be 7 for G_1 and 8 for G_3. So, G_3 is selected as the second group, i.e. $P_2 = \{G_2, G_3\}$.

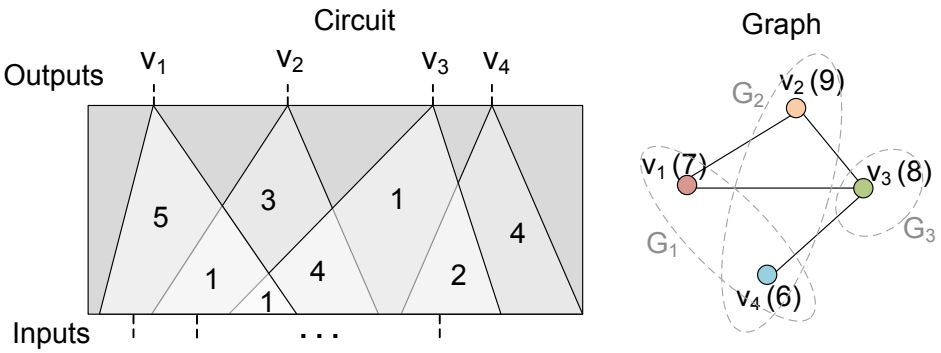


Figure 6.6: Graph construction and pre-partitioning example

Distribution of Unpartitioned Outputs

The yet unpartitioned outputs are assigned either to the so far established groups (two groups right after pre-partitioning), or to new groups if it is not possible to cover a gate in one of the existing groups. All unpartitioned outputs share logic with the outputs that have been partitioned during pre-partitioning. However, according to definition 6.4.4, we can assign an output v to a group G_i, if all gates in the shared logic between $\mathrm{Cone}(v)$ and $\mathrm{Cone}(G_i)$ are covered in another group.

Example 6.4.2. In Fig. 6.7, the output v_5 shares logic with the input cone of v_3 in the striped area *S*. But, v_5 is still assigned to group G_1, because the shared logic is part of the input cone of v_2 and has been already covered by G_2. The output v_5 cannot be added to G_2 because it shares some logic with v_2 which is not covered in any other group.

To determine the order in which the outputs are processed, we compute for each unassigned output its weight as:

$$w(v) = \max_{G_i \in P_k} \Delta\mathrm{Coverage}(v, G_i), \tag{6.8}$$

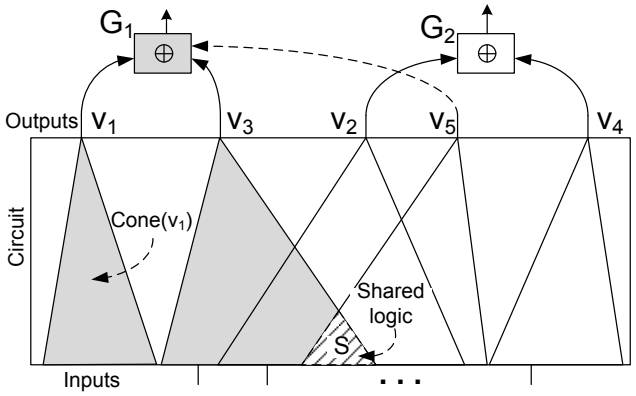


Figure 6.7: Distribution of unpartitioned outputs

where $\Delta\mathrm{Coverage}(v, G_i)$ is the number of additionally covered gates when adding v to G_i. The output with highest weight is assigned first.

The distribution of outputs among established groups continues until no further assignment to available groups is possible. Then, a new group is created, which is chosen among the maximal independent sets of outputs, while excluding already partitioned vertices. Topological analysis creates a small number of parity groups as a starting point for resolving all silent data corruptions.

6.4.2 Resolving Silent Data Corruption

The topological analysis cannot guarantee fault-secureness yet, since it only ensures that a fault effect is not masked in at least *one* parity group. There may be input vectors which propagate the fault effect *only* to the groups where error masking is not inhibited. However, topological analysis reduces the runtime of the formal analysis by decreasing the probability of such silent data corruptions (SDC).

Further splitting of parity groups until no logic is shared between the outputs in the same group would lead to a solution similar to [Sogomonyan93]. In our scheme, we perform firstly a formal analysis to check which faults cause SDC and then selectively split the parity groups to resolve SDCs.

To find faults that cause SDC, we add the parity trees of the critical region to the

switch including the flit checkers and extend the SAT instance Φ^f, constructed to find the hard faults (see section 6.3). The extended SAT instance Φ^f_{SDC} is a conjunction of Φ^f, the clauses for parity trees over the outputs of the critical region and the clauses for comparison as follows:

$$\Phi^f_{\mathrm{SDC}} = \Phi^f \wedge \mathrm{CNF}(P_V) \wedge \mathrm{CNF}(P_{Vf}) \wedge \bigwedge_{1 \le j \le k} \mathrm{CNF}(p_j \Leftrightarrow p^f_j)\,. \tag{6.9}$$

$\mathrm{CNF}(P_V)$ and $\mathrm{CNF}(P_{Vf})$ stand for the clauses of the parity trees over the outputs of the critical region in the good and faulty copy, respectively. p_js are the parity check bits in the good copy which are compared against those in the faulty copy, and k is the number of parity groups. In Eq. (6.9), $\mathrm{CNF}(p_j \Leftrightarrow p^f_j)$ is the CNF formula of two equivalent literals (see definition A.1.1 in appendix A.1), that ensures p_j and p^f_j get the same values in the satisfying instance.

The SAT instance Φ^f_{SDC} is satisfiable if there exists one satisfying assignment such that the fault is observable at switch outputs ($C \neq C^f$, defined in Φ^f) while it is not detectable by neither the flit checkers nor the parity check bits of the critical region. In this case, the satisfying assignment is an input vector which causes SDC. The group splitting algorithm refines the existing groups to reduce the number of faults with SDC effect. Details of the group splitting algorithm come in the following section.

Group Splitting

Figure 6.8 illustrates the steps for group splitting. After topological analysis, the circuit outputs have been distributed among k groups. Let F_{SDC} denote the faults which cause silent data corruption (SDC). For each fault $f \in F_{\mathrm{SDC}}$, the SAT solver computes one input vector i_f causing SDC (although i_f may not be the only vector causing SDC under f). We simulate all faults $f \in F_{\mathrm{SDC}}$ with the input vector i_f to find the outputs and parity groups to which the fault effect propagates. To avoid SDC of a fault f under i_f in this group, the group must be split into at least two groups.

Let V^{Gi}_f be the subset of outputs in group G_i to which the fault effect of f is propagated under input vector i_f. There are $2^{(|V^{Gi}_f|-1)} - 1$ possibilities of splitting G_i. For each possibility, we compute the number of faults for which SDC is avoided by

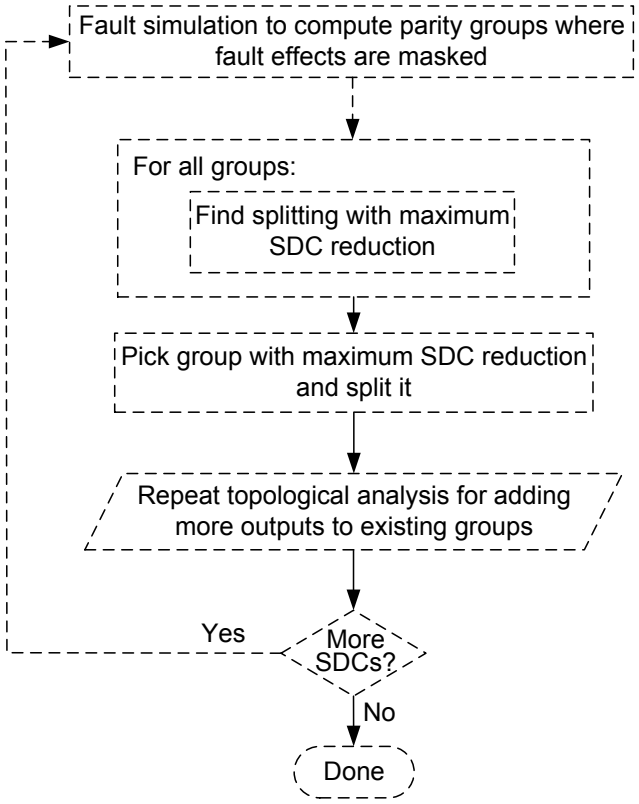


Figure 6.8: Steps of group splitting

the splitting. The number of avoidable SDCs is computed for all groups to which a fault propagates. The group and splitting with the highest reduction in SDC faults is selected and the parity group is modified. A topological analysis is performed after each splitting to investigate the possibility of adding some outputs to the available groups and avoiding SDCs that way.

For large $|V_f^{Gi}|$, the explicit analysis of each splitting possibility is computationally too expensive. In this case, G_i is divided into two groups, one of them is the maximal independent set of G_i, and the other has the minimum logic sharing in its input cones.

The process of SDC analysis and group splitting proceeds until no more faults cause silent data corruption. The parity prediction and checker logic are synthesized separately from the functional circuit. The resulting circuit is fault-secure. For a non-redundant circuit, the resulting circuit is even totally self-checking, since the proposed algorithm does not modify the original circuit and all errors are detected.

Complexity of the Algorithm

Synthesis of an optimal fault-secure design is inherently a complex process. On the one side, the fault-secure property must be examined for all faults, F, in the circuit. Moreover, for every fault, the entire input space (I^n) must be checked to ensure it never violates fault-secureness. In fact, the variable space of the problem is in the size of $F \times I^n$. On the other side, finding the optimum number of parity groups to distribute the outputs among them is an NP-complete problem similar to the graph's k-partitioning problems [Andreev06, Garey90].

The topological analysis reduces the number of faults to be processed during the splitting. Our investigation on a set of benchmark circuits (see Appendix C) shows that after topological analysis in the worst case, only around 35% of the faults cause silent data corruption. This drastically shortens the execution time for the splitting. Moreover, for the faults which propagate to the same set of outputs, the splitting processes the output set once and the result is reused for all faults with the same effect.

The complexity of the splitting algorithm could be still too high, as the number of splitting possibilities for each group is of exponential size. For bigger circuits, there might be a large number of outputs in the same group which are affected by a fault. To solve the problem, an upper bound for the size of $V_f^{G_i}$ (outputs which are affected by the same fault f) is defined. When the size of V_f is greater than the upper bound, the outputs are divided into two groups by topological analysis and without looking for the optimum splitting.

6.5 Experimental Results

The fault-secure synthesis scheme is evaluated for the typical NoC switch described in section 2.1.1. The considered switch of the 2D mesh topology, consists of five ports and is adjusted according to the desired flit width. The switch is synthesized using Synopsys Design Compiler. The target library lsi10k is constrained to basic gate primitives. Because memory elements are usually equipped with advanced BIST/BISR features [Zhang06], the storage elements of incoming ports are not considered in the

fault-secure synthesis flow. Still, the error detecting code (EDC) of the flits partially protects the memory elements. The memory elements are also excluded from the area computations.

In the rest of this section, different error detecting codes (EDCs) for data encoding at flit-level are evaluated and compared in terms of fault coverage and area overhead. The result of the fault-secure synthesis is then presented. Finally, we discuss the impact of the proposed fault-secure structure on test data compaction and diagnosis.

6.5.1 Evaluation of Flit Checkers

The error detecting code used for flit checkers influences both the number of easy-to-detect faults at flit-level and the area overhead of the fault-secure structure. A single-bit parity code needs only one check bit per flit and because of the simple encoder structure, it imposes a small hardware overhead. It detects single and multi-bit errors of odd multiplicity. More complex EDCs such as cyclic codes are better suited to detect burst errors but impose a larger area overhead. A multi-cycle implementation of the encoder is an alternative to reduce the hardware overhead, but the performance is compromised.

The single bit parity, Berger [Lala01c], Hamming (single error correcting/double error detecting), Hsiao [Hsiao70], and Cyclic(4) (cyclic redundancy code with the generator polynomial of degree 4) are implemented as the EDC at the flit checkers with a flit width of eight bits. The SAT instance Φ^f is constructed for each code to compute the number of easy/hard faults. The cost of the flit checkers is computed as the percentage of the area of the flit checkers with respect to the area of the switch as follows:

$$\text{Flit checker cost} = \frac{\text{flit checkers area}}{\text{switch area}} \times 100\% \,.$$

Figure 6.9 compares the codes in terms of the flit checker cost and fault coverage. In all cases, around 60% of faults are recognized as hard faults, which means they are not detectable by flit checkers for at least one input vector. A structural analysis shows that 6931 faults (56.85%) out of 12191 collapsed faults in the switch are not propagated to data outputs. Therefore, they are in principle not detectable by flit checkers. Among

the remaining 5260 faults, only those which are detectable by the flit checkers belong to the easy faults. In order to evaluate the detectability of EDCs in flit checkers, parameter γ is defined as:

$$\gamma = \frac{\text{number of easy faults}}{\text{number of faults propagated to data outputs}} \times 100\%.$$

γ quantifies the portion of faults that are detectable by flit checkers over the number of faults which are propagated to the data outputs. As it is observed in Fig. 6.9, the Berger code has the highest detectability of 99.38%, while the other codes have a detectability of approximately 87%. However, the encoder of the Berger code requires the highest area overhead of 15.54% of the switch area. The Berger is capable of detecting burst unidirectional errors. For example, it can provide fault-secureness for a defect scenario similar to what we explained in example 6.2.1. The difference between the γ value of the Berger and the other codes indicates that almost 11% of the faults that propagate to the data outputs produce burst errors.

To evaluate the flit checkers for switches with wider flits, we select the parity as the EDC and compare the detectability of the flit checkers among the switches with 16,

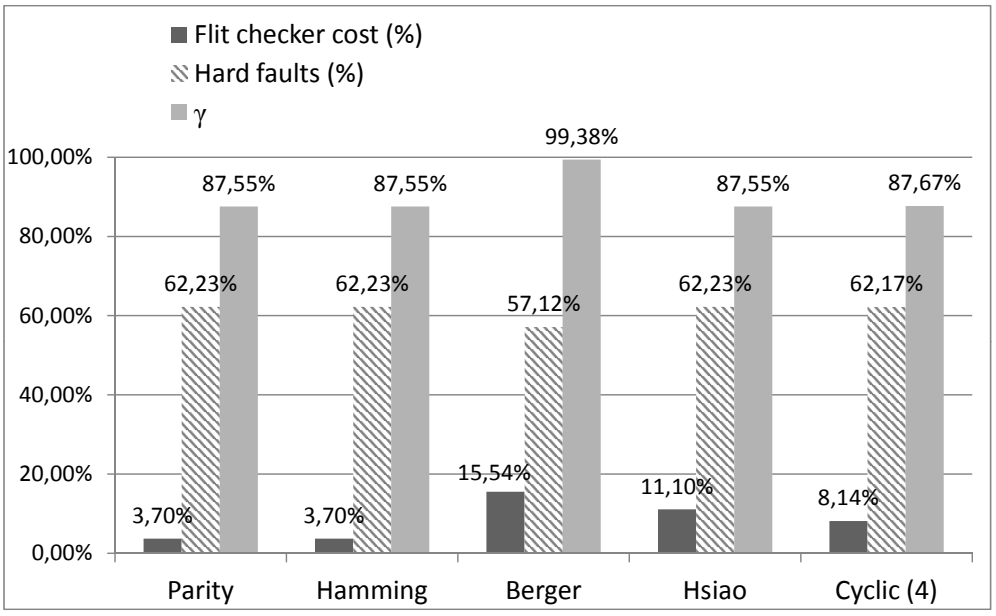


Figure 6.9: Error detecting codes: flit checker cost - hard faults - γ

32, 64, and 128 bits flit width. Results are illustrated in Table 6.1. The second column reports the cell area of each switch next to the flit width. The third and fourth column respectively report the number of collapsed faults and the number of hard faults in each switch. The fifth column shows the number of faults which are propagated to data outputs and the next column reports γ.

Table 6.1: Flit checker comparison for different flit width

Flit Width	Cell Area	# Collapsed Faults	Hard Faults	Faults propagated to data outputs	γ (%)	Time (s)
16-bit	7259	18901	10261 (54.29%)	9580 (50.69%)	90.19	784
32-bit	11979	32341	15061 (46.57%)	18220 (56.34%)	94.84	1924
64-bit	21419	59221	24661 (41.64%)	35500 (59.94%)	97.35	7266
128-bit	39017	112991	43871 (38.83%)	70060 (62.00%)	98.66	42963

The first observation is that the larger the flit width is, the portion of the hard faults is less. When the flit width increases, the width of datapath elements such as crossbar multiplexers increases as well and therefore more faults become observable at the data outputs (see column five of Table 6.1). Moreover, the fraction of faults detected by the flit checkers is higher as indicated by the parameter γ. In fact, the number of faults which are not detectable by flit checkers remained unchanged (940 faults). As the controlling logic remains unchanged when adjusting the flit width, we can conclude that the undetectable faults with an erroneous effect on the data outputs, are located at the controlling logic, for example multiplexer select signals.

The last column of Table 6.1 reports the time required to compute the hard faults and construct the corresponding critical region. To be compatible with our Java-based internal DfT framework, we have used Sat4j [sat], the Java library for Boolean satisfiability. As it is expected, the execution time is longer for larger switches, because more faults propagate to the data outputs and SAT must trace them searching for silent data corruption. But the time does not grow linearly, because the size of the switch, which influences the complexity of the SAT instance, increases as well. Nevertheless, the method can still generate the fault-secure switch in a reasonable time. The runtime can be reduced by employing other efficient solvers, which is not in the scope of this dissertation. For example, a C++ SAT solver is about 3.25 times faster than its counterpart in Java [sat].

6.5.2 Result of Fault-Secure Synthesis

Considering the parity as the EDC in flit checkers and based on the list of the hard faults, the critical region is constructed and the synthesis process is performed to enable concurrent error detection of the hard faults. Table 6.2 illustrates the result of fault-secure synthesis of the 8-bit switch for various synthesis techniques and compares them in terms of the number of parity groups (*# groups*), the number of faults with the silent data corruption effect (*# SDCs*) and the *cost* of synthesis technique. The cost reflects the hardware overhead of the fault-secure technique and is computed as follows:

$$\text{cost} = \frac{\text{additional area for fault-secureness}}{\text{switch area}} \times 100\% \,.$$

As shown in the first row of Table 6.2, by constructing only the flit checkers with 3.7% hardware cost, 7871 faults still cause SDC, i.e. the design is not fault-secure. Following the proposed synthesis flow, by constructing 43 extra parity groups over the outputs of the critical region, fault-secureness is ensured (i.e. no SDCs). The resulting fault-secure switch imposes 52% hardware cost. The overhead of fault-secure technique here includes the flit checkers, the prediction logic for the critical region and the dual rail checkers. Any single transient and permanent fault in the switch as well as the checker is detectable by this fault-secure structure.

Table 6.2: Result of fault-secure synthesis

Synthesis technique	# groups	# SDCs	cost
Switch + flit checkers	5	7871	3.70%
Proposed fault-secure switch	*48*	*0*	*52.01%*
Duplicated switch	-	0	371.29%
Duplicated critical region	-	0	89.09%

As shown in the third row of Table 6.2, duplication with comparison for the entire switch with 1040 primary and pseudo primary outputs imposes a huge hardware cost of more than 370% due to the large number of outputs and the overhead of dual rail checkers. Even without dual rail checkers duplication imposes an area overhead of 204%. For the larger switches with even more outputs, the cost of duplication will increase excessively.

The critical region occupies 41% of the switch area and contains 185 outputs. After extracting the critical region according to the proposed technique, one may exploit duplication with comparison to achieve fault-secureness of the critical region. In this case, the cost of the fault-secure synthesis will be 89% as reported in the last row of Table 6.2. Compared to duplication with comparison for the critical region, the presented fault-secure switch using multiple parity bits saves almost 37% area cost. Furthermore, the proposed synthesis technique significantly reduces the area cost compared to the conventional duplication with comparison for the entire switch.

The fault-secure synthesis scheme for critical region can be applied for arbitrary logic circuits targeting all single-point faults. The technique has been applied to a set of MCNC, ISCAS and ITC benchmark circuits and the result is covered in Appendix C. Also for the benchmark circuits, the presented fault-secure synthesis scheme has reduced the hardware overhead as compared to the previous self-checking approaches. Moreover, for the first time, efficient fault-secureness is achieved for circuits with up to 15K gates.

6.5.3 Structure Reuse for Test Compaction and Diagnosis

Parity trees of the fault-secure switch can be reused for test response compaction [Holst09] that reduces the storage and bandwidth requirements for manufacturing and in-field testing. The common approach for test compaction is that the data outputs of several scan chains are compacted into a single parity bit. However, test data compaction is only acceptable if the fault coverage and diagnosis resolution are maintained.

The fault-secure property ensures that every observable error at the circuit outputs, is detectable by the parity code. Therefore, by using parity trees of the fault-secure structure for test response compaction all testable faults remain testable and the fault coverage is not diminished.

The scan chain of the switch is restructured as depicted in Fig. 6.10, such that in each shift-out cycle, the output bits of one parity group are compacted. Therefore, to compact and shift-out the responses of k parity groups, k cycles are required. The number of scan chains is equal to the size of the largest parity tree. The scan flip-flops

of the outputs encoded at the flit checkers are also included in the scan chains.

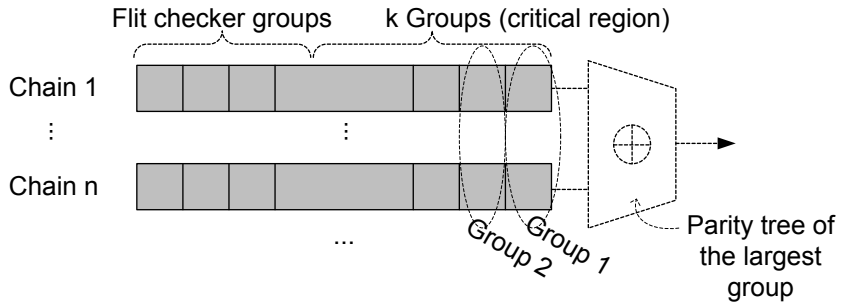


Figure 6.10: Scan restructuring for test compaction

We have used a commercial ATPG tool to generate test patterns for the original switch and the fault-secure switch. The test pattern generation result is depicted in Table 6.3. Although ATPG has generated a couple of more patterns for the fault-secure switch, the test response volume has been reduced drastically. With only 9% of test response data 100% stuck-at fault coverage is achieved in the fault-secure switch.

Table 6.3: Comparison of test and diagnosis results

	Switch	Fault-secure switch
# Test patterns	372	394
Fault coverage (%)	100	100
Test response volume (bit)	386880	34672
Diagnostic success (%)	98.76	98.50

The compacted test responses can be used to conduct the structural diagnosis of the switch [Dalirsani11] only when the diagnosis resolution is preserved. Using the algorithm in [Holst09], diagnosis is performed for the switch and the switch with parity trees. The last row of Table 6.3 compares the diagnostic success of the uncompacted response of the original switch with the diagnosis of the parity streams in the fault-secure switch targeting stuck-at faults. According to [Holst09], the diagnosis success indicates that how often the top-ranked fault suspect corresponds to the real defect. It is observed that even with the compacted test response, the diagnostic success rate is approximately the same as the uncompacted response. Indeed, without loss of fault coverage and diagnostic success, the parity trees of the fault-secure switch can be used

to reduce the amount of response data as well as the response shifting cycles by a factor of 11x.

6.6 Summary

This chapter presented a novel method to synthesize the fault-secure NoC switch employing data encoding at flit-level and concurrent error detection with multiple parity trees. A Boolean satisfiability-based algorithm identifies faults that cause silent data corruption and instructs the construction of an error detecting code.

The fault-secure NoC switch structure is an area-efficient scheme for error detection in-field. By means of a dedicated error signal for individual switches, the location of the faulty switch in the network is identified. Here, network diagnosis is switch-accurate as any detectable fault in the switch and its links is represented by a faulty switch in the abstracted fault model. The experimental results show significant area savings for implementing fault-secureness. In addition, the resulting structure is well suited to be reused for test compaction. This reduces the amount of test response data and the test time without loss of fault coverage and diagnostic resolution.

7

Simulation Results for Performability Measurement

In this chapter, we compare the performability of degraded NoCs in the presence of switch-accurate and port-accurate diagnosis schemes. Defective switches degrade the NoC performability in two ways: firstly, since the direct connection to a core gets lost, the number of available cores in the system decreases. Furthermore, as the network loses a switch, the available network resources and consequently the overall performance decreases. Both effects are measured here independently: (1) the connectivity measures the number of available cores in the degraded system, and (2) retransmission rate, latency, and throughput reflect available network resources as measures of communication performance.

7.1 Simulation Setup

Two cycle accurate SystemC simulation models of the degraded NoCs are created. In the first model, every failing switch is completely isolated by deactivating the ports of all adjacent switches. This represents the switch-accurate diagnosis. In the second model, only the ports necessary to tolerate the faults are deactivated. Here, we incorporate the results of the proposed port-accurate switch diagnosis approach. A fault

tolerant routing algorithm similar to [Zhang08b] bypasses the defective components. The routing algorithm uses the standard XY routing in the fault-free cases and routes the packets over a ring around the faulty switches. By avoiding the north-west and east-south turns, this fault tolerant routing scheme become deadlock-free. The algorithm in [Zhang08b] was originally developed for switch deactivation, but it can be applied for fine-grained port deactivation as well.

The experiments are performed with 100 defect conditions and the results are averaged. For each defect condition, a certain number of stuck-at faults are randomly injected into the random logic of the switches in each NoC. In the experiments, we assume that the NoC is constructed by 12-bit switches in which according to the switch diagnosis results in Chapter 4, about 56% of the faults are dedicated to a single port of the switch. The experiments have been repeated with 32-bit switches, in which 72% of the faults dedicated to a single port of the switch and the results are covered in Appendix B.

7.2 Connectivity

Let a and b denote two arbitrary cores of the system. The user expects that all available cores in the system can communicate with each other. Therefore, the NoC must provide a communication path both from a to b and in reverse. If we consider the cores as the nodes in a graph and add edges between all node pairs which are able to communicate bidirectionally, then the available cores are just the ones contained in the largest clique [Golumbic04] of this graph. The connectivity of a network equals the number of nodes in the largest clique.

We have conducted the experiments for three different network sizes, an 8×8 (commonly used in the NoC experiments), a 12×12 and a 20×20, a relatively large one. The results for the 8×8 and 12×12 networks are covered in Appendix B.1.

Figure 7.1 depicts the average number of linked cores (the average size of the largest cliques) for various fault counts in a 20×20 NoC. The experiments are reported for up to 20 faults, because the environmental condition causing such defect density in the NoC, will affect the cores as well. As the cores usually occupy a larger area on

the chip as compared to the area of the switches, such defect density leads to many defective cores.

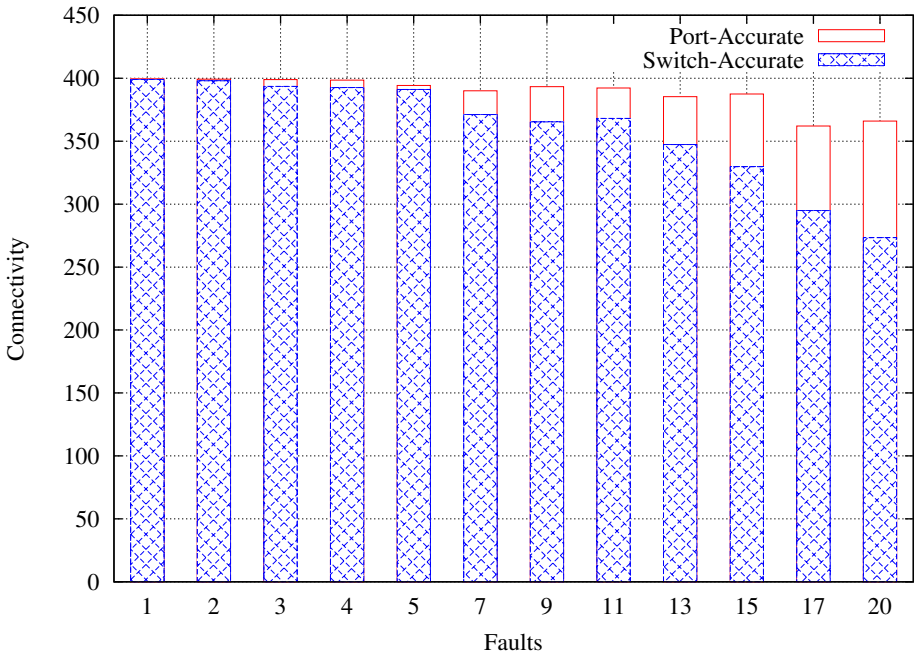


Figure 7.1: Connectivity of the network with switch-accurate and port-accurate diagnosis

The first observation is that despite several faults in the network, the NoC can provide bidirectional communication paths among a large portion of nodes. This is an attractive feature of inherently redundant NoC architectures which can be exploited for effective binning to increase the yield during manufacturing. The degraded NoCs can be used in smaller systems with less resources.

The number of usable nodes increases when using degraded switches in the network. As the fault count increases, the number of connected nodes with port-accurate diagnosis becomes significantly higher. This is due to two factors. Firstly, cores adjacent to defective switches are often still reachable, since the fault may disrupt only a single port towards a neighbouring switch. The affected port is discarded, nevertheless the switch and the core stay connected to the network. Secondly, a completely disabled switch may lead to a partitioning of the network.

For instance, when 20 faults are injected, in average 127 cores get separated from

the network, if every defective switch is completely turned off. Only a small subset of these inaccessible cores are directly connected to a defective switch (maximum 20 cores), thus the majority of the cores become inaccessible due to the lack of bidirectional communication paths within the degraded NoC. With fine-grained port deactivation, the number of inaccessible cores reduces drastically to only 35 cores in average. Here, the number of inaccessible cores is still larger than the actual number of faults in the network due to the lack of bidirectional connections.

In the lower fault rates, the number of usable nodes is also noticeable. Figure 7.2 gives a detailed view of the connectivity result, reported earlier in Fig. 7.1, for one to five faults. Upon injecting one or two faults, the network with switch-accurate diagnosis has only lost one or two nodes respectively. However, when the fault count increases to three, the number of usable nodes drops drastically. With port-accurate diagnosis, up to four faults, the network has lost less than two nodes in average, which is a significant achievement as compared with the switch-accurate diagnosis.

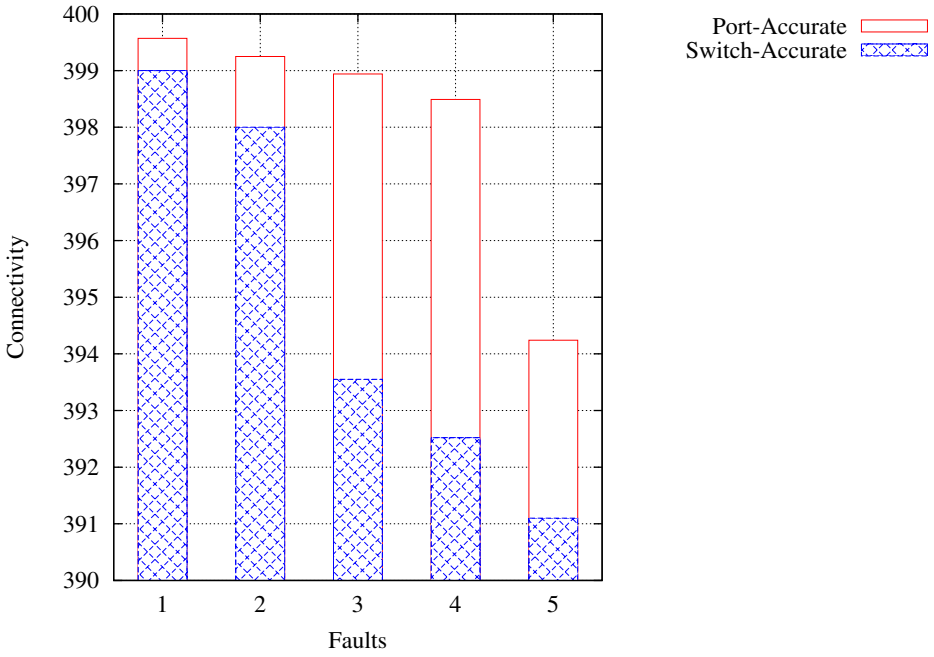


Figure 7.2: Connectivity of the network with switch-accurate and port-accurate diagnosis - 1 to 5 faults (detailed view of Fig. 7.1)

7.2.1 Network Reliability

In the experiments, we consider two networks of the same size: the first with switch-accurate diagnosis and the second with port-accurate diagnosis. As given in Eq. (2.7) of section 2.4.1, the network reliability in the simplified model is defined as: $R_{\mathrm{NoC}}(t) = e^{-tNh(t)}$. Here, $h(t)$ refers to the failure rate of the switch, which means the switch breaks down due to a fault. We assume that the faults in the switch appear with a constant failure rate λ.

Occurrence of any fault in the switches of the first network leads to switch deactivation, i.e. $h(t) = \lambda$. However, in the second network, the switch is deactivated only when the diagnosis fails to find the fault candidate, or when the fault occurs in the router of the switch. Thus, we have $h(t) = \gamma \times \lambda$, where γ is the probability that the switch is disabled entirely. According to the diagnosis results of section 4.5.2, all faults in the modelled switch are diagnosable, therefore:

$$\gamma = \frac{\text{number of faults in the router}}{\text{total number of faults}} \,. \tag{7.1}$$

Figure 7.3 depicts the reliability of both networks with 12-bit switches at $t = 10^4$ hours considering a network with $N = 100$ switches. The parameter λ is taken from the failure rate of some Intersil process technologies in 60% and 95% confidence levels as reported in [Vigrass10]. The failure rate varies between 3.20 FITs[1] up to 31.48 FITs. The reliability comparison indicates that when using port-accurate diagnosis (PA), the probability of which the network can continue its intended operation with all of its resources is higher than the case with switch-accurate diagnosis (SA). In the lower failure rates (up to 10 FITs) the reliability difference in both networks is very small (less than 0.005). In addition, the reliability of both networks is more than 0.99, which is sufficient for many commercial and industrial applications. Though for the safety-critical applications in which even marginal reliability loss is not acceptable, PA diagnosis would be preferred. It must be noted that fault tolerant enhancements can be used in both networks to improve the reliability. However, as the network with

[1]Failure In Time: measure of failure rate in 10^9 device hours.

port-accurate diagnosis has a higher reliability inherently, it can fulfil the reliability requirements at lower costs.

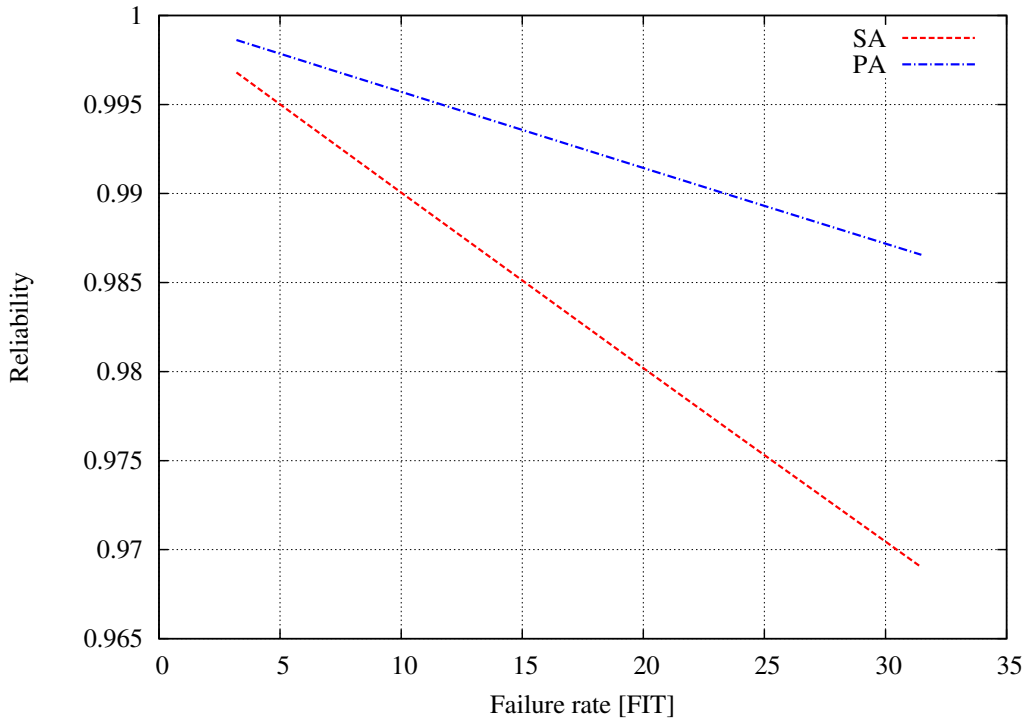


Figure 7.3: Reliability of the network with switch-accurate (SA) and port-accurate (PA) diagnosis at $t = 10^4$ hours: improved reliability by using internal redundancy of the switches

As it is observed in Fig. 7.3, for the higher failure rates, the difference between the reliability of two networks becomes noticeable. When the network with port-accurate diagnosis delivers a reliability of more than 0.98, the reliability of the network with switch-accurate diagnosis drops down to less than 0.97, for $\lambda = 31.48$ FITs. This indicates that in a process technology with a high failure rate, the reliability can be increased significantly by employing port-accurate diagnosis. To achieve a certain reliability level, one may choose the port-accurate diagnosis and a process technology with higher failure rate, or the switch-accurate diagnosis in a process technology with lower failure rates.

7.3 Communication Performance

The second set of experiments is conducted to measure various network parameters which reflect the communication performance. In the simulation models, each switch is connected to a traffic generator core, which generates uniform random traffic with various loads. Each core is sending messages to any other core with equal probability. Let $\Delta t_{\min}$ be the minimum time interval between two consecutive packet injections of a core. In a fault-free network, $\Delta t_{\min}$ is the minimum time that the network interface requires to serve an injected packet. If a core sends a packet at time t, it may send the next packet at time $t' \geq t + \Delta t_{\min}$. With $\Delta t_{\text{avg}} \geq \Delta t_{\min}$ being the average time between two packet injections, the load per core is defined as:

$$\text{load}_{\text{core}} = \frac{\Delta t_{\min}}{\Delta t_{\text{avg}}} \times 100\% \,. \tag{7.2}$$

All active cores in the network produce traffic with the same load, $\text{load}_{\text{core}}$. In a degraded network, some cores may be deactivated as described earlier and the overall load of the network decreases by the ratio of the number of linked cores over the network size:

$$\text{load}_{\text{net}} = \text{load}_{\text{core}} \times \frac{\text{linked cores}}{\text{network size}} \,. \tag{7.3}$$

If the network is fault-free and all cores are active, then the network load is the same as $\text{load}_{\text{core}}$.

7.3.1 Retransmission Rate

In the fault-free network under a balanced load, as long as the time interval between two consecutive packet injections is greater than $\Delta t_{\min}$, there are sufficient network resources to guarantee packet delivery and thus no packets are dropped. However, in the degraded network, packets will be dropped due to the lack of resources, e.g. overflowing buffers, or time to live expiry. In other words, the network resources are insufficient for the injected packets. Dropped packets need to be retransmitted.

Therefore, the retransmission rate is defined as follows:

$$\text{retransmission rate} = \frac{\text{number of dropped packets}}{\text{number of injected packets}} \times 100\% \,. \tag{7.4}$$

It reflects the lack of resources in the degraded network. Higher retransmission rate means that less network resources are available for the injected packets.

The retransmission rate is now measured under different loads and different number of faults both in the network with completely disabled switches (i.e. switch-accurate diagnosis) and the network with degraded switches (i.e. port-accurate diagnosis is at hand). Figure 7.4 compares the average retransmission rate in the presence of various number of faults in 20×20 NoCs with a fixed $\text{load}_{\text{core}}$=14.5% for 12-bit switches. The result for 32-bit switches is covered in Appendix B.2. Compared to the NoC in which defective switches are completely disabled, the retransmission rate is almost cut in half by using the proposed port-accurate diagnosis, particularly in the range of 4 to 13 faults in the network. This clearly reveals the advantage of the additional communication paths provided by the degraded switches especially with high fault densities. However, it is observed that in the presence of 15, 17, and 20 faults, the difference between the dropped packets of the two networks and consequently the retransmission rate is reduced. This is justified with the big difference between the number of linked cores in two networks: As it can be observed in Fig. 7.1, the number of linked cores in the network with switch-accurate diagnosis (where defective switches must be removed) is remarkably less than the network with port-accurate diagnosis (where degraded switches can be used). As a consequence and inspired by Eq. (7.3), the network with removed defective switches is loaded with less traffic (i.e. load_{net} decreases), meaning that the number of injected packets decreases. Therefore, the network resources are shared among less packets and consequently less packets might be dropped due to the lack of resources.

Similar reduction in retransmission rate can be observed under all network loads. Figure 7.5 shows a plot of average retransmission rate over all network loads in the network with 9 injected faults. The network with degraded switches is able to deliver more packets under any traffic condition. Moreover, Fig. 7.5 shows that in general the maximum network load handled by the NoC with degraded switches is higher than that

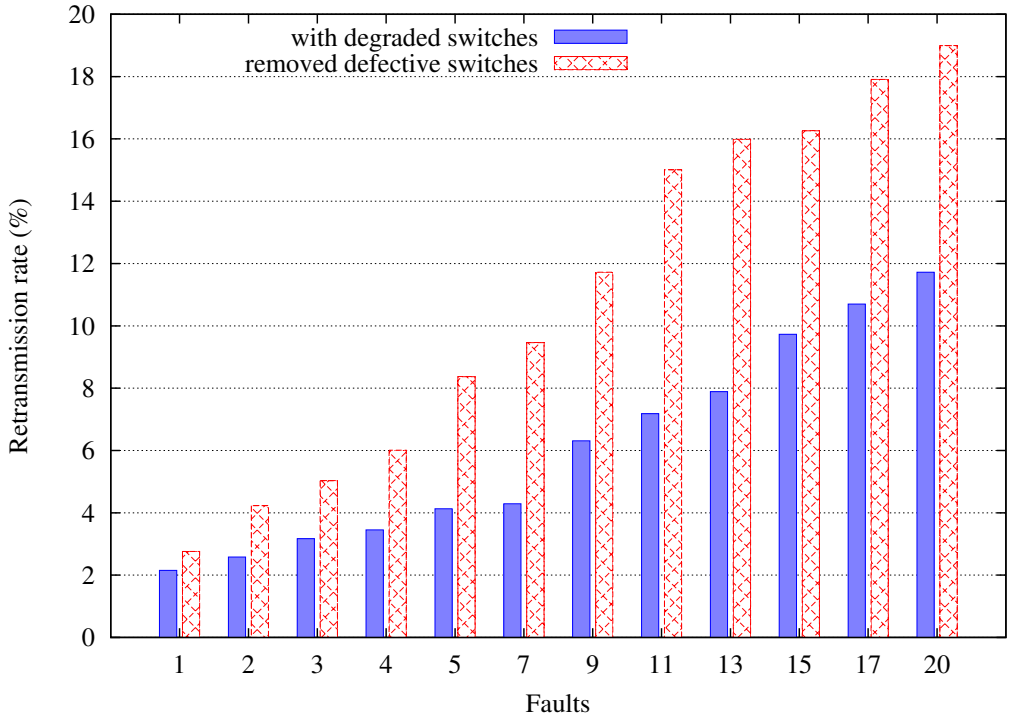


Figure 7.4: Average retransmission rate to fault count under $load_{core}$=14.5%

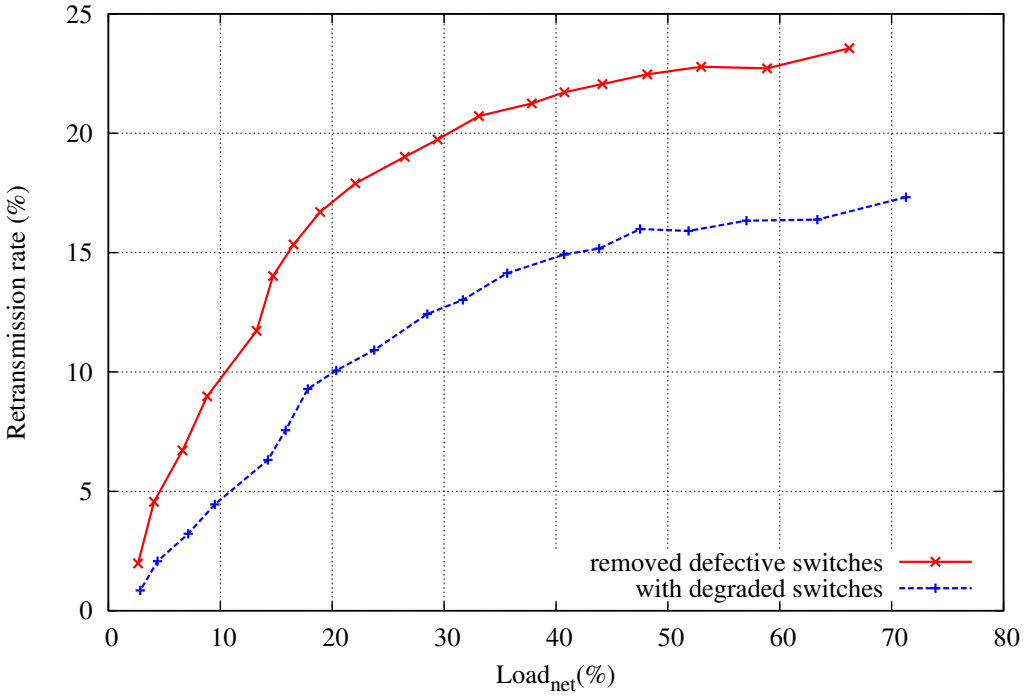


Figure 7.5: Average retransmission rate under load with 9 faults

handled by the NoC with disabled switches. This is due to the fact that the number of linked cores is higher in this case and each additional core adds to $\mathrm{load}_{\mathrm{net}}$.

7.3.2 Latency and Throughput

Figure 7.6 illustrates the latency-load curves of the two networks (SA: the network with switch-accurate diagnosis, and PA: the network with port-accurate diagnosis) for different fault counts. The result indicates that in general the average latency in the network with switch-accurate diagnosis is higher than that in the network with port-accurate diagnosis. For a single fault in the network, the latency in PA network is almost one cycle less than the SA network and the difference remains almost unchanged as the network load increases. With a higher number of faults, the latency difference becomes noticeable, however. In particular, the latency of the SA network becomes more sensitive to the network load and grows up more rapidly when the network load increases.

A similar effect is observed when comparing the throughput-load curves of the two networks for different fault counts as depicted in Fig. 7.7. For the lower network loads, we see that the throughputs of the two networks rise steadily as the network load increases. With a higher number of faults, both networks are saturated faster. However, the maximum throughput of the SA network is lower than that of the PA network in all cases. In other words, since in the SA network the available resources for traffic distribution are less than the PA network, the network is saturated faster.

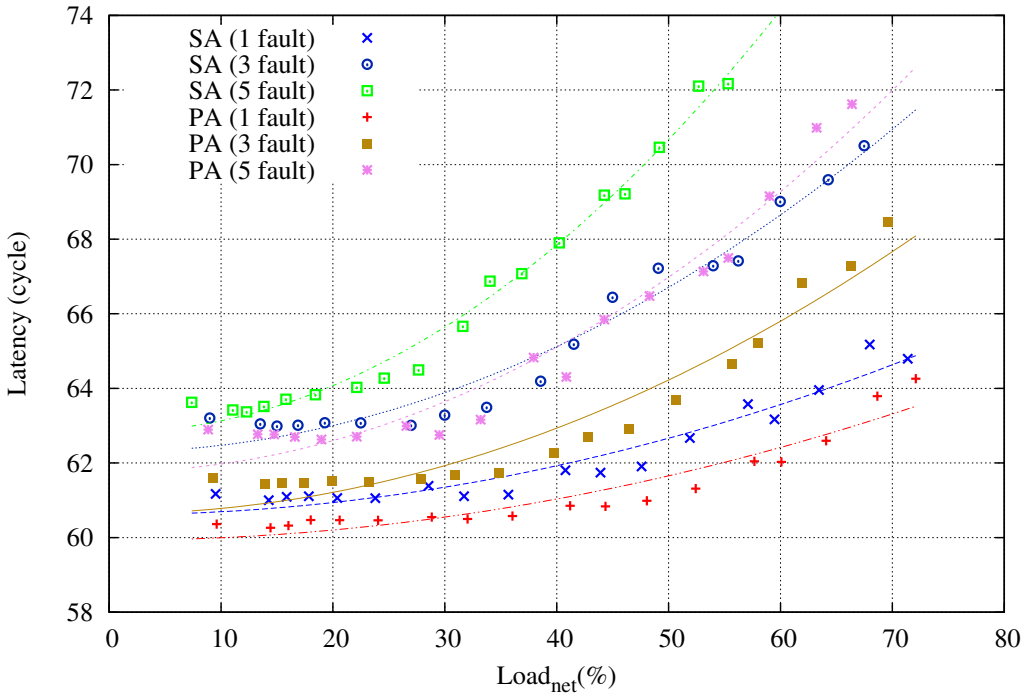


Figure 7.6: Latency-load curves for different fault counts

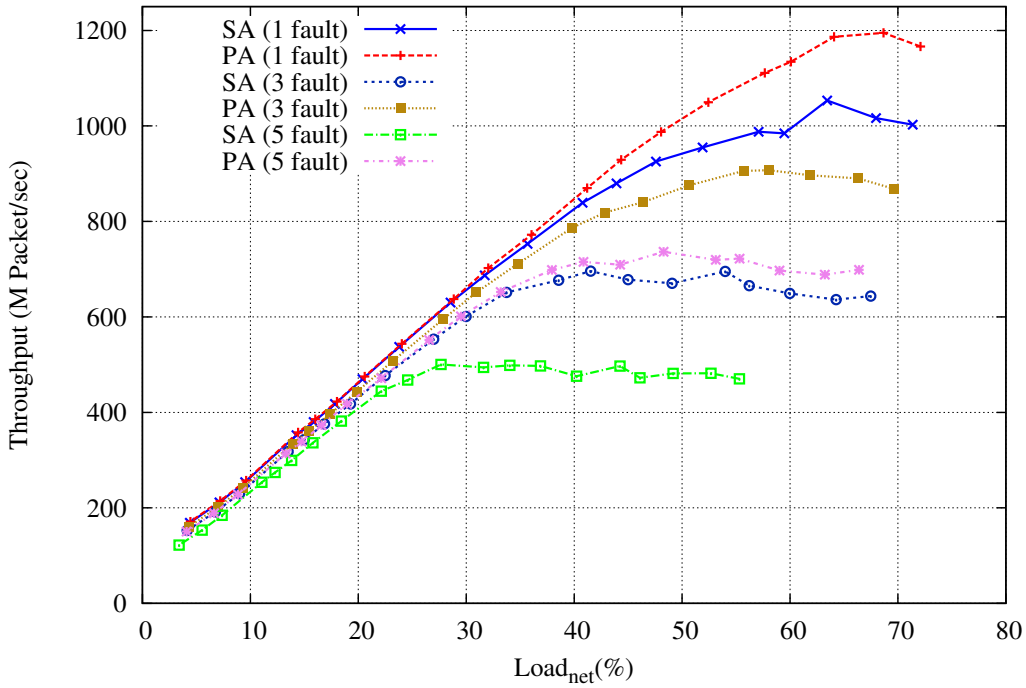


Figure 7.7: Throughput-load curves for different fault counts

7.4 Summary

In this chapter, we compared the performability of defective NoCs in the presence of the switch-accurate and the port-accurate diagnosis schemes and evaluated the impact of diagnosis granularity on the graceful degradation. Network parameters such as connectivity (number of available cores in the degraded network), reliability, retransmission rate, throughput and latency of the degraded networks were measured. Incorporating port-accurate diagnosis enables us to use degraded switches in the network with the advantage of the performance and reliability improvement in degraded NoCs.

8

Summary and Conclusions

The Network-on-Chip (NoC) is the communication alternative for today's many core architectures, with the advantages of latency, power, and dependability. Since cost is the key concern of the current and future technologies as acknowledged by the International Technology Roadmap 2013, inherent redundancy of the NoC makes it more attractive for current design processes which are more vulnerable to various defect mechanisms due to the reduced feature sizes. Reconfigurable NoC architectures allow defective switches and links to be discarded while the network can continue its operation at the expense of degraded performance. This, known as graceful degradation, increases the yield and reduces the overall production costs.

This study has been carried out to conduct test and diagnosis schemes for self-diagnosis of the NoCs aiming at graceful degradation. We define network diagnosis at two accuracy levels, namely, switch-accurate diagnosis and port-accurate diagnosis. Switch-accurate diagnosis considers the switch as the abstracted fault model. In the port-accurate diagnosis, however, switch ports are considered as the abstracted fault model. Port-accurate diagnosis enables graceful degradation in a fine-grained way, since instead of disabling the defective switches completely, only the defective parts of the switches are discarded. This readily improves the performability of defective NoCs. Incorporating fault-free parts of defective switches instead of eliminating the faulty switch entirely, improves the performance of degraded NoCs, as the average latency decreases. Moreover, the reliability of the switch and consequently the reliability

of the NoC-based system will increase.

This dissertation proposes switch-accurate and port-accurate diagnosis schemes to make the NoCs more robust. Having addressed the shortcomings of previous techniques in the NoC area, the main original contributions of this dissertation are:

1. Structural diagnosis of NoC switches (Chapter 4),
2. Structural Software-Based Self-Test of NoCs (Chapter 5).
3. Fault-secure switch architecture (Chapter 6),

To conduct the port-accurate diagnosis, existing techniques to date either divide the switch into several sub-blocks and conduct testing of each block individually, or they perform functional tests. The former increases the design and test costs. The latter, i.e. functional testing, may deliver some confidence about the correct functionality of the switch. However, since it does not search for the exact cause of the malfunction, it cannot be proved which functions are intact and which ones are affected by the fault. Exploring the literature in the past few years clearly indicates that finding the exact cause of a faulty behaviour in the switch and respectively reasoning about the faulty/non-faulty parts of the switch has been neglected in the previous studies. This reveals the originality of conducting **structural diagnosis of NoC switches** which has been presented as the first major contribution of this work.

The novel diagnosis approach integrates the precision of structural testing with information on the functional behaviour in the presence of defects. Through analysing the structural test fail data, the defect location inside the NoC switches is determined. A functional mapping approach identifies then the fault-free switch functions that can be retained for the NoC operation. This approach is based on the standard flow and architecture of industrial volume test and thus does not impose any additional hardware overhead and can deal with an arbitrary class of switches. Moreover, the same capabilities which are needed to isolate a defective switch or link from the network, suffice to support the individual port deactivation.

Over the last years, several proposals have been developed for testing the NoC, which identify defective switches of the network. Structural tests achieve a high struc-

tural fault coverage by implementing the standard scan-based testing. Functional test schemes reduce the test application time but sacrifice the structural fault coverage.

This gap is bridged by the proposed **structural Software-Based Self-Test (SBST)** method for NoCs as the second contribution of this dissertation. The SBST approach utilizes the NoC infrastructure to apply test patterns targeting structural faults in the switches and interconnect links.

As the SBST patterns are applied to the switch under test via the ambient processing elements, only the functional inputs and outputs of the switch are available. Here, in contrast to scan-based test approaches, direct controllability and observability of the internal states is not possible. Therefore, achieving a high structural fault coverage becomes a challenging task. Tackling this problem, the SBST pattern generation relies on the sequential switch, which is modelled by replicating the combinational switch logic for several time intervals. Test pattern generation is modelled as a Boolean satisfiability problem. The SAT instance comprises the switch and the faulty copy, including the structural fault. As a single fault appears in multiple places in the unrolled circuit, the model deals with multiple faults, and the fault propagation through several copies of the circuit is taken into account. Additional literals are added to the instance to specify the structure of the NoC-packets so that the input assignments are in the packet format.

The experiments reveal that the SBST method outperforms the state-of-the-art structural and functional testing approaches for NoCs by reducing the test time and test hardware overhead without the loss of structural fault coverage. Moreover, the method is quite effective for various types of structural faults. Therefore, it becomes attractive for both manufacturing and in-field testing.

By start-up or on-demand testing of the NoC in the non-functional periods, defective switches can be identified. However, a fault may appear in the operational mode, leading to erroneous message transfer over the NoC. Several many-core systems with different applications and requirements rely on the NoC as the communication platform and expect the NoC to fulfil their requirements. Therefore, concurrent error detection of NoCs is essential to provide a promising robust communication infrastructure.

To implement Concurrent Error Detection (CED), duplication with comparison and

Triple Modular Redundancy (TMR) have been examined for the NoCs so far. Besides, data encoding techniques have been typically used to conduct concurrent error detection. On the one side, duplication and TMR do not consider special characteristics of the switches and therefore impose a huge area overhead, which drastically grows up as the number of switches in the network increases. On the other side, data encoding techniques are capable of detecting only the faults in the inter-switch links and intra-switch datapath elements such as multiplexers. However, faults in the controlling logic, which manages the flow of data among the switch ports, remain undetected.

The third contribution of this dissertation is the **area efficient fault-secure synthesis technique** that overcomes the shortcomings of the state-of-the-art techniques for concurrent error detection of NoCs. The fault-secure structure ensures that any single fault in the datapath and controlling part with erroneous effect on the switch operation is announced by an error signal dedicated to each switch. The key idea is to establish a synthesis technique that takes advantage of data encoding, in order to detect faults in the datapath, and multiple parity bits, to cover the rest of the faults. Finding the area efficient fault-secure structure is defined as a Boolean satisfiability (SAT) problem, where initial structure includes data encoding at every flit of the NoC packets. The non-data switch outputs are then distributed among multiple parity groups. The synthesis technique significantly reduces the area overhead of the CED as compared to the available techniques, while it detects all errors resulting from single point combinational and transition faults in the switches and interconnect links.

8.1 A Glance at the Future

The NoC has been an attractive communication alternative because of its inherent redundancies and fault tolerant features. Introducing the port-accurate diagnosis makes the NoCs more robust and therefore much more attractive in today's many-core era. The granular diagnosis achieves the highest gain of the non-faulty structures of the chip and avoids any pessimistic elimination of the components that are partially usable. This improves the performability of defective NoCs and hence it will fulfil the requirements of a broader range of applications, making the NoCs much more popular.

As acknowledged by the international technology roadmap 2013, the industry is still seeking more efficient test methodologies to reduce the production and after production costs. Thus, the efficient structural SBST technique for NoCs will find its place next to the SBST methods for microprocessors. Furthermore, the unique modelling technique, which was introduced in this dissertation for SBST pattern generation, opens a new window for the future research to build up even more efficient test scenarios to further shorten the design and test phases. The model can relate the structural circuit information to the sequential switch functions. In fact, the accuracy of the structural circuit has been transferred to the switch functional level. This is what was missing in the research area, and therefore each of the structural and functional test approaches were suffering from their own restrictions. Using this model, structural faults can be directly targeted by the approaches created at the functional level. For example, the model can be used to develop dedicated structural patterns for a specific test access mechanism (TAM) and thus mitigating the overhead of adapting the structural patterns to the TAM requirements. Indeed, the method can be used to combine the test pattern generation and TAM adaptation phases to shorten the design and test time.

Another usage of this modelling is to define functional faults with a correspondence to the structural faults in the circuit. The so far introduced functional fault models for NoCs do not offer such direct correspondence. Therefore, the fault coverage of the respective functional test methods is rather limited so that it becomes unattractive for industrial purposes. This drawback can be overcome by the unique advantage of the proposed modelling technique to relate the structural and functional faults and consequently measuring the quality of functional tests in terms of structural fault coverage.

The fault-secure structure can be used for test response compaction as shown in the experiments. The SBST pattern generation approach can be further optimized by considering the fault-secure switch architecture in order to reduce the data volume and test application time. In this case, the components of the fault-secure switch must be introduced to the SAT instance during test pattern generation. The encoded outputs can be used as compacted test response to further reduce the response volume without loss of fault coverage.

The synthesis technique for fault-secure switch can be extended to model other popular fault detection mechanisms for NoCs, for example the packet encoding instead

of flit encoding. The packet encoding technique requires outputs from several cycles of the switch operation. Therefore, it requires modelling of the sequential switch. This is where the model for SBST pattern generation can be employed to extend the fault-secure synthesis technique.

The concluding remark is that special characteristics of the NoC must be considered during development of the test and diagnosis schemes in order to achieve efficient solutions. Such schemes should not be design-specific, while they should be easily integrated into the standard production flow, as it is the case for the proposed approaches in this dissertation. Automated NoC-specific approaches will be of special interest for the industry which is seeking faster solutions in order to reduce the costs.

Bibliography

[Abdel-Khalek12] R. Abdel-Khalek and V. Bertacco. Functional post-silicon diagnosis and debug for networks-on-chip. In *Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 557–563. San Jose, CA, 2012. (cited on page 43.)

[Abraham86] J. A. Abraham and W. K. Fuchs. Fault and Error Models for VLSI. *Proceedings of the IEEE*, 74(5):639–654, 1986. (cited on page 4.)

[Acken91] J. M. Acken and S. D. Millman. Accurate modeling and simulation of bridging faults. In *Proc. IEEE Custom Integrated Circuits Conference*, pages 17–4. San Diego, CA, 1991. (cited on page 24.)

[Agarwal04] A. Agarwal, B. Paul, and K. Roy. A novel fault tolerant cache to improve yield in nanometer technologies. In *Proc. 10th International On-Line Testing Symposium (IOLTS)*, pages 149–154. Funchal, Portugal, 2004. (cited on page 2.)

[Agarwal09] A. Agarwal, C. Iskander, and R. Shankar. Survey of Network on Chip (NoC) Architectures & Contributions. *Journal of engineering, Computing and Architecture*, 3(1):21–27, 2009. (cited on pages 1, 22, and 37.)

[Aisopos11] K. Aisopos, C.-H. Chen, and L.-S. Peh. Enabling system-level modeling of variation-induced faults in networks-on-chips. In *Proc. 48th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 930–935. San Diego, CA, 2011. (cited on page 43.)

[Ali05] M. Ali, M. Welzl, M. Zwicknagl, and S. Hellebrand. Considerations for fault-tolerant network on chips. In *Proc. 17th International Conference on Microelectronics (ICM)*, pages 178–182. Islamabad, Pakistan, 2005. (cited on page 57.)

[Almukhaizim03] S. Almukhaizim and Y. Makris. Fault tolerant design of combinational and sequential logic based on a parity check code. In *Proc. IEEE International Symposium on Defect and Fault Tolerance (DFT)*, pages 563–570. Cambridge, MA, 2003. (cited on page 44.)

[Almukhaizim06] S. Almukhaizim, P. Drineas, and Y. Makris. Entropy-Driven Parity-Tree Selection for Low-Overhead Concurrent Error Detection in Finite State Machines. *IEEE Transaction on Computer Aided Design (CAD)*, 25(8):1547–1554, 2006. (cited on pages 45, 104, 180, and 181.)

[Amory05a] A. Amory, E. Briao, E. Cota, M. Lubaszewski, and F. Moraes. A scalable test strategy for network-on-chip routers. In *Proc. IEEE International Test Conference (ITC)*, pages 551–559. Austin, TX, 2005. (cited on page 39.)

[Amory05b] A. M. Amory, E. W. Brião, É. F. Cota, M. S. Lubaszewski, and F. G. Moraes. A cost-effective test flow for homogeneous network-on-chip. In *Proc. European Test Symposium (ETS)*. Tallinn, Estonia, 2005. (cited on pages 8, 57, and 93.)

[Andreev06] K. Andreev and H. Racke. Balanced Graph Partitioning. *Journal of Theory of Computing Systems*, 39(6):929–939, 2006. (cited on page 114.)

[Augustin10] M. Augustin, M. Gossel, and R. Kraemer. Reducing the area overhead of TMR-systems by protecting specific signals. In *Proc. 16th IEEE International On-Line Testing Symposium (IOLTS)*, pages 268–273. Corfu Island, Greece, 2010. (cited on page 102.)

[Avizienis04] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, 2004. (cited on page 28.)

[Barrette96] T. Barrette, V. Bhide, K. De, M. Stover, and E. Sugasawara. Evaluation of early failure screening methods. In *Proc. IEEE International Workshop on IDDQ Testing*, pages 14–17. Washington, DC, 1996. (cited on page 11.)

[Bartenstein01] T. Bartenstein, D. Heaberlin, L. M. Huisman, and D. Sliwinski. Diagnosing combinational logic designs using the single location at-a-time (SLAT) paradigm. In *Proc. IEEE International Test Conference (ITC)*, pages 287–296. Baltimore, MD, 2001. (cited on pages 27 and 58.)

[Baumann05] R. Baumann. Soft Errors in Advanced Computer Systems. *IEEE Design & Test of Computers*, 22(3):258–266, 2005. (cited on page 43.)

[Benini02] L. Benini and G. De Micheli. Networks on chip: a new paradigm for systems on chip design. In *Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pages 418–419. Paris, France, 2002. (cited on page 1.)

[Bertozzi05] D. Bertozzi, L. Benini, and G. De Micheli. Error Control Schemes for On-Chip Communication Links: the Energy-Reliability Tradeoff. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(6):818–831, 2005. (cited on pages 10, 38, 46, and 47.)

[Bhardwaj12] K. Bhardwaj, K. Chakraborty, and S. Roy. Towards graceful aging degradation in NoCs through an adaptive routing algorithm. In *49th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 382–391. San Francisco, CA, 2012. (cited on pages 30 and 38.)

[Biere09] A. Biere, M. Heule, and H. van Maaren. *Handbook of Satisfiability*, volume 185. IOS Press, 2009. (cited on page 32.)

[Bolchini97] C. Bolchini, F. Salice, and D. Sciuto. Parity bit code: achieving a complete fault coverage in the design of TSC combinational networks. In *Proc. Seventh Great Lakes Symposium on VLSI*, pages 32–37. Urbana-Champaign, IL, 1997. (cited on pages 44 and 45.)

[Borkar05] S. Borkar. Designing Reliable Systems from Unreliable Components: the Challenges of Transistor Variability and Degradation. *IEEE Micro*, 25(6):10–16, 2005. (cited on pages 9 and 43.)

[Borkar07] S. Borkar. Thousand core chips: a technology perspective. In *Proc. 44th annual Design Automation Conference (DAC)*, pages 746–749. San Diego, CA, 2007. (cited on page 1.)

[Brahme84] D. Brahme and J. A. Abraham. Functional Testing of Microprocessors. *IEEE Transactions on Computers*, 100(6):475–485, 1984. (cited on page 41.)

[Bushnell00] M. Bushnell and V. D. Agrawal. *Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits*, volume 17, chapter 8. Springer, 2000. (cited on pages 77 and 90.)

[Chaix10] F. Chaix, D. Avresky, N.-E. Zergainoh, and M. Nicolaidis. Fault-tolerant deadlock-free adaptive routing for any set of link and node failures in multi-cores systems. In *Proc. 9th IEEE International Symposium on Network Computing and Applications (NCA)*, pages 52–59. Cambridge, MA, 2010. (cited on page 37.)

[Chakradhar95] S. T. Chakradhar, A. Balakrishnan, and V. D. Agrawal. An Exact Algorithm for Selecting Partial Scan Flip-flops. *Journal of Electronic Testing*, 7(1-2):83–93, 1995. (cited on page 77.)

[Chang11] Y.-C. Chang, C.-T. Chiu, S.-Y. Lin, and C.-K. Liu. On the design and analysis of fault tolerant NoC architecture using spare routers.

In *Proc. 16th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 431–436. Yokohama, Japan, 2011. (cited on page 35.)

[Chen96] C.-A. Chen and S. K. Gupta. A satisfiability-based test generator for path delay faults in combinational circuits. In *Proc. 33rd Design Automation Conference (DAC)*, pages 209–214. Las Vegas, NV, 1996. (cited on page 32.)

[Chen97] W. Chen, S. K. Gupta, and M. A. Breuer. Analytic models for crosstalk delay and pulse analysis under non-ideal inputs. In *Proc. International Test Conference (ITC)*, pages 809–818. Washington, DC, 1997. (cited on page 24.)

[Chen01] L. Chen and S. Dey. Software-based Self-testing Methodology for Processor Cores. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(3):369–380, 2001. (cited on page 74.)

[Chen12] C. Chen, Y. Lu, and S. Cotofana. A novel flit serialization strategy to utilize partially faulty links in networks-on-chip. In *Proc. Sixth IEEE/ACM International Symposium on Networks on Chip (NoCS)*, pages 124–131. Copenhagen, Denmark, 2012. (cited on page 48.)

[Chen13] H. Chen and J. Marques-Silva. A Two-Variable Model for SAT-Based ATPG. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(12):1943–1956, 2013. (cited on page 32.)

[Chiu00] G.-M. Chiu. The Odd-Even Turn Model for Adaptive Routing. *IEEE Transactions on Parallel and Distributed Systems*, 11(7):729–738, 2000. (cited on pages 37 and 57.)

[Clarke01] E. Clarke, A. Biere, R. Raimi, and Y. Zhu. Bounded Model Checking Using Satisfiability Solving. *Formal Methods in System Design*, 19(1):7–34, 2001. (cited on page 33.)

[Concatto09] C. Concatto, P. Almeida, F. Kastensmidt, E. Cota, M. Lubaszewski, and M. Herve. Improving yield of torus NoCs through fault-diagnosis-and-repair of interconnect faults. In *Proc. 15th IEEE International On-Line Testing Symposium (IOLTS)*, pages 61–66. Sesimbra-Lisbon, Portugal, 2009. (cited on page 41.)

[Constantinides06] K. Constantinides, S. Plaza, J. Blome, B. Zhang, V. Bertacco, S. Mahlke, T. Austin, and M. Orshansky. Bulletproof: A defect-tolerant CMP switch architecture. In *Proc. 12th International Symposium on High-Performance Computer Architecture*, pages 5–16. Austin, TX, 2006. (cited on pages 30, 35, and 38.)

[Corno03] F. Corno, G. Cumani, M. Sonza Reorda, and G. Squillero. Fully automatic test program generation for microprocessor cores. In *Proc. Design, Automation and Test in Europe (DATE)*, pages 1006–1011. Munich, Germany, 2003. (cited on page 74.)

[Cota08] É. Cota, F. L. Kastensmidt, M. Cassel, M. Herve, P. Almeida, P. Meirelles, A. Amory, and M. Lubaszewski. 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, 2008. (cited on pages 39 and 41.)

[Cota11] E. Cota, A. de Morais Amory, and M. S. Lubaszewski. *Reliability, Availability and Serviceability of Networks-on-chip*. Springer, 2011. (cited on page 35.)

[Cuviello99] M. Cuviello, S. Dey, X. Bai, and Y. Zhao. Fault modeling and simulation for crosstalk in system-on-chip interconnects. In *Proc. IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 297–303. San Jose, CA, 1999. (cited on page 48.)

[Dalirsani07] A. Dalirsani, M. Hosseinabady, and Z. Navabi. An analytical model for reliability evaluation of NoC architectures. In *Proc. 13th*

IEEE International On-Line Testing Symposium (IOLTS), pages 49–56. Crete, Greece, 2007. (cited on pages 30 and 38.)

[Dalirsani11] A. Dalirsani, S. Holst, M. Elm, and H. Wunderlich. Structural Test for Graceful Degradation of NoC Switches. In *Proc. 16th IEEE European Test Symposium (ETS)*, pages 183–188. Trondheim, Norway, 2011. (cited on pages 51 and 120.)

[Dalirsani12] A. Dalirsani, S. Holst, M. Elm, and H.-J. Wunderlich. Structural Test and Diagnosis for Graceful Degradation of NoC Switches. *Journal of Electronic Testing: Theory and Applications (JETTA)*, 28(6):831–841, 2012. (cited on page 51.)

[Dalirsani14a] A. Dalirsani, M. E. Imhof, and H.-J. Wunderlich. Structural Software-Based Self-Test of Network-on-Chip. In *Proc. 32nd IEEE VLSI Test Symposium (VTS)*, pages 1–6. Napa, CA, 2014. (cited on page 43.)

[Dalirsani14b] A. Dalirsani, M. A. Kochte, and H.-J. Wunderlich. Area-Efficient Synthesis of Fault-Secure NoC Switches. In *Proc. 20th IEEE International On-Line Testing Symposium (IOLTS)*, pages 13–18. Catalunya, Spain, 2014. (cited on page 47.)

[Dally87] W. J. Dally and C. L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, 100(5):547–553, 1987. (cited on page 22.)

[Dally03a] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers, 2003. (cited on page 1.)

[Dally03b] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*, chapter Routing Basics, pages 159–171. Morgan Kaufmann Publishers, 2003. (cited on pages 22 and 36.)

[Dally03c] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*, chapter Adaptive Routing, pages 189–201. Morgan Kaufmann Publishers, 2003. (cited on pages 22 and 36.)

[Dally03d] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*, chapter Deadlock and Livelock, pages 257–282. Morgan Kaufmann Publishers, 2003. (cited on pages 22 and 37.)

[Dally03e] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*, chapter Topology Basics, pages 45–74. Morgan Kaufmann Publishers, 2003. (cited on page 31.)

[De94] K. De, C. Natarajan, D. Nair, and P. Banerjee. RSYN: A System for Automated Synthesis of Reliable Multilevel Circuits. *IEEE Transaction on VLSI Systems*, 2(2):186–195, 1994. (cited on page 44.)

[Duato03a] J. Duato, S. Yalamanchili, and L. Ni. *Interconnection Networks*, chapter Message Switching Layer, pages 43–82. Morgan Kaufmann, 2003. (cited on page 21.)

[Duato03b] J. Duato, S. Yalamanchili, and L. Ni. *Interconnection Networks*, chapter Fault-Tolerant Routing, pages 287–356. Morgan Kaufmann, 2003. (cited on pages 22 and 37.)

[Eggersglüß13] S. Eggersglüß, R. Wille, and R. Drechsler. Improved SAT-based ATPG: More constraints, better compaction. In *Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 85–90. San Jose, CA, 2013. (cited on page 32.)

[Ejlali07] A. Ejlali, B. Al-Hashimi, P. Rosinger, and S. Miremadi. Joint consideration of fault-tolerance, energy-efficiency and performance in on-chip networks. In *Proc. Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–6. Nice, France, 2007. (cited on pages 30 and 38.)

[Elm10] M. Elm and H.-J. Wunderlich. BISD: scan-based built-in self-diagnosis. In *Proc. of Design, Automation and Test in Europe (DATE)*, pages 1243–1248. Dresden, Germany, 2010. (cited on page 54.)

[Fey08] G. Fey and R. Drechsler. A basis for formal robustness checking. In *Proc. International Symposium on Quality Electronic Design (ISQED)*, pages 784–789. San Jose, CA, 2008. (cited on page 105.)

[Fick09a] D. Fick, A. DeOrio, G. Chen, V. Bertacco, D. Sylvester, and D. Blaauw. A highly resilient routing algorithm for fault-tolerant NoCs. In *Proc. of the Conference on Design, Automation and Test in Europe (DATE)*, pages 21–26. Nice, France, 2009. (cited on page 37.)

[Fick09b] D. Fick, A. DeOrio, J. Hu, V. Bertacco, D. Blaauw, and D. Sylvester. Vicis: a reliable network for unreliable silicon. In *Proc. 46th Design Automation Conference (DAC)*, pages 812–817. San Francisco, CA, 2009. (cited on pages 8, 11, 49, 50, and 57.)

[Frantz06a] A. P. Frantz, L. Carro, E. Cota, and F. L. Kastensmidt. Evaluating SEU and crosstalk effects in network-on-chip routers. In *Proc. 12th IEEE International On-Line Testing Symposium (IOLTS)*, pages 191–192. Lake of Como, Italy, 2006. (cited on page 43.)

[Frantz06b] A. P. Frantz, F. L. Kastensmidt, L. Carro, and E. Cota. Dependable network-on-chip router able to simultaneously tolerate soft errors and crosstalk. In *Proc. IEEE International Test Conference (ITC)*, pages 1–9. Santa Clara, CA, 2006. (cited on pages 9, 10, 46, and 47.)

[Frantz07] A. P. Frantz, M. Cassel, F. L. Kastensmidt, É. Cota, and L. Carro. Crosstalk-and SEU-Aware Networks on Chips. *IEEE Design & Test of Computers*, 24(4):340–350, 2007. (cited on page 39.)

[Friedman67] A. Friedman. Fault Detection in Redundant Circuits. *IEEE Transactions on Electronic Computers*, (1):99–100, 1967. (cited on page 26.)

[Fukushima09] Y. Fukushima, M. Fukushi, and S. Horiguchi. Fault-tolerant routing algorithm for network on chip without virtual channels. In *Proc. 24th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, pages 313–321. Chicago, IL, 2009. (cited on pages 2 and 37.)

[Garey90] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1990. (cited on page 114.)

[Ghofrani12] A. Ghofrani, R. Parikh, S. Shamshiri, A. DeOrio, K.-T. Cheng, and V. Bertacco. Comprehensive online defect diagnosis in on-chip networks. In *Proc. 30th IEEE VLSI Test Symposium (VTS)*, pages 44–49. Hyatt Maui, HI, 2012. (cited on pages 10, 39, 43, 47, 49, and 50.)

[Ghosh05] S. Ghosh, N. Touba, and S. Basu. Synthesis of low power CED circuits based on parity codes. In *Proc. 23rd IEEE VLSI Test Symposium*, pages 315–320. Palm Springs, CA, 2005. (cited on page 44.)

[Gielen08] G. Gielen, P. De Wit, E. Maricau, J. Loeckx, J. Martin-Martinez, B. Kaczer, G. Groeseneken, R. Rodriguez, and M. Nafria. Emerging yield and reliability challenges in nanometer CMOS technologies. In *Proc. Design, Automation and Test in Europe (DATE)*, pages 1322–1327. Munich, Germany, 2008. (cited on page 9.)

[Glass92] C. J. Glass and L. M. Ni. The Turn Model for Adaptive Routing. In *ACM SIGARCH Computer Architecture News*, volume 20, pages 278–287. 1992. (cited on page 37.)

[Goessel08] M. Goessel, V. Ocheretny, E. Sogomonyan, and D. Marienfeld. *New Methods of Concurrent Checking*. Springer, 2008. (cited on pages 9 and 44.)

[Golumbic04] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*, volume 57. Elsevier, 2004. (cited on pages 108 and 124.)

[Goyal87] A. Goyal and A. N. Tantawi. Evaluation of Performability for Degradable Computer Systems. *IEEE Transactions on Computers*, 100(6):738–744, 1987. (cited on page 28.)

[Grecu05] C. Grecu, P. Pande, B. Wang, A. Ivanov, and R. Saleh. Methodologies and algorithms for testing switch-based NoC interconnects. In *Proc. 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, pages 238–246. Monterey, CA, 2005. (cited on pages 8, 39, and 40.)

[Grecu06a] C. Grecu, A. Ivanov, R. Saleh, E. S. Sogomonyan, and P. P. Pande. On-line fault detection and location for NoC interconnects. In *Proc. 12th IEEE International On-Line Testing Symposium (IOLTS)*, pages 145–150. Lake of Como, Italy, 2006. (cited on page 47.)

[Grecu06b] C. Grecu, P. Pande, A. Ivanov, and R. Saleh. BIST for network-on-chip interconnect infrastructures. In *Proc. 24th IEEE VLSI Test Symposium (VTS)*, pages 30–35. Berkeley, CA, 2006. (cited on pages 39, 40, and 57.)

[Grecu07a] C. Grecu, L. Anghel, P. P. Pande, A. Ivanov, and R. Saleh. Essential fault-tolerance metrics for NoC infrastructures. In *13th IEEE International On-Line Testing Symposium (IOLTS)*, pages 37–42. Crete, Greece, 2007. (cited on pages 10, 46, and 47.)

[Grecu07b] C. Grecu, A. Ivanov, R. Saleh, and P. Pande. Testing Network-on-Chip Communication Fabrics. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(12):2201–2214, 2007. (cited on pages 39 and 48.)

[Gruning91] T. Gruning, U. Mahlstedt, and H. Koopmeiners. DIATEST: A fast diagnostic test pattern generator for combinational circuits.

In *Proc. IEEE International Conference on Computer-Aided Design (ICCAD)*, pages 194–197. Santa Clara, CA, 1991. (cited on page 4.)

[Hamzaoglu99] I. Hamzaoglu and J. H. Patel. New Techniques for Deterministic Test Pattern Generation. *Journal of Electronic Testing*, 15(1-2):63–73, 1999. (cited on page 106.)

[Hennessy12] J. L. Hennessy and D. A. Patterson. *Computer architecture: a quantitative approach*. Elsevier, 2012. (cited on page 31.)

[Hetherington99] G. Hetherington, T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan, and J. Rajski. Logic BIST for large industrial designs: real issues and case studies. In *Proc. International Test Conference (ITC)*, pages 358–367. Atlantic City, NJ, 1999. (cited on page 40.)

[Holst09] S. Holst and H.-J. Wunderlich. Adaptive Debug and Diagnosis Without Fault Dictionaries. *Journal of Electronic Testing*, 25(4-5):259–268, 2009. (cited on pages 4, 27, 58, 59, 69, 119, and 120.)

[Hora02] C. Hora, R. Segers, S. Eichenberger, and M. Lousberg. An effective diagnosis method to support yield improvement. In *Proc. International Test Conference (ITC)*, pages 260–269. Baltimore, MD, 2002. (cited on page 27.)

[Hosseinabady06] M. Hosseinabady, A. Banaiyan, M. N. Bojnordi, and Z. Navabi. A concurrent testing method for NoC switches. In *Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1171–1176. Munich, Germany, 2006. (cited on page 39.)

[Hosseinabady07] M. Hosseinabady, A. Dalirsani, and Z. Navabi. Using the inter-and intra-switch regularity in NoC switch testing. In *Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. Nice, France, 2007. (cited on pages 8, 39, and 57.)

[Hsiao70] M.-Y. Hsiao. A class of optimal minimum odd-weight-column SEC-DED codes. *IBM Journal of Research and Development*, 14(4):395–401, 1970. (cited on page 115.)

[Hu04] J. Hu and R. Marculescu. DyAD: smart routing for networks-on-chip. In *Proc. 41st annual Design Automation Conference (DAC)*, pages 260–263. San Diego, CA, 2004. (cited on page 37.)

[Huisman04] L. M. Huisman. Diagnosing Arbitrary Defects in Logic Designs Using Single Location at a Time (SLAT). *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(1):91–101, 2004. (cited on pages 27, 39, and 58.)

[Huisman05] L. M. Huisman. *Data Mining and Diagnosing IC Fails*. Springer, 2005. (cited on pages 27 and 58.)

[Hunger08] M. Hunger and S. Hellebrand. Verification and Analysis of Self-Checking Properties through ATPG. In *Proc. 14th IEEE International On-Line Testing Symposium (IOLTS)*, pages 25–30. Rhodes, Greece, 2008. (cited on page 105.)

[Jantsch03] A. Jantsch and H. Tenhunen. *Networks on chip*, chapter Will Networks on Chip Close the Productivity Gap?, pages 3–19. Kluwer Academic Publishers, 2003. (cited on page 1.)

[Jo12] S. Jo, T. Matsumoto, and M. Fujita. SAT-based automatic rectification and debugging of combinational circuits with LUT insertions. In *Proc. 21st IEEE Asian Test Symposium (ATS)*, pages 19–24. Niigata, Japan, 2012. (cited on page 32.)

[Kakoee11a] M. Kakoee, V. Bertacco, and L. Benini. A distributed and topology-agnostic approach for on-line NoC testing. In *Proc. Fifth IEEE/ACM International Symposium on Networks on Chip (NoCS)*, pages 113–120. Pittsburg, PA, 2011. (cited on page 43.)

[Kakoee11b] M. R. Kakoee, V. Bertacco, and L. Benini. ReliNoC: A reliable network for priority-based on-chip communication. In *Proc.*

Design, Automation & Test in Europe Conference & Exhibition (DATE), pages 1–6. Grenoble, France, 2011. (cited on pages 11 and 35.)

[Kang10] Y. H. Kang, T.-J. Kwon, and J. Draper. Fault-tolerant flow control in on-chip networks. In *Proc. Fourth ACM/IEEE International Symposium on Networks-on-Chip (NOCS)*, pages 79–86. Grenoble, France, 2010. (cited on pages 35 and 47.)

[Karimi08] N. Karimi, A. Alaghi, M. Sedghi, and Z. Navabi. Online Network-on-Chip Switch Fault Detection and Diagnosis Using Functional Switch Faults. *Journal of Universal Computer Science*, 14(22):3716–3736, 2008. (cited on pages 39 and 43.)

[Keckler11] S. W. Keckler, W. J. Dally, B. Khailany, M. Garland, and D. Glasco. GPUs and the Future of Parallel Computing. *IEEE Micro*, 31(5):7–17, 2011. (cited on page 1.)

[Kim01] Y. C. Kim, V. D. Agrawal, and K. K. Saluja. Combinational test generation for various classes of acyclic sequential circuits. In *Proc. International Test Conference (ITC)*, pages 1078–1087. Baltimore, MD, 2001. (cited on page 76.)

[Kohler10] A. Kohler, G. Schley, and M. Radetzki. Fault Tolerant Network on Chip Switching with Graceful Performance Degradation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(6):883–896, 2010. (cited on pages 7, 11, 47, 56, and 57.)

[Kranitis05] N. Kranitis, A. Paschalis, D. Gizopoulos, and G. Xenoulis. Software-Based Self-Testing of Embedded Processors. *IEEE Transactions on Computers*, 54(4):461–475, 2005. (cited on page 74.)

[Krstic98] A. Krstic and K.-T. Cheng. *Delay Fault Testing for VLSI Circuits*, volume 14. Springer, 1998. (cited on page 24.)

[Kundu05] S. Kundu, S. T. Zachariah, Y.-S. Chang, and C. Tirumurti. On Modeling Crosstalk Faults. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(12):1909–1915, 2005. (cited on page 24.)

[Kunzmann90] A. Kunzmann and H.-J. Wunderlich. An Analytical Approach to the Partial Scan Problem. *Journal of Electronic Testing*, 1(2):163–174, 1990. (cited on page 77.)

[Lala01a] P. K. Lala. *Self-Checking and Fault Tolerant Digital Design*. Morgan Kaufmann, 2001. (cited on pages 13, 28, and 175.)

[Lala01b] P. K. Lala. *Self-Checking and Fault Tolerant Digital Design*, chapter Fault-Tolerant Design, pages 161–198. Morgan Kaufmann, 2001. (cited on page 31.)

[Lala01c] P. K. Lala. *Self-Checking and Fault Tolerant Digital Design*, chapter Error Detecting and Correcting Codes, pages 15–41. Morgan Kaufmann, 2001. (cited on pages 47 and 115.)

[Lala01d] P. K. Lala. *Self-Checking and Fault Tolerant Digital Design*, chapter Self-Checking Checkers, pages 79–129. Morgan Kaufmann, 2001. (cited on page 100.)

[Laprie85] J.-C. Laprie. Dependable computing and fault-tolerance. In *Proc. the Fifteenth Annual International Symposium on Fault-Tolerant Computing (FTCS-15)*, pages 2–11. Ann Arbor, MI, 1985. (cited on page 28.)

[Laprie91] J. Laprie. *Dependability: Basic Concepts And Terminology*. Springer-Verlag, 1991. (cited on page 28.)

[Larrabee92] T. Larrabee. Test Pattern Generation Using Boolean Satisfiability. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(1):4–15, 1992. (cited on pages 32 and 33.)

[Lavo96] D. B. Lavo, T. Larrabee, and B. Chess. Beyond the Byzantine generals: Unexpected behavior and bridging fault diagnosis. In *Proc. International Test Conference (ITC)*, pages 611–619. Washington, DC, 1996. (cited on page 24.)

[Lehtonen07a] T. Lehtonen, P. Liljeberg, and J. Plosila. Online Reconfigurable Self-Timed Links for Fault Tolerant NoC. *Hindawi VLSI design*, pages 1–13, 2007. (cited on page 48.)

[Lehtonen07b] T. Lehtonen, P. Liljeberg, and J. Plosila. Self-timed NoC links using combinations of fault tolerance methods. In *Proc. IEEE Design Automation and Test in Europe (DATE)*. Nice, France, 2007. (cited on page 48.)

[Lehtonen10] T. Lehtonen, D. Wolpert, P. Liljeberg, J. Plosila, and P. Ampadu. Self-Adaptive System for Addressing Permanent Errors in On-Chip Interconnects. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 18(4):527–540, 2010. (cited on pages 10, 35, 46, and 48.)

[Lin09] S.-Y. Lin, W.-C. Shen, C.-C. Hsu, C.-H. Chao, and A.-Y. Wu. Fault-tolerant router with built-in self-test/self-diagnosis and fault-isolation circuits for 2D-mesh based chip multiprocessor systems. In *Proc. IEEE International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, pages 72–75. Hsin Chu, Taiwan, 2009. (cited on pages 2, 36, 39, 40, 49, and 57.)

[Lindholm08] E. Lindholm, J. Nickolls, S. Oberman, and J. Montrym. NVIDIA Tesla: A Unified Graphics and Computing Architecture. *IEEE Micro*, 28(2):39–55, 2008. (cited on page 1.)

[M-Silva99] J. P. M-Silva and K. A. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, 1999. (cited on page 32.)

[M-Silva00] J. P. M-Silva and K. A. Sakallah. Boolean satisfiability in electronic design automation. In *Proc. Design Automation Conference (DAC)*, pages 675–680. Los Angeles, CA, 2000. (cited on page 32.)

[Maly87] W. Maly. Restricted fault modeling for VLSI testing. In *Proc. Design Automation Conference (DAC)*, pages 173–180. New York, NY, 1987. (cited on pages xiii, 4, and 23.)

[Marinissen02] E. J. Marinissen, R. Kapur, M. Lousberg, T. McLaurin, M. Ricchetti, Y. Zorian, and K. Chakrabarty. On IEEE P1500s Standard for Embedded Core Test. *Journal of Electronic Testing: Theory and Applications (JETTA)*, 18(4/5):365–383, 2002. (cited on page 39.)

[Maxwell93] P. C. Maxwell and R. C. Aitken. Biased voting: a method for simulating CMOS bridging faults in the presence of variable gate logic thresholds. In *Proc. International Test Conference (ITC)*, pages 63–72. Baltimore, MD, 1993. (cited on page 24.)

[Maxwell00] P. Maxwell, I. Hartanto, and L. Bentz. Comparing functional and structural tests. In *Proc. IEEE International Test Conference (ITC)*, pages 400–407. Atlantic City, NJ, 2000. (cited on page 41.)

[Metra08] C. Metra, D. Rossi, M. Omana, A. Jas, and R. Galivanche. Function-inherent code checking: A new low cost on-line testing approach for high performance microprocessor control logic. In *Proc. IEEE European Test Symposium (ETS)*, pages 171–176. Lago Maggiore, Italy, 2008. (cited on page 45.)

[Meyer80] J. F. Meyer. On Evaluating the Performability of Degradable Computing Systems. *IEEE Transactions on Computers*, 100(8):720–731, 1980. (cited on pages 6, 7, and 28.)

[Miczo03] A. Miczo. *Digital logic testing and simulation*. John Wiley & Sons, 2003. (cited on page 75.)

[Mitra00] S. Mitra and E. McCluskey. Which concurrent error detection scheme to choose? In *Proc. IEEE International Test Conference (ITC)*, pages 985–994. Atlantic City, NJ, 2000. (cited on pages 45, 180, and 182.)

[Mohanram03a] K. Mohanram, E. Sogomonyan, M. Gossel, and N. Touba. Synthesis of low-cost parity-based partially self-checking circuits. In *Proc. 9th IEEE On-Line Testing Symposium (IOLTS)*, pages 35–40. Kos Island, Greece, july 2003. (cited on page 46.)

[Mohanram03b] K. Mohanram and N. Touba. Cost-effective approach for reducing soft error failure rate in logic circuits. In *Proc. IEEE International Test Conference (ITC)*, pages 893–901. Charlotte, NC, 2003. (cited on page 46.)

[Moskewicz01] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *Proc. 38th Design Automation Conference (DAC)*, pages 530–535. Las Vegas, NV, 2001. (cited on page 32.)

[Nassif01] S. R. Nassif. Modeling and analysis of manufacturing variations. In *Proc. IEEE Conference on Custom Integrated Circuits*, pages 223–228. San Diego, CA, 2001. (cited on page 43.)

[Ni93] L. M. Ni and P. K. McKinley. A Survey of Wormhole Routing Techniques in Direct Networks. *IEEE Transaction on Computers*, 26(2):62–76, 1993. (cited on pages 22 and 37.)

[Nickolls10] J. Nickolls and W. J. Dally. The GPU Computing Era. *IEEE micro*, 30(2):56–69, 2010. (cited on page 1.)

[Nicolaidis98] M. Nicolaidis and Y. Zorian. On-line Testing for VLSI - A Compendium of Approaches. *Journal of Electronic Testing*, 12(1-2):7–20, 1998. (cited on pages 9 and 99.)

[Palesi10] M. Palesi, S. Kumar, and V. Catania. Leveraging Partially Faulty Links Usage for Enhancing Yield and Performance in Networks-

on-Chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(3):426–440, 2010. (cited on pages 2 and 49.)

[Palesi11] M. Palesi, G. Ascia, F. Fazzino, and V. Catania. Data Encoding Schemes in Networks on Chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30(5):774–786, 2011. (cited on pages 10, 46, and 47.)

[Pande05] P. P. Pande, C. Grecu, A. Ivanov, R. Saleh, and G. De Micheli. Design, Synthesis, and Test of Networks on Chips. *IEEE Design & Test of Computers*, 22(5):404–413, 2005. (cited on page 1.)

[Parikh13] R. Parikh and V. Bertacco. uDIREC: unified diagnosis and reconfiguration for frugal bypass of NoC faults. In *Proc. 46th IEEE/ACM International Symposium on Microarchitecture*, pages 148–159. Davis, CA, 2013. (cited on pages 49, 50, and 57.)

[Park88] E. S. Park, M. Mercer, and T. Williams. Statistical delay fault coverage and defect level for delay faults. In *Proc. International Test Conference (ITC)*, pages 492–499. Washington, DC, 1988. (cited on page 24.)

[Paschalis05] A. Paschalis and D. Gizopoulos. Effective Software-Based Self-Test Strategies for On-line Periodic Testing of Embedded Processors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(1):88–99, 2005. (cited on page 74.)

[Petersén07] K. Petersén and K. Oberg. Toward a scalable test methodology for 2D-mesh network-on-chips. In *Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. Nice, France, 2007. (cited on page 40.)

[Prasad05] M. R. Prasad, A. Biere, and A. Gupta. A Survey of Recent Advances in SAT-Based Formal Verification. *International Journal on Software Tools for Technology Transfer*, 7(2):156–173, 2005. (cited on page 32.)

[Qiaoyan11] Y. Qiaoyan, Z. Meilin, and P. Ampadu. Exploiting inherent information redundancy to manage transient errors in NoC routing arbitration. In *Proc. Fifth IEEE/ACM International Symposium on Networks on Chip (NoCS)*, pages 105–112. Pittsburg, PA, 2011. (cited on pages 43 and 46.)

[Radetzki12] M. Radetzki, C. Feng, X. Zhao, and A. Jantsch. Methods for Fault Tolerance in Networks on Chip. *ACM Computing Surveys*, 44:1–35, 2012. (cited on pages 28 and 35.)

[Raik06] J. Raik, V. Govind, and R. Ubar. An external test approach for network-on-a-chip switches. In *Proc. 15th Asian Test Symposium (ATS)*, pages 437–442. Fukuoka, Japan, 2006. (cited on page 42.)

[Raik07] J. Raik, R. Ubar, and V. Govind. Test configurations for diagnosing faulty links in NoC switches. In *Proc. 12th IEEE European Test Symposium (ETS)*, pages 29–34. Freiburg, Germany, 2007. (cited on pages 42 and 57.)

[Raik09] J. Raik, V. Govind, and R. Ubar. Design-for-Testability-Based External Test and Diagnosis of Mesh-Like Network-on-a-Chips. *IET Computers Digital Techniques*, 3(5):476–486, 2009. (cited on pages 8 and 42.)

[Raik12] J. Raik and V. Govind. Low-area boundary BIST architecture for mesh-like network-on-chip. In *Proc. 15th IEEE International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, pages 95–100. Tallinn, Estonia, 2012. (cited on page 42.)

[Refan08] F. Refan, H. Alemzadeh, S. Safari, P. Prinetto, and Z. Navabi. Reliability in application specific mesh-based NoC architectures. In *Proc. 14th IEEE International On-Line Testing Symposium (IOLTS)*, pages 207–212. Rhodes, Greece, 2008. (cited on pages 30 and 38.)

[Ren14] P. Ren, Q. Meng, X. Ren, and N. Zheng. Fault-tolerant routing for on-chip network without using virtual channels. In *Proc. 51st Annual Design Automation Conference (DAC)*, pages 1–6. San Francisco, CA, 2014. (cited on page 37.)

[Renovell10] M. Renovell, F. Azais, J. Figueras, R. Rodrigues-Montanes, and D. Arumi. *Models in Hardware Testing*, chapter Models for Bridging Defects, pages 159–184. Springer, 2010. (cited on page 24.)

[Richter09] M. Richter and M. Goessel. Concurrent checking with split-parity codes. In *Proc. 15th IEEE International On-Line Testing Symposium (IOLTS)*, pages 159–163. Sesimbra-Lisbon, Portugal, 2009. (cited on page 46.)

[Rossi07] D. Rossi, P. Angelini, and C. Metra. Configurable error control scheme for NoC signal integrity. In *Proc. 13th IEEE International On-Line Testing Symposium (IOLTS)*, pages 43–48. Crete, Greece, 2007. (cited on pages 10, 46, and 101.)

[Roy95] K. Roy, R. Roy, and A. Chatterjee. Stress testing of combinational VLSI circuits using existing test sets. In *Proc. IEEE International Symposium on VLSI Technology, Systems, and Applications*, pages 93–98. Taipei, Taiwan, 1995. (cited on page 11.)

[Rusu08] C. Rusu, C. Grecu, and L. Anghel. Communication aware recovery configurations for networks-on-chip. In *Proc. 14th IEEE International On-Line Testing Symposium (IOLTS)*, pages 201–206. Rhodes, Greece, 2008. (cited on page 35.)

[sat] Sat4j: the boolean satisfaction and optimization library in Java [Online]. Available: www.sat4j.org. (cited on pages 96 and 117.)

[Schaefer78] T. J. Schaefer. The complexity of satisfiability problems. In *Proc. of the tenth annual ACM symposium on Theory of Computing*, pages 216–226. New York, NY, 1978. (cited on page 32.)

[Segura04] J. Segura and C. F. Hawkins. *CMOS Electronics: How It Works, How It Fails*. John Wiley & Sons, 2004. (cited on page 23.)

[Shamshiri11] S. Shamshiri, A. Ghofrani, and K.-T. Cheng. End-to-end error correction and online diagnosis for on-chip networks. In *Proc. IEEE International Test Conference (ITC)*, pages 1–10. Anaheim, CA, 2011. (cited on pages 49 and 50.)

[Smith05] A. Smith, A. Veneris, M. Fahim Ali, and A. Viglas. Fault Diagnosis and Logic Debugging Using Boolean Satisfiability. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(10):1606–1621, 2005. (cited on page 32.)

[Sogomonyan93] E. S. Sogomonyan and M. Goessel. Design of self-testing and on-line fault detection combinational circuits with weakly independent outputs. *Journal of Electronic Testing (JETTA)*, 4:267–281, 1993. (cited on page 111.)

[Stewart06] K. Stewart and S. Tragoudas. Interconnect testing for networks on chips. In *Proc. 24th IEEE VLSI Test Symposium (VTS)*, pages 100–105. Berkeley, CA, 2006. (cited on page 41.)

[Strano11] A. Strano, C. Gómez, D. Ludovici, M. Favalli, M. E. Gomez, and D. Bertozzi. Exploiting network-on-chip structural redundancy for a cooperative and scalable built-in self-test architecture. In *Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. Grenoble, France, 2011. (cited on pages 39 and 50.)

[Tanenbaum03] A. S. Tanenbaum. *Computer Networks*. Pearson Education, 2003. (cited on page 31.)

[Tomita06] E. Tomita, A. Tanaka, and H. Takahashi. The Worst-case Time Complexity for Generating All Maximal Cliques and Computational Experiments. *Journal of Theoretical Computer Science*, 363(1):28–42, 2006. (cited on page 109.)

[Touba93] N. A. Touba and E. J. McCluskey. Logic synthesis for concurrent error detection. Technical report, Stanford, CA, USA, 1993. (cited on page 44.)

[Touba97] N. Touba and E. McCluskey. Logic Synthesis of Multilevel Circuits with Concurrent Error Detection. *IEEE Transaction on CAD*, 16(7):783–789, 1997. (cited on pages 44, 180, and 182.)

[Tran06a] E. N. Tran, V. Kasulasrinivas, and S. Chakravarty. Silicon evaluation of logic proximity bridge patterns. In *Proc. 24th IEEE VLSI Test Symposium (VTS)*, pages 78–83. Berkeley, CA, 2006. (cited on page 95.)

[Tran06b] X.-T. Tran, J. Durupt, F. Bertrand, V. Beroulle, and C. Robach. A DFT architecture for asynchronous networks-on-chip. In *Proc. 11th IEEE European Test Symposium (ETS)*, pages 219–224. Southampton, UK, 2006. (cited on page 40.)

[Tran08] X.-T. Tran, Y. Thonnart, J. Durupt, V. Beroulle, and C. Robach. A design-for-test implementation of an asynchronous network-on-chip architecture and its associated test pattern generation and application. In *Proc. Second ACM/IEEE International Symposium on Networks-on-Chip (NOCS)*, pages 149–158. Newcastle upon Tyne, UK, 2008. (cited on page 40.)

[Tseitin68] G. S. Tseitin. On the Complexity of Derivation in Propositional Calculus. *Studies in constructive mathematics and mathematical logic*, 2(115-125):10–13, 1968. (cited on page 33.)

[Vangal08] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and S. Borkar. An 80-Tile Sub-100-W TeraFLOPS Processor in 65-nm CMOS. *IEEE Journal of Solid-State Circuits*, 43(1):29 –41, 2008. (cited on pages 1 and 68.)

[Vemu08] R. Vemu, A. Jas, J. Abraham, S. Patil, and R. Galivanche. A low-cost concurrent error detection technique for processor control

logic. In *Proc. Design, Automation and Test in Europe (DATE)*, pages 897–902. Munich, Germany, 2008. (cited on page 46.)

[Vigrass10] W. J. Vigrass. Calculation of semiconductor failure rates. *Harris Semiconductor*, 2010. (cited on page 127.)

[Vitkovskiy10] A. Vitkovskiy, V. Soteriou, and C. Nicopoulos. A fine-grained link-level fault-tolerant mechanism for networks-on-chip. In *Proc. IEEE International Conference on Computer Design (ICCD)*, pages 447–454. Amsterdam, Netherlands, 2010. (cited on page 48.)

[Vitkovskiy12] A. Vitkovskiy, V. Soteriou, and C. Nicopoulos. A Dynamically Adjusting Gracefully Degrading Link-Level Fault-Tolerant Mechanism for NoCs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 31(8):1235–1248, 2012. (cited on page 48.)

[Wang06a] L.-T. Wang, C.-W. Wu, and X. Wen. *VLSI Test Principles and Architectures: Design for Testability*. Academic Press, 2006. (cited on page 26.)

[Wang06b] L.-T. Wang, C.-W. Wu, and X. Wen. *VLSI Test Principles and Architectures: Design for Testability*, chapter Design for Testability, pages 37–104. Academic Press, 2006. (cited on pages 26, 27, and 40.)

[Wang06c] L.-T. Wang, C.-W. Wu, and X. Wen. *VLSI Test Principles and Architectures: Design for Testability*, chapter Logic Diagnosis, pages 397–461. Academic Press, 2006. (cited on page 39.)

[Wang10] L.-T. Wang, C. E. Stroud, and N. A. Touba. *System-on-Chip Test Architectures: Nanometer Design for Testability*, chapter Fault-Tolerant Design, pages 123–171. Morgan Kaufmann, 2010. (cited on page 29.)

[Williams83] T. W. Williams and K. P. Parker. Design for testability - a survey. In *Proceedings of the IEEE*, volume 71, pages 98–112. 1983. (cited on page 26.)

[Worm05] F. Worm, P. Ienne, P. Thiran, and G. De Micheli. A Robust Self-Calibrating Transmission Scheme for On-Chip Networks. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 13(1):126–139, 2005. (cited on page 38.)

[Wunderlich98] H.-J. Wunderlich. BIST for Systems-on-a-Chip. *INTEGRATION, the VLSI journal*, 26(1):55–78, 1998. (cited on page 40.)

[Wunderlich05] H.-J. Wunderlich. From embedded test to embedded diagnosis. In *Proc. 10th IEEE European Test Symposium (ETS)*, pages 216–221. Tallinn, Estonia, 2005. (cited on page 26.)

[Wunderlich10] H.-J. Wunderlich and S. Holst. *Models in Hardware Testing*, chapter Generalized Fault Modeling for Logic Diagnosis, pages 159–184. Springer, 2010. (cited on pages 24, 25, 58, and 85.)

[Yanamandra10] A. Yanamandra, S. Eachempati, N. Soundararajan, V. Narayanan, M. Irwin, and R. Krishnan. Optimizing power and performance for reliable on-chip networks. In *Proc. 15th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 431–436. Taipei, Taiwan, 2010. (cited on pages 9, 35, and 46.)

[Yu11] Q. Yu, M. Zhang, and P. Ampadu. Exploiting inherent information redundancy to manage transient errors in NoC routing arbitration. In *Proc. Fifth IEEE/ACM International Symposium on Networks on Chip (NoCS)*, pages 105–112. Pittsburg, PA, 2011. (cited on pages 30 and 38.)

[Zeng99] C. Zeng, N. Saxena, and E. J. McCluskey. Finite state machine synthesis with concurrent error detection. In *Proc. IEEE International Test Conference (ITC)*, pages 672–679. Atlantic City, NJ, 1999. (cited on page 175.)

[Zhang97] H. Zhang. SATO: An efficient prepositional prover. In *Automated Deduction–CADE-14*, pages 272–275. Springer, 1997. (cited on page 32.)

[Zhang06] M. Zhang, S. Mitra, T. M. Mak, N. Seifert, N. Wang, Q. Shi, K. S. Kim, N. Shanbhag, and S. Patel. Sequential Element Design With Built-In Soft Error Resilience. *IEEE Transaction on VLSI Systems*, 14(12):1368–1378, 2006. (cited on page 114.)

[Zhang08a] Y. Zhang, H. Li, and X. Li. Reliable network-on-chip router for crosstalk and soft error tolerance. In *Proc. 17th Asian Test Symposium (ATS)*, pages 438–443. Sapporo, Japan, 2008. (cited on pages 9, 35, and 46.)

[Zhang08b] Z. Zhang, A. Greiner, and S. Taktak. A reconfigurable routing algorithm for a fault-tolerant 2D-mesh network-on-chip. In *Proc. 45th ACM/IEEE Design Automation Conference (DAC)*, pages 441–446. Anaheim, CA, 2008. (cited on pages 2, 37, 57, 68, and 124.)

[Zhang10] Z. Zhang, A. Greiner, and M. Benabdenbi. Fully distributed initialization procedure for a 2D-mesh NoC, including off-line BIST and partial deactivation of faulty components. In *Proc. IEEE 16th International On-Line Testing Symposium (IOLTS)*, pages 194–196. Corfu Island, Greece, 2010. (cited on pages 36 and 40.)

[Zheng10] Y. Zheng, H. Wang, S. Yang, C. Jiang, and F. Gao. Accelerating strategy for functional test of NoC communication fabric. In *Proc. 19th IEEE Asian Test Symposium (ATS)*, pages 224–227. Shanghai, China, 2010. (cited on page 42.)

[Zhou06] J. Zhou and H.-J. Wunderlich. Software-based self-test of processors under power constraints. In *Proc. Design, Automation and Test in Europe (DATE)*, pages 430–435. Munich, Germany, 2006. (cited on page 74.)

Appendix A

Definitions

A.1 Equivalent and Nonequivalent Literals

In this section, we illustrate some definitions, which are used in Chapters 5 and 6 while constructing the respective SAT instances in Conjunctive Normal Form (CNF) for test pattern generation and fault-secure synthesis.

Definition A.1.1. *(CNF of equivalence of literals)* Two literals l_1 and l_2 are equivalent, denoted by $l_1 \Leftrightarrow l_2$, if they get the same value. It is represented in CNF as:

$$\mathrm{CNF}(l_1 \Leftrightarrow l_2) = (l_1 \vee \bar{l_2}) \wedge (\bar{l_1} \vee l_2)\,.$$

Definition A.1.2. *(equivalence of literal sets)* Let A and B be two ordered sets of n literals, $A = (a_1, a_2, \ldots, a_n)$ and $B = (b_1, b_2, \ldots, b_n)$. A and B are equivalent, denoted by $A \overset{*}{\Leftrightarrow} B$, iff:

$$\forall i,\, 1 \leq i \leq n: \quad a_i \Leftrightarrow b_i\,.$$

Definition A.1.3. *(CNF of equivalent literal sets)* The CNF representation of two equivalent sets of literals A and B of size n is the conjunction of the CNF clauses of the equivalent literal pairs as:

$$\mathrm{CNF}(A \overset{*}{\Leftrightarrow} B) = \bigwedge\nolimits_{i=1}^{n} CNF(a_i \Leftrightarrow b_i)\,.$$

Definition A.1.4. *(CNF of nonequivalent literals)* Two literals l_1 and l_2 are nonequivalent, denoted by $l_1 \not\Leftrightarrow l_2$, if they get opposite values. It is represented in CNF as:

$$\mathrm{CNF}(l_1 \not\Leftrightarrow l_2) = (l_1 \vee l_2) \wedge (\bar{l_1} \vee \bar{l_2})\,.$$

Definition A.1.5. *(nonequivalent literal sets)* Let A and B be two ordered sets of n literals, $A = (a_1, a_2, \ldots, a_n)$ and $B = (b_1, b_2, \ldots, b_n)$. These sets of literals A and B are nonequivalent, denoted by $A \overset{*}{\not\Leftrightarrow} B$, iff:

$$\exists i,\, 1 \leq i \leq n : \quad a_i \not\Leftrightarrow b_i\,.$$

Definition A.1.6. *(CNF of nonequivalent literal sets)* For $\{m_1, \ldots, m_n\}$ being a set of free literals, the CNF representation of two nonequivalent sets of literals A and B of size n is defined as:

$$\mathrm{CNF}(A \overset{*}{\not\Leftrightarrow} B) = \bigwedge\nolimits_{i=1}^{n} CNF(\mathrm{XOR}, \{a_i, b_i\}, \{m_i\}) \wedge \bigvee\nolimits_{i=1}^{n} m_i\,.$$

Appendix B

Simulation Results

B.1 Connectivity

Our further investigation for the connectivity of the network with the port-accurate and switch-accurate diagnosis methods for larger switches correlate with the results in section 7.2. The connectivity result in a 20×20 network in which the 32-bit switches are employed is depicted in Table B.1.

In the 32-bit switch, the probability of which the fault causes entire switch deactivation is around 27% (according to the diagnosis result in Chapter 4, Table 4.2). Up to 3 faults, the network with switch-accurate (SA) diagnosis loses only one of its cores for every single fault and preserves the connectivity among the rest. However, the network with port-accurate (PA) diagnosis can improve the connectivity up to 399 cores. After that and up to 11 faults, the network with PA diagnosis has lost almost a single core for every injected fault. This is not the case for the other network which has lost 40 cores in the presence of 11 faults for example.

With 20 faults, the network with PA diagnosis can preserve the connectivity among 368 cores which is noticeably higher than 261 connected cores in the network with SA diagnosis.

Further connectivity investigations have been performed for 8×8 network (Fig. B.1) and 12×12 network (Fig. B.2). Without regard to the network size, the net-

Table B.1: Number of linked cores (averaged over 100 defect conditions) in a 20x20 NoC, with port-accurate (PA) and switch-accurate (SA) diagnosis

# faults	32-bit switch	
	PA Diagnosis	SA Diagnosis
1	399.74	399.00
2	399.45	398.00
3	399.22	397.03
4	395.05	392.29
5	398.39	391.47
7	396.18	391.77
9	390.36	372.72
11	390.12	360.01
13	376.32	325.75
15	384.75	314.82
17	373.66	277.55
20	368.08	261.87

work diagnosis approaches are quite effective for preserving the connectivity among the cores. In the 8×8 network, since the number of switches is less, the connectivity of the network with SA diagnosis drops down faster as the number of faults increases. With 3 faults, the 8×8 network has already lost around 10% of its cores. In the 12×12 network, the connectivity remains more than 95% in the presence of 5 faults. In both networks, the PA diagnosis can be employed to improve the connectivity.

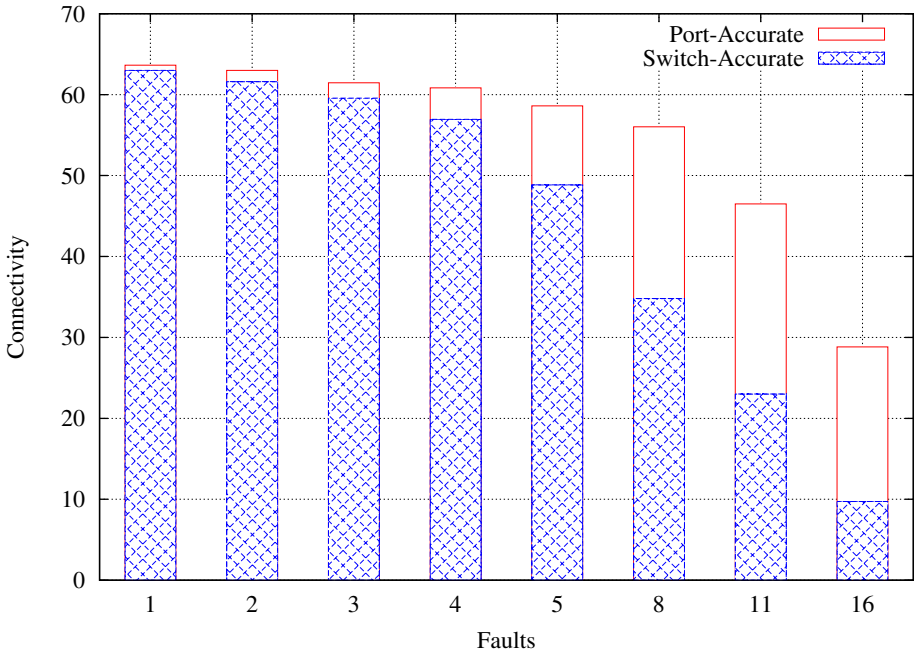


Figure B.1: 8×8 2D mesh network: Connectivity

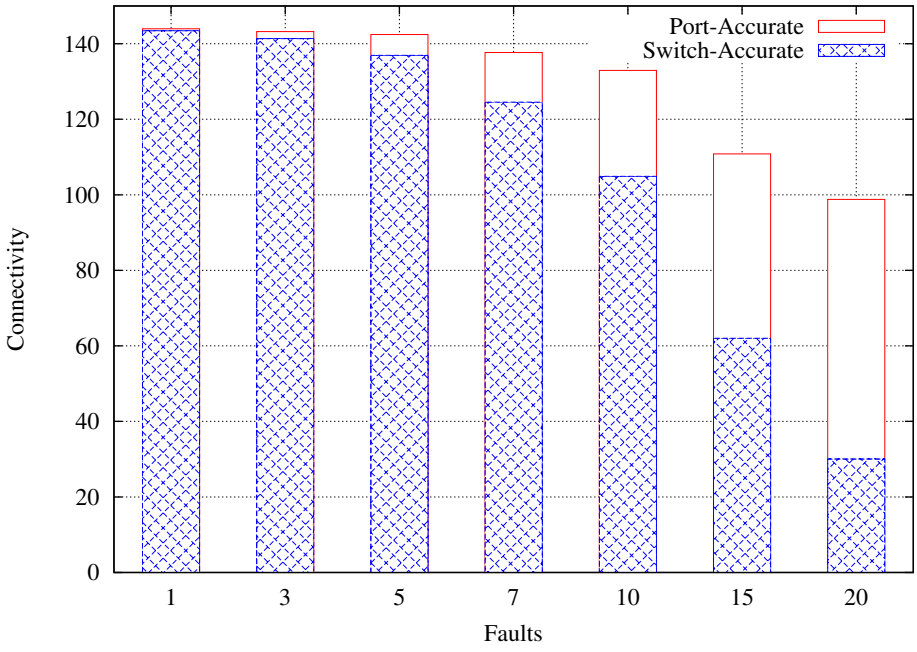


Figure B.2: 12×12 2D mesh network: Connectivity

B.1.1 Network Reliability

Figure B.3 depicts the reliability of the network according to Eq. (2.7) in three years operational time period, considering switch-accurate (SA) and port-accurate (PA) diagnosis. In this experiment, we set $\lambda = 20 \times 10^{-9}$ and $N = 100$, and we compute the result for PA diagnosis once for the network with 12-bit switches and once with 32-bit switches. As expected, when using port-accurate diagnosis, the probability of which the system can continue its intended operation with all of its cores is higher than the case with switch-accurate diagnosis. The reliability improvement by incorporating PA diagnosis is even higher when larger switches are employed. This is due to the fact that in a switch with wider flits, the probability of which a fault disrupts only a single port is higher as revealed by the diagnosis results in section 4.5.

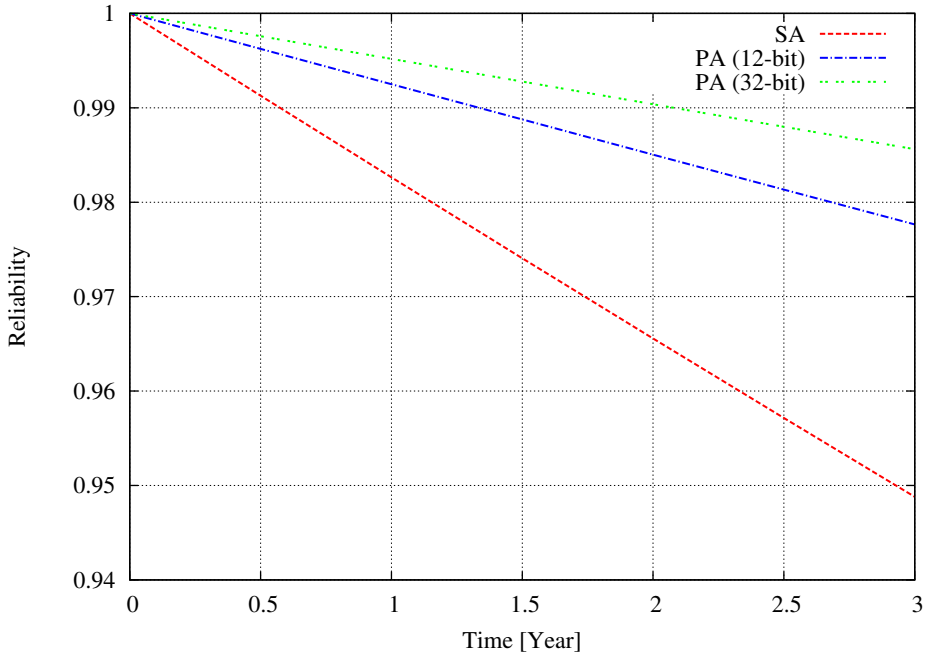


Figure B.3: Reliability of the network with switch-accurate (SA) and port-accurate (PA) diagnosis

B.2 Communication Performance

We have conducted the same set of experiments explained in section 7.3.1 with the 32-bit switches and compute the retransmission rate according to Eq. (7.4). Figure B.4 compares the average retransmission rate in the presence of various number of faults in the 20×20 network with a fixed $load_{core} = 14.5\%$. Here, we observe that with a port-accurate diagnosis method, which enables the use of degraded switches, the retransmission rate is decreased. As any retransmission imposes extra traffic on the network, it will degrade the overall performance.

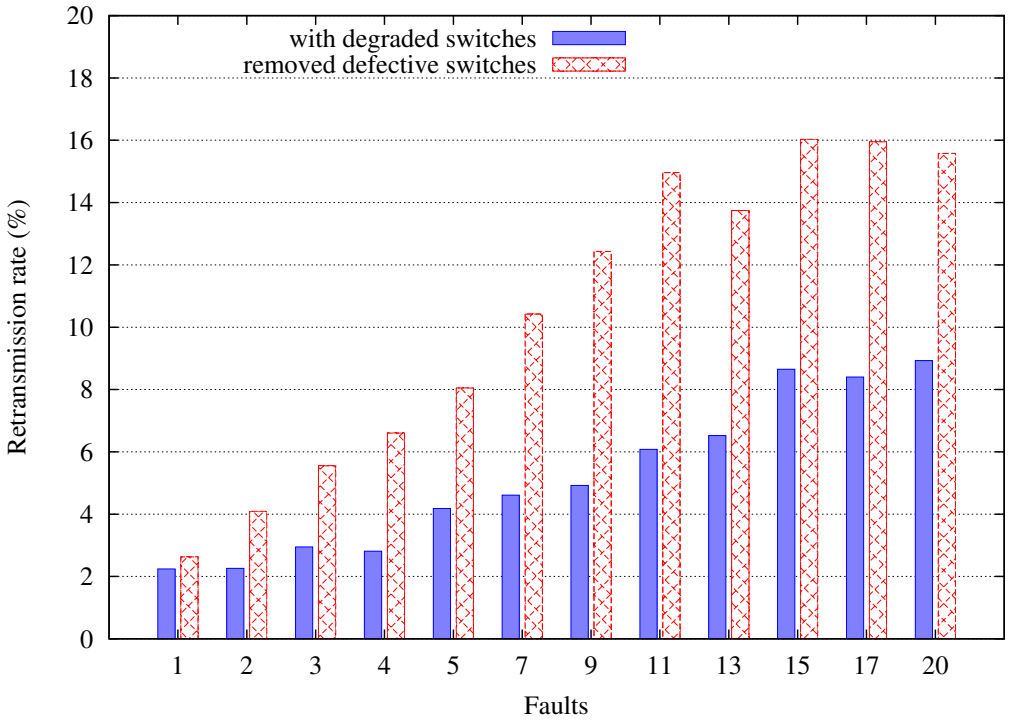


Figure B.4: Average retransmission rate - fault count under load$_{\text{core}}$=14.5%, 32-bit switches

Figure B.5 compares the retransmission rate under various network loads with 9 injected faults. We observe the same reduction in the retransmission rate by using degraded switches in the network. The retransmission rate reduction is more significant under higher network loads.

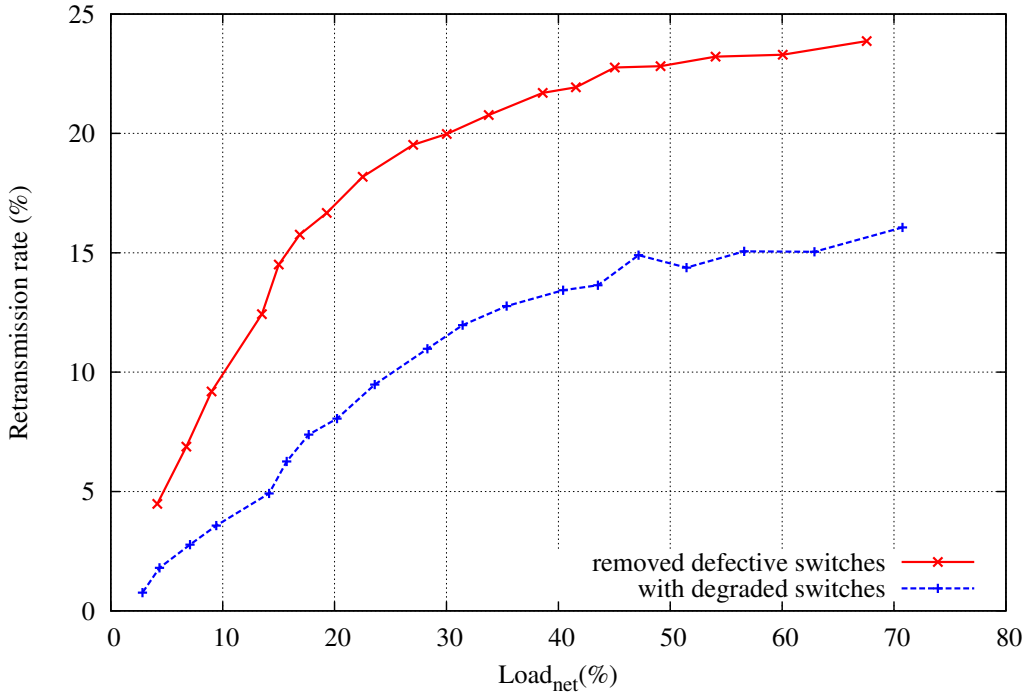


Figure B.5: Retransmission rate under load with 32-bit switch

Appendix C

Results of Fault-Secure Synthesis for Benchmark Circuits

The proposed method for fault-secure synthesis of the critical region of the switch is applied to benchmark circuits of ISCAS'85, ISCAS'89, ITC'99, and MCNC and the area cost is compared with previous work. As the proposed method does not modify the original circuit, for non-redundant circuits, the generated fault-secure structure, is totally self-checking.

C.1 Area Cost of Fault-Secure Synthesis

The area overhead of the fault-secure circuit consists of the parity prediction logic, the dual-rail checkers [Lala01a], the parity trees to generate the parity bits from functional outputs, and the hold registers as shown in Fig. C.1. Hold registers must be added [Zeng99] for the self-checking structure (Fig. C.1) to avoid a longer critical path due to the checker logic.

The prediction logic and the checker are synthesized using Synopsys Design Compiler, optimized for area. The target library lsi10k is constrained to two-input gate primitives. The cell area of a two input NAND gate is 1 area unit.

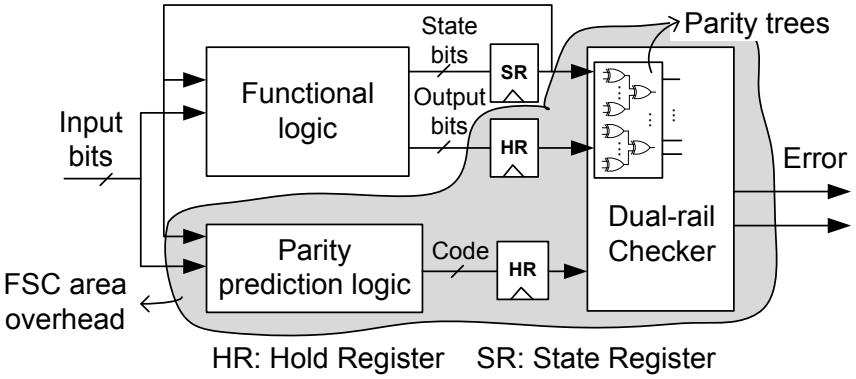


Figure C.1: Area overhead of the fault-secure circuit (FSC)

Tables C.1 and C.2 show the results for the ISCAS'85 and ISCAS'89 circuits with less than 15K gates. The circuit c6288 has been excluded as it is a multiplier for which efficient arithmetic coding techniques exist. The first five columns list the circuit name, the number of inputs and outputs (primary plus pseudo primary), the size in cell area units, and the number of collapsed faults. The column *DWC* reports the cell area overhead of the duplicated system, which consists of a copy of the circuit as predictor.

The remainder of each table reports the results of the proposed method. The number of parity groups after topological analysis is given in column *Groups*. The column *SDCs* shows the number (percentage) of faults which cause silent data corruption after topological analysis. Our investigation shows that after topological analysis, for every testable fault in the circuit there exists at least one input pattern for which the functional outputs differ and a wrong parity bit is generated. In other words, by means of a few parity bits the self-testing property for all the benchmark circuits is achieved. Furthermore, we observe that typically only a low percentage of faults still cause silent data corruption (SDC). This result emphasizes the importance of the topological analysis to reduce the effort of the formal fault-secureness analysis and group splitting.

The results of the final fault-secure circuits are given in columns 9 to 12: Column *Parity bits* shows the number of parity groups after splitting. The next column gives the cell area overhead of the fault-secure circuit. For comparison with the duplication

method, we list the *relative area cost* κ of the proposed method with respect to DWC:

$$\kappa = \frac{\text{Fault-secure circuit area overhead}}{\text{Duplication with comparison (DWC) area overhead}} \tag{C.1}$$

For $\kappa < 1$, the proposed parity-based synthesis method incurs less area overhead than DWC, which is the case for all ISCAS'89 circuits. For two circuits of ISCAS'85 benchmarks (c449, c1355) the area overhead of the fault-secure structure is equal to the area overhead of duplication with comparison. In this case, the number of parity bits is equal to the number of outputs. The weighted average of κ with regard to the circuit size over ISCAS'85 benchmarks is 0.77.

The area saving depends on the circuit size and the structure. The weighted average of κ with regard to the circuit size over all ISCAS'89 benchmark circuits (up to 15K gates) is 0.56, i.e. only 56% of the area of DWC is required. To the best of our knowledge, this is the first method able to generate area-efficient fault-secure circuits for medium size circuits (up to 15K gates) in reasonable time. For most circuits, the runtime is less than a few minutes. For large circuits the runtime increases due to a higher number and more complex SAT checks. For s15850, the number of SAT checks increases by a factor of 3 compared to s13207 which causes higher runtime.

Table C.1: Results for ISCAS85 benchmark circuits

Circuit					DWC	Proposed Method					
Name	PIs + PPIs	POs + PPOs	Cell area	Faults	Cell area overhead	Groups	SDCs	Parity bits	Cell area overhead	**Relative cost** κ	Time (s)
c499	41	32	244	858	1160	4	165 (19%)	32	1160	1.00	1381
c880	60	26	661	1042	1403	5	318 (31%)	14	1151	0.82	96
c1355	41	32	660	1618	1576	4	292 (18%)	32	1576	1.00	4251
c1908	33	25	1193	2100	1906	3	152 (7%)	10	1591	0.83	895
c2670	233	140	1819	3226	5867	4	248 (9%)	10	3173	0.54	538
c3540	50	22	2788	3332	3414	4	1395 (42%)	17	3309	0.97	14736
c5315	178	123	4274	6246	7829	8	685 (11%)	22	5708	0.73	9895
c7552	207	108	5078	8284	8198	6	842 (10%)	9	6119	0.75	3987

Table C.2: Results for ISCAS89 benchmark circuits

Circuit					DWC	Proposed Method					
Name	PIs + PPIs	POs + PPOs	Cell area	Faults	Cell area overhead	Groups	SDCs	Parity bits	Cell area overhead	**Relative cost** κ	Time (s)
s27	7	4	37	34	93	3	0 (0%)	3	45	0.48	0
s298	17	20	378	392	722	3	55 (14%)	11	533	0.74	4
s344	24	26	349	396	851	4	56 (14%)	10	515	0.61	15
s349	24	26	351	402	853	4	56 (14%)	10	517	0.61	14
s382	24	27	459	492	894	5	38 (8%)	9	516	0.58	6
s400	24	27	470	520	905	5	38 (7%)	9	527	0.58	7
s444	24	27	480	720	915	5	41 (6%)	10	558	0.61	12
s510	25	13	390	614	659	5	30 (5%)	8	554	0.84	2
s526	24	27	667	712	1102	4	73 (10%)	10	745	0.68	58
s641	54	43	740	598	1671	4	160 (27%)	14	1062	0.64	129
s713	54	42	774	716	1676	4	186 (26%)	9	983	0.59	24
s820	23	24	889	1104	1493	3	36 (3%)	5	1094	0.73	24
s953	45	52	786	1208	1818	6	127 (11%)	14	1020	0.56	154
s1196	32	32	1076	1386	1704	6	297 (21%)	22	1494	0.88	412
s1238	32	32	1108	1514	1736	5	263 (17%)	19	1463	0.84	494
s1423	91	79	1617	1680	2712	5	535 (32%)	21	1494	0.55	2881
s1488	14	25	1619	1710	2236	5	73 (4%)	13	1984	0.89	74
s1494	14	25	1629	1734	2246	5	73 (4%)	14	2015	0.90	79
s5378	214	228	5249	5474	8949	7	901 (16%)	41	5062	0.57	5342
s9234	247	250	9473	7760	13063	8	1622 (21%)	27	8380	0.64	4451
s13207	700	790	16179	12024	28368	6	1012 (8%)	21	12204	0.43	6394
s15850	611	684	17646	14016	27898	9	2957 (21%)	35	14289	0.51	122609

C.2 Comparison to Related Work

Most of the previous parity-based methods were applied to small size circuits only. We compare the area cost of the proposed method with the reported results of the parity-based self-checking methods in [Almukhaizim06] and [Touba97, Mitra00].

In [Almukhaizim06], results for an error detection table and entropy based method are reported for MCNC and smaller ITC'99 circuits. Table C.3 lists the result of our synthesis scheme for ITC benchmark circuits and compares it against those available by reference [Almukhaizim06]. For comparison, the relative cost κ' is computed according to eq. (C.1) using the area overhead of the self-checking circuit and the area overhead of duplication as reported in [Almukhaizim06]. The last two columns of Table C.3 list the number of parity bits and κ' of the method in [Almukhaizim06]. The table also shows our results for much larger ITC'99 circuits which were not investigated in [Almukhaizim06].

The weighted average of κ over seven circuits in comparison with [Almukhaizim06] is 0.51, that is 14% less than [Almukhaizim06]. In addition, our method is also applicable to much larger circuits with a weighted average value of κ of 0.58.

The proposed method incurs significantly less area overhead as compared to duplication with comparison (DWC). For some circuits only half of the DWC area is required for fault-secureness with respect to all combinational and transition faults on a single node (including all stuck-at faults). This area saving needs to be traded off against the potentially higher coverage of multiple faults offered by DWC.

Table C.3: Results for ITC'99 benchmark circuits

Circuit					DWC	Proposed Method						[Almukhaizim06]	
Name	PIs + PPIs	POs + PPOs	Cell area	Faults	Cell area overhead	Groups	SDCs	Parity bits	Cell area overhead	**Relative cost** κ	Time (s)	Parity bits	Relative cost κ'
b01	7	7	62	136	213	3	2 (1%)	4	150	0.70	1	5	0.53
b02	5	5	34	72	135	3	0 (0%)	3	93	0.69	2	3	0.38
b03	34	34	418	478	912	5	80 (16%)	10	408	0.45	95	4	0.83
b04	77	74	1381	1888	1803	4	256 (13%)	23	1140	0.63	636	-	-
b05	35	60	1708	2748	2892	5	966 (35%)	29	2241	0.77	6152	-	-
b06	11	15	140	176	419	3	11 (6%)	5	209	0.50	1	-	-
b07	50	57	893	1238	1750	5	347 (26%)	11	784	0.45	1144	6	0.58
b08	30	25	395	528	772	4	66 (12%)	12	499	0.65	12	5	0.60
b09	29	29	424	476	805	4	169 (35%)	6	322	0.40	56	7	0.40
b10	28	23	399	596	782	4	88 (14%)	12	551	0.70	12	7	0.67
b11	38	37	1180	1876	1745	6	223 (11%)	18	1346	0.77	355	-	-
b12	126	127	2394	3310	4129	7	449 (13%)	19	1861	0.45	457	-	-
b13	63	63	796	1002	1763	4	111 (11%)	14	734	0.42	94	-	-

In [Mitra00, Touba97], totally self-checking circuits are synthesized using multiple parity codes over groups of outputs. The shared logic is replicated when required and therefore the original circuit is modified. The experiments use the MCNC benchmark circuits, which are relatively small and have a few outputs. For such circuits, parity based fault-secureness does not save much area compared to duplication with comparison (DWC) [Mitra00], and DWC itself is not very expensive. As reported in Table C.4, our method in average imposes a slightly lower overhead compared to [Touba97, Mitra00] even without requiring the circuit modification.

The authors of [Touba97, Mitra00] have used a special synthesis tool. For fair comparison, we computed the relative cost κ' according to Eq. (C.1) based on the *reported* area cost of duplication and *reported* area cost of the related work.

Table C.4 declares that for the large circuits like rd73 and rd84 which have respectively 3 and 5 output ports, the previous parity based method in [Mitra00] imposes about 1.5 times more overhead as compared to duplication. But since our method does not modify the original circuit, it ends up in a solution similar to duplication. For other circuits, the overhead of parity based methods are almost similar.

Table C.4: MCNC benchmark circuits, area comparison

Circuit				Previous Work		Proposed Method		
Name	PIs	POs	Cell area	Parity bits	Relative cost κ'	Parity bits	Area overhead	**Relative cost κ**
misex2	25	18	55	2	0.73 [1]	11	159	0.75
bw	5	28	68	8	0.66 [1]	9	303	0.61
alu	14	8	5908	8	1.00 [1]	7	5987	1.00
misex1	10	10	217	-	0.89 [2]	4	257	0.85
squar5	7	11	412	-	0.92 [2]	4	452	0.88
rd73	7	3	921	-	1.44 [2]	3	925	1.00
sao2	12	6	936	-	0.99 [2]	4	976	0.99
rd84	8	5	1898	-	1.51 [2]	4	1908	1.00
z5xp1	9	13	2908	-	1.02 [2]	5	2961	0.98

[1] [Touba97]

[2] [Mitra00]

Index

Publications of the Author

- A. Dalirsani, N. Hatami, M. E. Imhof, M. Eggenberger, G. Schley, M. Radetzki, and H.-J. Wunderlich. On covering structural defects in NoCs by functional tests. In *Proc. 23rd IEEE Asian Test Symposium (ATS)*, pages 87–92. Hangzhou, China, 2014.

- A. Dalirsani, M. A. Kochte, and H.-J. Wunderlich. Area-efficient synthesis of fault-secure NoC switches. In *Proc. 20th IEEE International On- Line Testing Symposium (IOLTS)*, pages 13–18. Catalunya, Spain, 2014.

- A. Dalirsani, M. E. Imhof, and H.-J. Wunderlich. Structural software-based self-test of network-on-chip. In *Proc. IEEE VLSI Test Symposium (VTS)*, pages 1–6. Napa, CA, 2014.

- A. Dalirsani, M. A. Kochte, and H.-J. Wunderlich. SAT-based code synthesis for fault-secure circuits. In *Proc. IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pages 39–44. New York City, NY, 2013.

- A. Dalirsani, S. Holst, M. Elm, and H.-J. Wunderlich. Structural Test and Diagnosis for Graceful Degradation of NoC Switches. *Journal of Electronic Testing*, 28(6):831–841, 2012.

- A. Dalirsani, S. Holst, M. Elm, and H. Wunderlich. Structural test for graceful degradation of NoC switches. In *Proc. IEEE European Test Symposium (ETS)*, pages 183–188. Trondheim, Norway, 2011.

- A. Dalirsani, S. Holst, M. Elm, and H. Wunderlich. Structural test for graceful degradation of NoC switches. In *Proc. 23. GI/GMM/ITG-Workshop Testmethoden und Zuverlässigkeit von Schaltungen und Systemen (TuZ)*, pages 33–38. Passau, Germany, 2011.

- A. Dalirsani and Z. Navabi. Testing of routers in NoC mesh architecture using routers functionality. In *Proc. East-West Design and Test Symposium (EWDTS)*, pages 570–575. Yerevan, Armenia, 2007.

- A. Dalirsani, M. Hosseinabady, and Z. Navabi. An analytical model for reliability evaluation of NoC architectures. In *Proc. 13th IEEE International On-Line Testing Symposium (IOLTS)*, pages 49–56. Crete, Greece, 2007.

- M. Hosseinabady, A. Dalirsani, and Z. Navabi. Using the inter- and intra-switch regularity in NoC switch testing. In *Proc. Design, Automation and Test in Europe (DATE)*, pages 361–366. Nice, France, 2007.

Curriculum Vitae of the Author

Atefe Dalirsani started her bachelor study in computer engineering in the Faculty of Electrical and Computer Engineering at the University of Tehran, Iran, after achieving a top rank in the Iran's national university entrance exam in 2001. She joined then the graduate program at the University of Tehran and obtained her master's degree in computer architecture in 2007. The focus of her research was test and reliability of Network-on-Chip architectures. During her master study, the author worked as a teaching and research assistance contributing in design, development, and verification of a multi-language mixed-signal simulator, consisting of HDL language compilers and simulators. Before starting her PhD, she has gained working experience by getting involved in several projects such as design and development of automatic car gearbox controllers, voice over IP applications, and digital video broadcasting components.

The author has joined the Institute of Computer Architecture and Computer Engineering at the University of Stuttgart, Germany in 2009 and has been working as a research and teaching assistant under the supervision of Prof. Hans-Joachim Wunderlich. She has been involved in the research project "ROCK: Robust On-Chip Communication through Hierarchical Online-Diagnosis and Reconfiguration" funded by the German Research Foundation (DFG), and contributed in writing project proposals. She supported graduate courses, supervised students in several seminars, master's theses, and student projects. Pursuing her PhD at the University of Stuttgart, the author has received four years scholarship from the German Academic Exchange Service (DAAD). Her research interests include design and test of Systems-on-Chip, multi-level test and diagnosis, design for testability, reliability and fault tolerance.

Declaration

All the work contained within this thesis, except where otherwise acknowledged, was solely the effort of the author. At no stage was any collaboration entered into with any other party.

Atefe Dalirsani