TALLINN UNIVERSITY OF TECHNOLOGY
DOCTORAL THESIS
12/2018

Dependability Improvements of NoC-Based Systems

BEHRAD NIAZMAND

TTÜ PRESS

TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies
Department of Computer Systems
This dissertation was accepted for the defence of the degree of Doctor of Philosophy in Computer and Systems Engineering on 11/04/2018.

Supervisor: Prof. Gert Jervan, PhD
Department of Computer Systems
Tallinn University of Technology
Tallinn, Estonia

Co-supervisor: Prof. Jaan Raik, PhD
Department of Computer Systems
Tallinn University of Technology
Tallinn, Estonia

Opponents: Prof. Dr.-Ing. Thilo Pionteck, PhD
Institut für Informations-und Kommunikationstechnik-IIKT
Otto-von-Guericke- Universität Magdeburg
Magdeburg, Germany

Prof. Masoud Daneshtalab, PhD
Embedded Systems,
Mälardalen University College,
Västerås, Sweden

Defence of the thesis: 16/05/2018, Tallinn

Declaration:
Hereby I declare that this doctoral thesis, my original investigation and achievement, submitted for the doctoral degree at Tallinn University of Technology, has not been previously submitted for doctoral or equivalent academic degree.

BEHRAD NIAZMAND **BEHRAD NIAZMAND**

signature

ISSN 2585-6898 (publication)
ISBN 978-9949-83-230-9 (publication)
ISSN 2585-6901 (PDF)
ISBN 978-9949-83-231-6 (PDF)

TALLINNA TEHNIKAÜLIKOOL
DOKTORITÖÖ
12/2018

Töökindluse parandamine kiipvõrkudel põhinevates süsteemides

BEHRAD NIAZMAND

Table of Contents

List of Publications

List of publications included to the thesis

The work of this thesis is based on the following publications:

A **P. Saltarelli, B. Niazmand, R. Hariharan, J. Raik, G. Jervan and T. Hollstein,** "Automated Minimization of Concurrent Online Checkers for Network-on-Chips", 10th International Symposium on Re-configurable Communication-centric Systems-on-Chip (ReCoSoC 2015), June 29- July 1, 2015, Bremen, Germany.

B **P. Saltarelli, B. Niazmand, J. Raik, R. Hariharan, V. Govind, T. Hollstein and G. Jervan,** "A framework for combining concurrent checking and on-line embedded test for low-latency fault detection in NoC routers", 9th International Symposium on Networks-on-Chip (NOCS) 2015, September 28-30, 2015, Vancouver, Canada.

C **B. Niazmand, S. P. Azad, J. Flich, J. Raik, G. Jervan and T. Hollstein**, "Logic-based implementation of fault-tolerant routing in 3D network-on-chips," 2016 Tenth IEEE/ACM International Symposium on Networks-on-Chip (NOCS), Nara, 2016, pp. 1-8.

D **S. P. Azad, B. Niazmand, A. K. Sandhu, J. Raik, G. Jervan and T. Hollstein,** "Automated area and coverage optimization of minimal latency checkers," 2017 22nd IEEE European Test Symposium (ETS), Limassol, 2017, pp. 1-2.

E **S. P. Azad, B. Niazmand, K. Janson, N. George, A. S. Oyeniran, T. Putkaradze, A. Kaur, J. Raik, G. Jervan, R. Ubar, T. Hollstein,** "From Online Fault Detection to Fault Management in Network-on-Chips: A Ground-Up Approach", 2017 The IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS), Apr 19-21 2017, Dresden, Germany.

Other related publications

Other related publications that the author has had contribution to them are as follows:

F **B. Niazmand, R. Hariharan, V. Govind, G. Jervan, T. Hollstein and J. Raik**, "Extended Checkers for Logic-Based Distributed Routing in Network-on-Chips", The 14th Biennial Baltic Electronics Conference (BEC), October 6-8, 2014, Laulasmaa, Estonia.

G **R. Hariharan**, **B. Niazmand, T. Hollstein, J. Raik and G. Jervan,** "Extended Checkers for Control Part of Routers in Network-on-Chips", MEDIAN 2015, Manufacturable and Dependable Multicore Architectures at Nanoscale, DATE Friday Workshop, March 13, 2015, Grenoble, France (Workshop paper, not indexed).

H **P. Saltarelli, B.Niazmand, J. Raik, R. Hariharan, G. Jervan and T. Holistein,** "A Framework for Comprehensive Automated Evaluation of Concurrent Online Checkers", Euromicro Conference on Digital System Design (DSD) 2015, August 26-28, 2015, Funchal, Madeira, Portugal.

Other publications

Other publications that the author has had contribution to them are as follows:

I **S. P. Azad, B. Niazmand, T. Kogge, K. Janson, J. Raik, G. Jervan and T. Hollstein,** "An Enumeration of All 2D-Turn Models and Their Characteristics in Network-on-Chips", 2017 International Symposium on Circuits and Systems (ISCAS), May 28-31 2017, Baltimore, MD, USA.

J **S. P. Azad, B. Niazmand, J. Raik, G. Jervan and T. Hollstein**, "Holistic Approach for Fault-Tolerant Network-on-Chip based Many-Core Systems", 2016 2nd International Workshop on Dynamic Resource Allocation and Management in Embedded, High Performance and Cloud Computing (DREAMCloud), Co-Located with HiPEAC 2016 Conference, Prague, Czech Republic, Jan 2016 (Indexed on arXiv.org).

K **S. P. Azad, B. Niazmand, P. Ellervee, J. Raik, G. Jervan and T. Hollstein,** "SoCDep^2: A framework for dependable task deployment on many-core systems under mixed-criticality constraints," 2016 11th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), Tallinn, 2016, pp. 1-6.

L **T. Hollstein, S. P. Azad, T. Kogge and B. Niazmand**, "Mixed-criticality NoC partitioning based on the NoCDepend dependability technique," 2015 10th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), Bremen, 2015, pp. 1-8.

M **T. Putkaradze, S. P. Azad, B. Niazmand, J. Raik and G. Jervan,** "Fault-resilient NoC router with transparent resource allocation," 2017 12th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), Madrid, 2017, pp. 1-8.

Awaiting publication

Awaiting publications that have been accepted but not indexed yet and the author has had contribution to them, are as follows:

N **B. Niazmand, S. P. Azad, T. Ghasempouri, J. Raik and G. Jervan,** "A Hierarchical Approach for Devising Area Efficient Concurrent Online Checkers", 2018, 2nd International Test Conference in Asia (ITC-Asia), Harbin, China.

O **T. Ghasempouri, S. P. Azad, B. Niazmand and J. Raik**, "An automatic approach to evaluate assertions' quality based on data-mining metrics", 2018, 2nd International Test Conference in Asia (ITC-Asia), Harbin, China.

P **S. Avramenko, S. P. Azad, S. Esposito, B. Niazmand, M. Violante, J. Raik and M. Jenihhin,** "QoSinNoC: Analysis of QoS-Aware NoC Architectures for Mixed-Criticality Applications", 2018, IEEE 21st International Symposium on Design and Diagnostics of Electronic Circuits (DDECS), Budapest, Hungary.

Author's Contribution to the Publications

Author's contribution to the publications included to the thesis

Contributions of the author to the papers based on which this dissertation is written, are as follows:

A This paper proposes a methodology for automated minimization of concurrent online checkers, guaranteeing shortest fault detection latency, for control part of Network-on-Chips (NoCs) under given fault detection quality and area constraints. In this work, the author has devised the initial set of checkers for the routing logic and arbitration module of a NoC router.

B In this paper, a framework of tools for formally evaluating the quality of concurrent online checkers and for optimizing the overhead area with given fault coverage constraints is proposed. The proposed methodology has been applied to the full control part of a NoC router, consisting of the control part of input buffer, routing logic and arbitration logic. The author devised a new set of checkers for the router's input buffer's control part. Moreover, the author participated in implementing the single parity checker for protecting the data-path of the router against single stuck-at faults. This work is an extension of **publication A**.

C The author has introduced the idea of logic-based distributed routing for 3D Network-on-Chips (NoCs) in order to address implementation of fault-tolerant deadlock free routing algorithms in 3D NoCs with partially faulty vertical links. Implementation of the mechanism in an open-source NoC simulator and performing the experiments of the paper have also been done by the author.

D The author, in collaboration with the first author of the paper, has devised the initial set of combinational checkers for the control part of an open-source NoC router (comprising of routing logic, arbitration module and control part of FIFO) using the systematic methodology explained in details in this dissertation. The author has performed the fault simulation experiments which is part of the framework that performed the evaluation of the checkers in terms of fault detection capability and minimization of checkers in terms of area. The minimized set of checkers have been integrated in the open-source router design.

E This paper proposes a ground-up approach from fault detection to fault management for NoC-based System-on-Chips (SoCs). This work use an open-source NoC router. The fault detection in the control part of the routers is performed using the concurrent online checkers, devised based on the proposed methodology in this dissertation. The author has devised and integrated the full set of checkers in the router design. Furthermore, the author has re-used the fault classification module in the router and provided fault classification for checker outputs. Finally, in order to compress fault information acquired from a large set of checkers (more than 1000 bits), the author has proposed and implemented a fault localization module, which compresses the checkers information to a final set of 20 bits, representing turn faults in the router.

Contributions of other related publications

F This paper proposes concurrent online checkers for structural faults in the NoC routing algorithms utilizing the Logic-Based Distributed Routing (LBDR) concept. Using fault injection experiments and an extended set of checkers for LBDR, the fault coverage is increased more than three-fold facilitating detection of the majority of structural faults within the routing logic.

G In this paper, the initial idea of the methodology for minimization of aa set of concurrent online checkers for the control part of NoC router based on area constraints and target fault coverage has been introduced. In this work, the routing logic and arbitration unit of the router have been considered as the control part for devising checkers.

H This paper proposes a framework for automated evaluation of concurrent online checkers. The novelty of the underlying approach lies in its completeness (i.e. ability of formally proving the presence or absence of true misses), minimal fault detection latency and accurate, fully automated evaluation of the fault detection characteristics of the checkers. The control part of a NoC router, consisting of the routing and arbitration logic has been used as an example for applying the proposed framework for devising checkers.

Contributions of other publications

I In this paper, all the uniform turn models for the 2D Mesh-based NoCs are enumerated and the deadlock free ones are extracted, which provide full connectivity in the network. An extended adaptivity metric is introduced to classify the turn models. The turn models are compared in terms of adaptivity, robustness and latency.

J In this paper, a holistic approach for fault-tolerant NoC-based many-core systems is described that incorporates a System Health Monitoring Unit (SHMU) which collects all the fault information from the system, classifies them and provides different solutions for different fault classes. A Mapper/Scheduler Unit (MSU) is used for online generation of different mapping and scheduling solutions based on the current fault configuration of the system. All the experiments in this work are performed in an open source tool, able to perform the mapping, scheduling and simulation of the system.

K In this paper, an open-source framework for task deployment of mixed-critical and non-critical applications under dependability constraints in Network-on-Chip (NoC) based systems has been introduced. The system level design space exploration is guided by a System Health Monitoring Unit which keeps a holistic view of system health status. The framework supports task clustering, mapping and scheduling of different applications, using different heuristics, on a NoC-based architecture which can have different topologies. This enables exploration of 2D and 3D topologies, any turn model based routing algorithm, fault monitoring mechanisms and different fault models (Link, Turn, Node). The author has contributed to the related work section of the paper, by finding different frameworks and tools that have addressed mapping and scheduling, mixed criticality support and fault-tolerance support in NoCs.

L In this paper, an approach for encapsulation of critical NoC communication resources is presented, which guarantees no interference of non-critical data packets with critical communication data on the network. The proposed mechanism can be used in order to achieve partitioning of the NoC into several criticality domains without additional overhead.

M In this paper, three novel fault-resilient Network-on-Chip (NoC) router architectures have been proposed. The proposed architectures exploit the regularity of the router and reallocate available existing and spare units to maintain functionality of certain turns. The resource reallocation is performed transparently from system's resource manager and is based on predefined priorities. A new metric for architecture reliability comparison based on reliability block diagrams is introduced. All proposed architectures have shown remarkable reliability improvement compared to original, Triple and Unit Duplication architectures, while at the same time, their area overhead is less than or equal to unit-duplication mechanisms.

1 Introduction

The shifting from computation-centric to computation- and communication-centric operations in digital systems, has motivated moving from single processing core systems to multi-processing core designs. This has led to integration of multiple components on the same chip. In such systems, the communication infrastructure can become a bottleneck, as the performance of the system also depends on the interconnection, providing the communication between the components [1]. Traditional shared-medium bus-based systems cannot catch up with the growing number of on-chip cores in terms of performance, thus, Network-on-Chip has emerged as a scalable solution for providing the interconnection infrastructure in Multi-Processor System-on-Chips (MPSoCs) [2]. In a System-on-Chip using NoC as its communication infrastructure, the network usually consists of routers, Processing Elements (PE), Network Interfaces (NIs) and communication links. Routers are in charge of transmitting data to the corresponding destination.

The miniaturization of semi-conductor technologies has jeopardized the reliability of integrated circuits and has made transistors more susceptible to different types of faults, including permanent, intermittent and transient. This also affects the reliability of NoCs, including the control part of NoC routers which is the focus of this dissertation. The control part of a NoC router plays an important role in successful transmission of data. A transient or permanent fault in the control part can lead to malfunction of the whole router, eventually leading to loss of data, mis-routing of packets or in worst case the break-down of part of/the entire network.

Even though early-life failures are handled by techniques such as manufacturing testing, it is impossible to ignore the adverse effect of run-time faults caused by phenomena such as aging and wear-out. The waning reliability threat against NoCs has been one of the focuses of research during the last years. Especially, capturing and detecting the faults online in the NoC components is crucial for transient faults, because, even if the permanent faults are detected by testing, transient faults (due to the nature of their random occurrence and being active for short duration of time) manifest themselves during system's life-time. Approaches based on Built-In Self-Test (BIST) introduced in the literature usually suffer from delayed fault detection as they require the system operation to be partially/fully paused while being in test mode. On the other hand, approaches based on Triple Modular Redundancy (TMR) or NMR based techniques would be expensive in terms of area overhead for providing fault-tolerance in NoC routers. Thus, in this dissertation, the concurrent online checking of faults in the control part of NoC routers via checkers [3] is chosen. Checkers allow monitoring the control part modules in parallel with their operation, without pausing the system functionality. They raise up a flag denoting the captured fault. However, it is important that the goal in this thesis is to have checkers with instantaneous fault detection latency, able to detect faults within maximum one clock cycle of their occurrence. Because, otherwise, the fault could get propagated to the rest of the system and causes total system failure [4]. One advantage of using checkers instead of DMR and TMR-based approaches is that they provide fault localization possibility.

This thesis proposes a set of techniques to improve the dependability of NoCs, *i.e.* online detection of faults in the control part while meeting the area constraints, abstracting the fault detection information, and implementing a generic and re-confugrable fault-tolerant routing mechanism to circumvent faults on inter-router links.

1.1 Motivation

The trend in shrinking size of transistors, extreme down-scaling of the nanometer technologies beyond the sub-micron domain and shrinking voltage levels, makes devices more susceptible to faults, both to permanent and especially transient ones. This also applies to the on-chip components, including Network-on-Chip (NoC), introduced as an alternative infrastructure to overcome the performance and scalability limitations of traditional shared-bus architectures [5], [6]. All these circuits are prone to different fault sources, *e.g.* Electro-Migration (EM), wear-out, Alpha particles and cosmic radiation [7], [8].

Although a lot of efforts have been made in order to capture faults before the final product is released (such as manufacturing testing), faults can still manifest themselves during the life-time of the circuit. More specifically, Integrated Circuits (ICs) are susceptible to wear-out and aging occurring during their life-time and if not handled properly, they can corrupt system's functionality and its normal operation. Online detection is especially critical for transient faults. This is because, even if the permanent faults are detected via testing or other techniques [9], transient faults (due to nature of their random occurrence and being active for short duration of time) can occur during system run-time and affect system's operation. This motivates the need for instantaneous detection of such faults [4]. In addition to the detection, fault localization is of utmost importance, which would eventually facilitate re-configurating the NoC.

1.2 Problem Formulation

One of the main targets of this dissertation is online detection of faults in control part of NoCs. This is due to the fact that detection of faults with lowest possible latency is important in order to avoid propagation of the fault to the whole system. Therefore, there is a need for a mechanism that can react as rapid as possible to the occurrence of transient and permanent faults in the system. However, the area overhead of the augmented fault detection circuitry should be taken into account, since the higher the overhead, the higher the chance of faults occurring in the fault detection logic itself.

In addition to the importance of near-instantaneous online detection of faults, the topic of fault localization is also significant. It is important to locate the faulty component, so that the system could be re-configured with degraded performance by bypassing the defective component, while leaving the healthy components intact. However, as the fault information overhead grows, care must be taken that the acquired data would be transmitted to higher layers (such as application layer) in form of compact and meaningful information, which can further be used for system re-configuration. For instance, a global fault manager can use the compressed fault information in the process of computing a new routing algorithm to address the faulty system.

Finally, the implementation of fault-tolerant routing algorithms in NoCs is an important issue to be addressed. The mechanism used for implementing a routing algorithm must be generic, re-configurable and must guarantee deadlock and live-lock freeness, in order not to affect system performance. It must also must not depend on the location and number of faulty links in the network. Moreover, the scalability of the mechanism is of utmost concern due to the on-chip limited area budget, as it should not grow with network size. And finally, the mechanism must guarantee connectivity in the network as long as faults do not disconnect it.

1.3 Contributions of the Thesis

This dissertation focuses on the following topics: (1) a methodology for devising concurrent online checkers for performing *online fault detection* in control part of NoCs, providing a trade-off between fault coverage and area overhead of the checkers, (2) *fault localization* via combining the checker outputs in order to find the location of the fault in the circuit and compress fault information in order to model more abstract information regarding *turn faults*, and (3) *implementing fault-tolerant routing algorithms* in 2D and 3D Network-on-Chips using a scalable logic-based mechanism.

The contributions of this dissertation are three-fold and summarized as follows:

1. In order to address the first problem in this dissertation, a methodology for devising concurrent online checkers for performing *online fault detection* in control part of NoCs is proposed. The proposed methodology provides a trade-off between fault coverage and area overhead of the checkers. It allows devising checkers at two levels, *i.e.* functional and structural, independent of the architecture of the NoC router. The proposed methodology guarantees (1) single-cycle fault detection latency for all the checkers, (2) formal proof of absence of cases that faults occur in the circuit, but not captured by the checkers (called True Misses) using fault simulation, and (3) automated minimization of the checkers in terms of area using greedy heuristic, while meeting the requirements of the target fault coverage. It is worth noting that the concurrent online checkers operate in parallel with system's operation. In order to be able to measure the fault detection capability of the checkers, new metrics have been proposed which enable clear definition of fault detection quality of the checkers. This contribution has led to the **publications A, B, and D**, mentioned in the list of publications included to the thesis.
2. A fault localization module is developed which takes into account the fault information acquired from the concurrent online checkers for the control part of the NoC. The fault localization circuitry is fully combinational, and takes advantage of the single-cycle fault detection latency of the checkers, by grouping them and providing compact, meaningful information regarding faults for higher levels of abstraction. In case of the control part NoC router, in addition to router-level and component-level, a third level of fault localization has been proposed which models *turn faults*. This would compress the fault information and tackle the issue of generation of excessive amount of data by the checkers. At the same time, it reduces rendering the whole router as faulty, and making it possible to be re-used with degraded performance with the intact healthy turns. A System Health Monitoring Unit that keeps a holistic view of the system's health will make use of such information provided by the fault localization module to re-configure the underlying routing algorithm, if needed. This contribution has led to the **publication E**, mentioned earlier in the list of publications included to the thesis.
3. A logic-based mechanism is developed for implementation of turn model-based routing algorithms in partially vertically connected 3D NoCs, named LBDR3D. The mechanism does not use routing tables at the routers and only relies on a fixed set of configuration bits which specify the topology, routing algorithm and the existence of at least one node with vertical link in each layer. The proposed

approach exceeds the state-of-the-art, by not storing the location address of such nodes with vertical links at each router, thus, making it scalable. It also does not incur any additional overhead to the packets being transmitted from one layer to another. Moreover, it does not depend on the location and number of faulty vertical links. Furthermore, LBDR3D guarantees live-lock freeness and live-lock freeness, and also guarantees connectivity as long as faults do not disconnect the network. The third contribution has led to the **publication C**, mentioned earlier in the list of publications included to the thesis.

1.4 Thesis Organization

This thesis is organized in 5 Chapters and 7 Appendices.

In Chapter 1, an introduction to the thesis is provided, including the motivation, problem formulation and the main contributions.

Chapter 2 covers the background of the topics discussed in this dissertation. It consists of two sections. In the first section, a literature review, regarding the state-of-the-art approaches related to the topics discussed in this thesis, *i.e.* approaches for online detection of faults in control part of NoCs, approaches for fault localization in NoCs, and approaches for implementing fault-tolerant routing algorithms for 3D NoCs. In the second section, background information about the subjects that are used as baseline in the next chapters, including the explanation of the open-source Bonfire Network-on-Chip (NoC) project and Logic-Based Distributed Routing (LBDR), based on which the contributions of this thesis are introduced.

Chapter 3 is dedicated to the explanation of the first contribution of this thesis, *i.e.* the proposed methodology for devising *concurrent online checkers* for control part of NoCs and automated evaluation of fault detection quality of checkers and minimization in terms of area while meeting the target fault coverage.

Chapter 4 discusses the next contribution of the thesis, fault localization in NoCs, taking into account the checker outputs information and providing meaningful and abstract *turn faults*, by compressing the data acquired from checkers output. This would facilitate the process of re-configuring the routing algorithm of the network by the system fault manager in case of a fault occurrence.

Chapter 5 explains the third contribution of this dissertation, a scalable and re-configurable mechanism for *implementing fault-tolerant routing* algorithms in 3D NoCs with faulty vertical links.

Finally, the last chapter concludes the dissertation, remarking the theoretical novelties of this work and summarizing the contributions.

This dissertation is accompanied with seven Appendices, from which Appendices A and B serve as supplementary information regarding checkers in Chapter 3. Appendix A includes the complete list of checkers for one of the examples used for applying the proposed methodology for devising checkers, which is the control part of Bonfire handshaking router. Appendix B covers the application of the same methodology for devising checkers to the control part of Bonfire credit-based router as one of the other examples. Appendices C to G present the research papers which form the basis of this dissertation.

2 BACKGROUND

2.1 Introduction

This chapter starts with a literature review regarding approaches related to the contributions of this thesis, which are threefold: (1) a study of online fault detection approaches for NoCs, mostly focusing on the control part, (2) a state-of-the-art review regarding approaches for fault localization in NoCs with the focus on control part, and (3) a literature review of fault-tolerant routing mechanisms addressing 3D NoCs with faulty vertical links.

Afterwards, the chapter continues with the pre-requisite background information which would be referred to in this thesis continuously in the following chapters, including the fault model used in this thesis, different router architectures used as examples (three different architectures) for applying the proposed methodology for devising and minimizing checkers and also fault localization to abstract fault information. Finally, a background regarding the logic-based distributed routing and its variations according to the literature is provided, used as the baseline mechanism for implementation of fault-tolerant routing algorithms in partially vertically connected 3D NoCs.

2.2 Literature Review

In the following three sub-sections, the state-of-the-art regarding the topics that this dissertation focuses on, are reviewed.

2.2.1 Online Fault Detection Approaches for NoCs

The online fault detection approaches reviewed in this sub-section are *reactive*, meaning that they detect the fault after its occurrence and react to it. In other words, they tackle run-time failures by detecting hardware failures shortly after they manifest. The other category of approaches would be *pro-active*, which predict the occurrence of faults before their occurrence and try to mitigate the effects beforehand. However, pro-active approaches are not in the scope of this dissertation. It is noteworthy that the focus of the reviewed approaches in this sub-section is on control part of NoCs.

Online detection of errors in logic is a thoroughly studied research area. One of the well-known techniques is *hardware redundancy*, which has also been studied in the field of NoCs. Approaches such as traditional Triple-Modular Redundancy (TMR) and Duplication With Comparison (DWC) approaches [10] exist, however, they are costly in terms of multiplying the area and correspondingly the power consumption. Moreover, despite providing fault detection capability, such approaches lack providing information facilitating fine-grain fault localization. An alternative to minimize the area overhead of such approaches is the selective TMR that identifies Single Event Upset (SEU) sensitive sub-circuits that are to be protected [11], but it still suffers from the inability to localize faults.

On the other hand, some of the approaches address detection of faults via *information redundancy*, including a variety of solutions based on coding techniques, such as Berger [12] or Bose-Lin [13] codes. In many works the coding techniques are combined with synthesis [14], [15]. However, these approaches suffer from significant area overhead, and they require alteration of the original circuit in order to generate the codes.

Concurrent on-line built-in self-test techniques such as Built-In Concurrent Self-Test (BICST) [16] and Reduced Observation Width Replication (ROWR) [17] provide high fault coverage at low area overhead, but only consider a limited subset of pre-computed test vectors. Hence, these approaches are likely to miss faults occurring in a normal circuit operation.

Several alternatives based on *checkers* that do not require modification of the circuit under test have been developed. Creating checkers automatically based on logic implications derived from the circuit structure [18], [19] is feasible but suffers from low fault coverage and high area overhead, often exceeding the duplication-based solutions.

On the other hand, deriving checkers from functional assertions, or reusing verification assertions, is similarly known to yield low coverage of structural faults as it is difficult to correlate functional coverage to structural one [20]. In [21], Grecu et al. have introduced a method for online fault detection and location in NoC communication fabrics. The proposed method is able to distinguish between faults in the communication links and the ones in the NoC switches. This work is based on the utilization of code-disjoint routing elements, combined with parity check encoding for the inter-switch links. However, the method targets faults in the data-path only.

A group of works in the literature have focused on monitoring control part components of NoC switches, such as [22]–[28]. Authors of [29] have introduced SafeNoC, an end-to-end error detection and recovery solution, for ensuring the functional correctness of Chip Multi-Processor (CMP) interconnects. In this solution, a lightweight checker network is added to the existing interconnect, that guarantees to deliver messages correctly. Therefore, for each data message, a look-ahead signature is transmitted over the checker network, which is used for detecting errors in the corresponding data message. The solution does not provide checking for faults within the routers. Moreover, in case of the increase in the number of faults in the system, the reconstruction and recovery process can take up to 39M execution cycles. It should be noted that the focus of this dissertation is on fault detection and localization approaches in NoCs, however, fault recovery approaches are not in the scope of this thesis.

Several works have proposed utilization of *concurrent online checkers*[1] for checking faults in the control part of on-chip routers. In [26], the *Inherent Information Redundancy (IIR)* in the control path of NoC routers is utilized to manage transient errors. The goal is to prevent packet loss and misrouting by detecting such faults in the routing computation and in the arbitration unit of a NoC router. However, their approach is only limited to XY routing.

Yu et al. [27] have proposed a set of checkers for the NoC routing algorithmic blocks implemented using LBDRhr for topologies with high-radix. To this end, the Inherent Information Redundancy (IIR) [26] in LBDRhr logic is exploited in order to manage transient errors in the routers. Despite the advantages their approach provides compared to routing tables in terms of scalability, the proposed checkers for LBDRhr logic cannot reach 100% fault coverage. Furthermore, the work in [27] only focuses on the routing logic of a NoC router and not considering the full control part.

In [30], the set of checkers introduced in [27] are extended for the baseline LBDR logic in order to increase the fault coverage (up to 64.9%). A final set of five checkers are

[1] The concept of checkers used in this dissertation will be explained in more detail in the Chapter 3.

proposed, which cover the majority of single stuck-at faults occurring in the LBDR circuitry. Fault injection experiments have shown that the proposed method in [30] allows increasing the fault coverage 3 times (compared to [27]), of course at the price of 26.8% checker area overhead. However, still 100% fault coverage is not reached and the area overhead minimization aspect of the checkers is also not addressed neither in [27] nor in [30].

Authors of [31] have presented a method for online error detection and diagnosis of NoC switches. The proposed method deals with routing faults that cause NoC packets to be forwarded to output ports that are not intended to. Regarding modelling routing faults in switches, a high-level fault model has been introduced in this work. However, this work targets functional level fault coverage only and does not guarantee a good coverage for structural faults.

Parikh et al. have proposed ForEVeR [25], a solution that complements the use of formal methods and runtime verification to ensure functional correctness in NoCs. In order to deliver correctness guarantees for the complete network, a network-level detection and recovery solution is proposed in [25] that monitors the traffic in the NoC and protects it against functional bugs that were not detected during design time. To this end, ForEVeR augments the baseline NoC with a lightweight checker network that alerts destination nodes of incoming packets ahead of time and is used for the recovery process. The use of an end-to-end, epoch-based scheme, such as ForEVeR, results in significantly delayed fault detection. Additionally, Only 30% of the faults are detected during the first clock cycle by their approach.

Authors of [23] (*NoCAlert*) have proposed checkers synthesized from a set of 32 assertions. The checkers detect most of the injected faults with minimum detection latency. The faults that are not covered correspond to non-catastrophic failures. However, it is not clarified with which type of checkers (9 in total reported in [23]) 100% fault coverage is reached. Furthermore, the minimization aspect of the area overhead of the checkers is not addressed in [23]. In addition, in high-level evaluation process, NoCAlert checkers only consider faults occurring on the primary inputs and outputs of the control logic and the modules themselves are viewed as black boxes, thus, not considering fault locations inside the control part modules of the NoC router.

In [32], Secure Model Checkers (SMCs) have been proposed. Similar to NoCAlert, they target the control part of NoC routers, but also focusing on the security aspects, for instance, protection against Hardware Trojan (HT) attacks. However, similar to NoCAlert, the methodology in [32] has not addressed the minimization aspects of the checkers in terms of area overhead. It is worth noting that the focus of this dissertation is on online fault detection in control part of NoCs and security aspects of NoCs is out of the scope of this thesis.

In [33], an online checking mechanism is proposed for the switch allocator of a NoC router that is able to detect every possible single transient or permanent fault in the arbiter and handle it appropriately. The proposed checkers for the switch allocator of the router in [33] have self-checking property. Despite the advantages, they have neglected detection of faults in the full control part of the NoC router, *i.e.* the control part of input buffers and the routing logic.

Park et al. [22] have examined the impact of transient faults on the reliability of on-chip interconnects and have developed an approach to recover from them. For the inter-router link faults, they use Hop-By-Hop (HBH) retransmission method. However, the retransmission buffer can add latency to the system in case of a fault occurrence. Moreover, it is not mentioned whether the retransmission buffer itself is protected against SEUs or not. Regarding the control part of router, an Allocation Comparator (AC) unit is proposed, which provides full error protection to the Virtual Channels (VCs) and Switch Allocation (SA) units at minimal cost, without affecting the router's critical path. The work in [22] assumes successful speculative allocation for the allocator. However, mis-speculation can incur overhead.

In [28], illegal turns in the routers are detected, however, each router depends on the information from its neighbour routers for online fault detection and judgment.

The following works have addressed detection of faults in NoC switches via Built-In Self-Test (BIST)-based approaches. In [34], fault detection is performed via a BIST mechanism executed during system boot-up. In [35], fault detection is performed via an automatic go/no-go BIST operation at the start-up of the network. However, the approach is only limited to 2D Mesh NoCs, and the fault coverage of the switch controller is low. Petersen et al. have extended the idea of [35] in [36], in which, fault coverage close to 100% is reached, with few thousand clock cycles fault detection latency. Despite the advantages, their BIST architecture still incurs significant area and fault detection latency overhead. Authors of [37] have taken advantage of the regularity of intra-switch ports and also the regularity of inter-switch communication infrastructure of a NoC in order to decrease test application time and decrease test data volume of NoC testing. One of the main drawbacks of BIST-based architectures is that system operation needs to be partially or fully paused, while the module under test is being examined, which can, in turn, degrade performance.

There have also been works in the literature that have focused both on monitoring the data-path and control part components of NoC router. In Cardio architecture [38], fault detection is handled by hardware, whereas software is in charge of conducting reconfiguration, leading to reduction of area overhead (as stated in [38]). To this end, a distributed resource manager is utilized. Cardio targets run-time permanent faults in (a) processor cores (by implementing counters and acknowledgement buffers in Network Interface (NI)), (b) interconnect routers (via configurable routing table logic) and (c) links in the intra-chip communication subsystem (via link monitors). In [39], both the data-path and control part faults in Network-on-Chips are addressed via a multi-layer diagnosis approach. However, the structural diagnosis approach imposes significant fault localization latency. In [40], a fault-tolerant routing method is proposed which works based on partial fault model. The approach addresses permanent faults in input buffer, control unit, crossbar and output buffer of the NoC router and in the processor core attached to the router. However, no details regarding handling and detection of permanent faults are provided.

Even though there have been many approaches covering faults in the control part of NoC online - as summarized in the above-mentioned paragraphs - to the best of this dissertation's author's knowledge, none of the previous works have addressed a methodology for devising checkers for the part of a circuit, which would guarantee single cycle fault detection latency for Single Event Upsets (SEUs) and evaluation of checkers under all possible set of valid input stimuli for all possible fault locations in the circuit,

addressing the area minimization of the checkers, while guaranteeing the target fault coverage. It is also noteworthy that since the focus of this dissertation is not on protection of data-path components of NoCs, it is already assumed that the architecture has a fault detection/correction mechanism integrated for handling faults in the data-path (inter-router links and intra-router data-path components), *e.g.* one of the approaches mentioned in [41], [42].

2.2.2 Online Fault Localization Approaches for NoCs

Online fault localization is one of the necessities in addition to fault detection for performing fault diagnosis in a system. A variety of approaches have addressed diagnosis of faults in NoCs at different levels. Some of the works in the literature have addressed detection and localization of faults in NoCs using broadcasting of test vectors, at run-time [7], [43]–[48]. In [43], *Vicis* architecture is introduced, in which a BIST procedure tests the individual sub-blocks of the on-chip switch, covering both control part and data-path. One of the drawbacks is the area overhead imposed by the wrapper cells used to isolate the faulty sub-block from the system. Also, Vicis lacks the capability to find the exact root of the faulty behaviour in the switch and does not have reasoning regarding defective switch functionalities. Moreover, as mentioned before, one of the disadvantages of run-time BIST in general is that the system operation needs to be either partially or fully paused, while the module under test is being examined, which, in turn, does not allow detecting soft errors at run-time and degrades performance.

Approaches based on online monitoring of faults such as [32], [15], [17], [21], [33] have addressed faults in the control part components of NoC switches. It is worth noting that since the focus of this dissertation is on fault detection and localization in control part of NoCs, the works addressing only data-path components are not in the scope of the reviewed literature in this chapter.

Alaghi et al. [31] have presented a method based on high-level fault model for online error detection and diagnosis of routing faults in NoC switches. However, this work targets only functional level fault coverage and does not guarantee a high coverage for structural faults. In addition, for some of the fault models explained in the paper, fault localization cannot be achieved. In [24], the NoCAlert mechanism for online detection of faults in the control part of NoCs has been augmented with fault localization capability. However, their approach lacks the area minimization aspect of the checkers and cannot guarantee 100% fault coverage via checkers within a single cycle. Furthermore, the proposed fault localization module in [24] does not address modelling *turn faults* (taking fault information contributing to both input and output port related control part components of the router). One of the approaches that addresses turn faults is the one proposed in [28], in which illegal turns in the NoC routers are detected. However, each router depends on the information from its neighbour routers for online fault detection and judgment.

On the other hand, authors of [49] have introduced an online-structural approach. They target the diagnosis of permanent faults on NoC links and the control-logic faults. However, their approach might consider an entire switch as faulty when a fault occurs in part of it, thus, suffering from low fault localization accuracy. Moreover, the work in [49] does not guarantee low latency error detection. In [40], a fault-tolerant routing method is proposed which works based on partial fault model. Their approach addresses permanent faults in input buffer, control unit, crossbar and output buffer of the NoC

router and in the processor core attached to the router. However, to simplify the fault model, an occurrence of fault in crossbar, control unit and the processor core are all modelled as a node failure. This, reduces the granularity of fault localization. Also, no details are provided in [40] regarding how the permanent faults are detected. Authors of [50] have presented a mechanism for detection and localization of faults in dynamic NoCs, with varying number of PEs during run-time. They target both transient and permanent faults in data packets and errors related to adaptive routing algorithms. However, their approach can localize faults in NoC routers at the level of input port, output port and/or data bus. Moreover, in order to provide routing error detection, an additional field is added to the transmitted packets, and also the routers communicate diagonally with their neighbour routers for transmission of state information, in total incurring a 63% area overhead in a 6x6 2D NoC.

The authors of [49] have addressed a diagnosis approach involving multiple layers, in which the software part is responsible for locating faulty links and crossbar connections in hardware. However, their approach lacks the cross-layer interaction and the proposed diagnosis techniques for each layer are separated from each other. In [39], a multi-layer diagnosis architecture is proposed for NoCs with cross-layer interaction. They have demonstrated the combination of layer-specific diagnosis techniques could be beneficial compared to using only individual layer-specific approaches. Both a top-down and a bottom-up flow for cross-layer information flow is presented. The former is used to narrow down the position of a fault, while the latter provides diagnostic feedback from lower to higher layers, also, solving the cases of false positives. However, despite obtaining 100% fault coverage, the fault localization latency suffers from significant overhead. But, it should be noted that works such [39] target best fault localization resolution via offline diagnostic reasoning. Therefore, their goal is different from this dissertation, which is online concurrent detection of faults with minimal latency.

Aghaei et al. [41] have performed a survey on different link testing mechanisms addressing stuck-at, bridge, delay and crosstalk fault detection and diagnosis in on-chip inter-router links and links between Network Interface (NI) and router. They reached the conclusion that none of the approaches up to that point had addressed a single platform for fault detection, diagnosis and fault tolerance under the one single framework.

To the best of the author's knowledge, none of the above-mentioned works have addressed a cross-layer fault resilient NoC router architecture, utilizing online fault localization in addition to fault detection for the control part of NoC routers, while targeting minimal latency and matching acquired fault information with abstraction level of system healthy information (*e.g.* health status of the turns of a router).

2.2.3 Fault-Tolerant Routing Mechanisms for 3D NoCs

The aggressive transistor scaling also affects the reliability of inter-router links in NoCs. Especially in the domain of Mesh-based 3D NoCs, in which the vertical links are present in addition to the horizontal links, faults in the vertical links can cause performance bottlenecks. Moreover, if the vertical links are implemented using Through-Silicon Via (TSV), it would not be area-efficient to have a full 3D Mesh NoC, as TSVs impose larger area overheads compared to the horizontal links [51]. Due to these reasons, *partially vertically connected* 3D NoCs can be formed. Similar to their 2D counterparts, fault-tolerant routing in such 3D NoCs and how they are implemented can also bring some

challenges. The following works review the state-of-the-art regarding fault-tolerant routing mechanisms for handling partially vertically connected 3D NoCs.

In [52], 4NP-First is introduced, which is a low overhead fault-tolerant routing algorithm for 3D NoCs. The algorithm is able to achieve high arrival rates of packets at destination. It utilizes a hybrid turn model (based on an extension of the Negative-First turn model to the 3D domain): 4N (Negative) First and 4P (Positive) First. Using a set of forbidden turns in each layer, 4NP-First guarantees deadlock freeness. However, the approach suffers from information overhead, as it replicates each packet if number of faulty links in the network exceeds a specific threshold, thus, sending one replica of the packet via one virtual channel using 4N-First and the other via another virtual channel using the 4P-First routing algorithm.

In [53], a low-overhead fault-tolerant deflection routing algorithm is proposed for 3D Mesh-based NoCs. The limitation of this work is scalability due to using routing tables per layer. Authors of [54] have introduced *AFRA*, a deadlock-free and deterministic routing algorithm (based on an extension of ZXY routing algorithm) for 3D NoCs. Normally, the algorithm performs as ZXY. In case of a fault on a vertical link on the path to the destination of a node, the algorithm tries to find an escape node with healthy vertical link along the X direction, thus, changing the algorithm to XZXY. One of the drawbacks of this work is the assumption of faults occurring only in one direction on the vertical links. The other drawback is that if an escape node does not exist, the algorithm cannot handle the faulty network, therefore, AFRA has limitations regarding the location of faulty vertical links.

Ebrahimi et al. have proposed *HamFA* [55], which takes advantage of Hamiltonian paths in order to tolerate faults in 2D and 3D NoCs without the need for any Virtual Channels (VCs). Despite the advantages compared to 4NP-First [52] and [53], HamFA is not able to address faults on vertical links at the end of the Hamiltonian paths and also some of the horizontal links in each layer, as stated by the authors in [55]. Jiang et al. have presented an efficient fully adaptive fault- tolerant routing algorithm for 3D NoCs [56]. The algorithm consists of two phases: inter-layer and intra-layer routing. Two assumptions that limit this work are as follows: Processing Elements (PEs) will never get faulty and faults on links are considered as bidirectional. Also, the deadlock recovery mechanism used in this work can impose additional performance overhead.

Authors of [57] have proposed a high-performance reliable and deadlock-free routing scheme (*HARS*), which follows a mid-node searching method in 3D NoCs without requiring any Virtual Channels (VCs). However, reliability results are only provided when up to 10% of the network vertical links are faulty. In [51], *Elevator-First*, a distributed routing algorithm has been proposed for partially vertically connected 3D Network-on-Chips. The algorithm is able to tolerate faults on vertical links, regardless of the location and the number of faults. In order to guarantee deadlock freeness, the method depends on using two virtual channels along X and Y dimensions. Despite the advantages, the algorithm relies on an additional overhead in header flits, when steering packets to nodes with vertical links (called as elevator nodes). Also, each router stores the location of at least one up and one down elevator node in its layer for fault-tolerance purposes which can impose additional memory overhead and scalability issues as the network scales up. In [58], TARAS, a topology-agnostic routing algorithm for 3D NoCs is proposed. However, it depends on the Segment-based Routing (SR) and therefore it would only cover a set of routing algorithms that address fault tolerance in 3D NoCs. In the proposed mechanism,

it is possible set the routing bits of LBDR3D, so that it would be programmed to the SR routing in each layer of the 3D NoC, thus making it a more generic approach.

In [59], [60], East-Then-West (ETW), an adaptive routing algorithm for supporting partially vertically connected 3D NoCs is introduced, able to tolerate faults on vertical links. It is claimed to be lightweight and only relies on using one Virtual Channel (VC) along the Y dimension. Nevertheless, ETW is not fully independent on the location of faulty links and it only works as long as there exists at least one vertical link at the east-most column of each layer. Ying et al. have introduced North-East To Z (NETZ) [61] routing algorithm based on Dynamic Quadrant Partitioning (DQP) for partially vertically connected 3D NoCs, able to improve performance in comparison to deterministic routing algorithms such as ZXY. The algorithm is implemented by disabling a set of turns in the 3D domain, thus, removing the need for routing tables and also guaranteeing deadlock freeness without using Virtual Channels (VCs). However, similar to ETW [59], [60], NETZ depends on the location of faulty vertical links. It requires the existence of a pillar at the North-East corner position on all layers to guarantee the routing algorithm delivers packets successfully to all destinations. Authors of [62] have introduced Advertiser Elevator algorithm to address partially vertically connected 3D NoCs, however, the approach in [62] depends on the existence of at least four healthy vertical links in the network (corner links).

In [63], a logic-based mechanism is proposed for implementing fault-tolerant routing algorithms in 3D NoCs, which is based on an extension to LBDR, however, the proposed technique relies on high number of configuration bits per router for the routing logic.

To the best of author's knowledge, based on the previous works studied, there is still an open research direction for proposing a re-configurable and scalable mechanism that would make it possible to implement deadlock- and livelock-free routing algorithms in 3D NoCs with faulty vertical links, while not sacrificing performance significantly, not imposing any information overhead and not relying on the location and number of faulty vertical links.

2.3 NoC Router Architectures Used in This Dissertation

This section is dedicated to explanation of the three NoC router architectures used in Chapters **3** and **4** of this dissertation, which are used as examples of applying the proposed methodology for devising checkers from the control part and one of them is also used for integrating the fault localization module to model turn faults. The router architectures explained in this chapter have the following features in common.

The NoC routers consist of a control part and a data-path. The data-path is composed of input buffers (implemented as circular First-In-First-Out (FIFO)), one per each input port, and a crossbar switch per each output port. The input buffers have one-hot encoded pointers for reading from and writing to them. Each input buffer has 4 slots for storing maximum 3 flits. This is due to the fact that one slot is used to distinguish the empty case of the input buffer from the case when it is full. None of the three router architectures in this dissertation use Virtual Channels (VCs) at input ports. However, the proposed methodology for devising checkers can also be applied to the control part of a router with VCs.

The flow of data through the data-path is managed and controlled by the control part of the NoC router. The control part of the router architectures explained in this chapter,

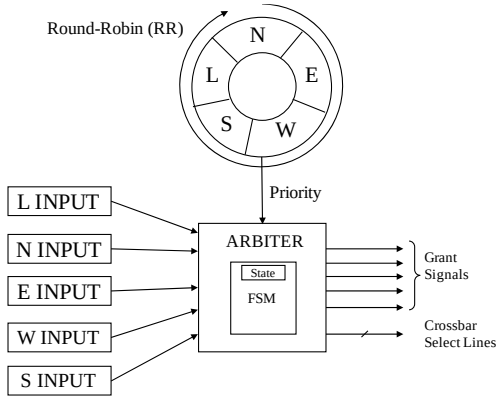


Figure 2.1 A Round-Robin (RR) arbiter for a 5 port of a 2D NoC router.

consist of a routing computation unit per each input port and an arbitration unit (arbiter) for each output port, which prioritizes the requests from different input ports to the same output port. Each router has 5 input/output ports, four ports connected to four cardinal directions (North – N, East – E, South – S, West – W) and one Local (L) port connected to the local Processing Element (PE). All three NoC router architectures utilize wormhole switching. Therefore, packets are sent in form of flits, consisting of header flit, body flit(s) and tail flit.

As faults in the control part can cause severe issues in the network (such as deadlock, live-lock, misrouting of packets, loss/dropping of packets) [64], protection of the control part against transient Single Event Upsets (SEUs) and permanent faults is of utmost importance. Regarding the data-path, there has already been many approaches proposed in the literature for protecting the data and links against faults and it is assumed in this thesis that the data-path is already protected by an Error Detection/Correction technique [41].

In the control part of all three router architectures, for the routing computation unit, Logic-Based Distributed Routing (LBDR) [65] mechanism is used, which is a scalable solution compared to routing tables. The mechanism describes the topology and the routing function in form of fixed sets of *connectivity* and *routing* bits, therefore, the logic can be easily re-configured. Routing decision is distributed and only requires local and destination addresses for forwarding flits. The routing computation is only performed on the header flit of a packet. Moreover, it must be noted that in all three router architectures, U-turns (an input port sending data to itself in output direction) are not allowed in order to avoid deadlock.

For the arbitration unit (shortly called the *arbiter* hereafter) Round-Robin (RR) policy has been chosen in all three architectures. Round-Robin arbitration (as shown in **Figure 2**.1) prioritizes multiple requests from the routing logic of different input ports to avoid contention. Prioritization is performed in a circular manner, starting from the N, E, W, S and then L and back to N. Arbiter grants the access to the requesting input port winning the eventual contention, allowing data to go from the input FIFO of the requesting input port to the granted corresponding output port, through its crossbar switch. The RR arbitration mechanism is implemented in form of a Finite State Machine (FSM). In all three router architectures, one-hot encoding has been considered for the state variables of Arbiter's FSM. Moreover, one-hot encoding is extended to grant signals

and select lines for the crossbar switch for similar reason. One of the main reasons such decision is to increase the fault detection quality of single-stuck at faults, which of course comes at the price of additional area compared to binary-encoded state variables.

2.3.1 NoC Router Architecture 1

As the first example, the proposed methodology for devising checkers in this dissertation has been applied to the control part of a generic NoC router, written in Verilog RTL. Hereafter, for future reference throughout the dissertation, this NoC router is named as **Architecture 1**, which is a 2D-Mesh based NoC router. **Figure 2**.2 demonstrates the high-level overview of NoC router Architecture 1, illustrating both the control part and the data-path. In addition to the input FIFOs and the crossbar switch, NoC router **Architecture 1** uses an output buffer per each output port, which can store one flit.

In NoC router **Architecture 1**, the routing logic (LBDR) is configured to the deterministic XY turn model. Taking into account the fact that U-turns and also turns from the Y dimension to X dimension (*i.e.* North and South, to East and West) are not allowed in XY routing, this can lead to simplification of the logic of LBDR. As it will be explained later in the experiments in Chapter 3, for router **Architecture 1**, the focus is on

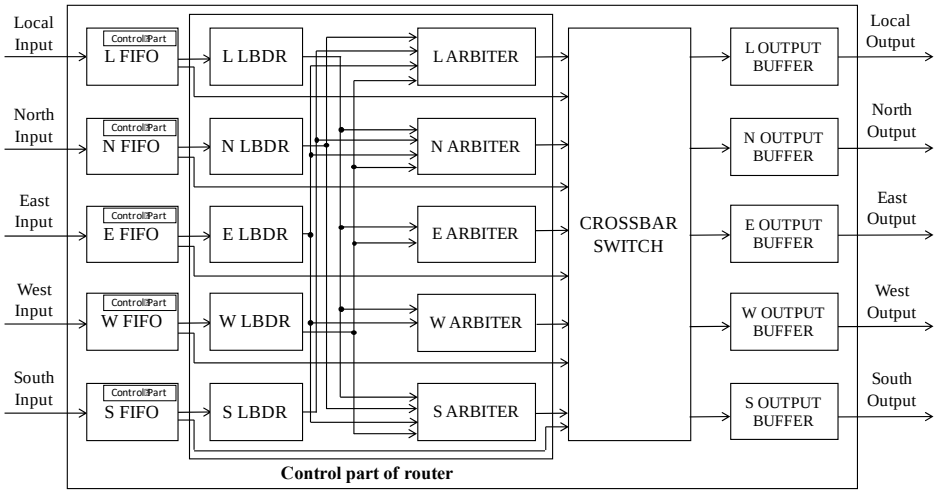


Figure 2.2 High-level overview of NoC router **Architecture 1**

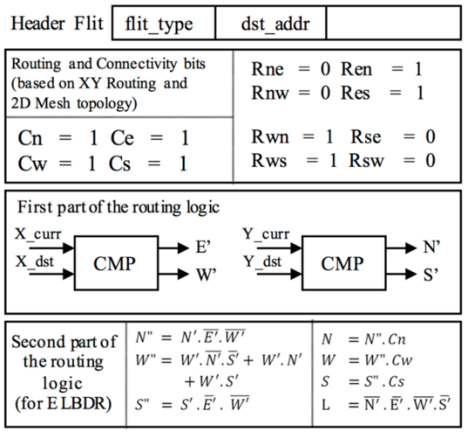


Figure 2.3 Logic-based Distributed Routing (LBDR) logic for the East input port

the control part of router addressing the East input port and South output port and further in the last experiment, the control part of FIFO is also considered.

The simplified logic of LBDR based on XY is shown in **Figure 2**.3 [3] for the East input port of the router. According to XY routing, packets coming from East input, are only allowed to send requests to North, West, South and Local directions.

2.3.2 Project Bonfire Router Architectures

The Bonfire project [66], [67] proposes a fault-tolerance framework for implementing dependability mechanisms in a NoC-based System-on-Chips (SoCs). The targeted NoC in Bonfire project is a 2D mesh topology where each tile of the network consists of a wormhole switching router equipped with fault tolerance mechanisms and a Processing Element (PE) connected to it via a Network Interface (NI). The project consists of two types of NoC routers with two different flow control mechanisms, *i.e.* with handshaking and credit-based flow control. Details of the components of the framework are available online[2], however, a brief explanation is also provided in this chapter for future references in this dissertation. One of the shortcomings of the baseline router architectures of Bonfire was the lack of fault detection mechanism. The author has contributed to devising checkers for the control part of the Bonfire routers and also developing the fault localization and abstract module.

Similar to NoC router **Architecture 1**, both NoC router designs of Bonfire consist of a data-path and control part. However, their difference lie within the components used for the data-path and the flow control mechanism used in the control logic. The data-path comprises the inter-router links, the input buffers (implemented as circular FIFO) and crossbar switch (no output buffer is used in Bonfire routers). The control part is composed of the control part of input buffer (FIFO), routing logic (LBDR [65]), and arbitration logic. As it will be explained later, the data-path and control part components in both router designs have similarities, however, the way the flow control mechanism is

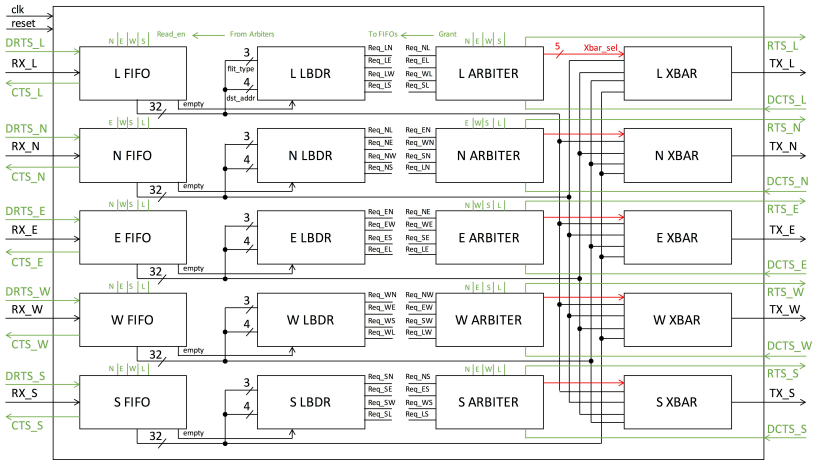


Figure 2.4 High-level overview of Bonfire NoC router with handshaking flow control

[2] Project Bonfire is developed in department of Computer Systems Engineering at Tallinn University of Technology, and maintained as an open-source project at: https://github.com/Project-Bonfire/Bonfire

implemented for transmitting the flits between routers is different in the two architectures. It is also worth noting that none of the router architectures in Bonfire use VCs.

Bonfire Handshaking Router

The first router design in project Bonfire is a 32-bit wormhole switching NoC router with handshaking flow control. The names **Bonfire Handshaking Router** and NoC Router **Architecture 2**, are used interchangeably in the following chapters of this thesis, which refer to this router architecture. **Figure 2**.4 [66] shows an overview of the baseline handshaking flow control router without any fault-tolerance mechanisms.

Bonfire Credit-Based Router

The second router design in project Bonfire is a 32-bit wormhole switching NoC router which has credit-based flow control. The names **Bonfire Credit-based Router** and NoC Router **Architecture 3** are used interchangeably in the following chapters when referring to this router architecture. Figure 2.5 [66] shows an overview of the baseline credit-based router without any fault-tolerance mechanisms implemented in it. In credit-based flow control, the transmitter router keeps a credit counter, which is initially set to the number of free slots of the receiver router's input buffer. Each time a flit is sent, the counter gets decremented by one. In case the receiver passes a flit, it issues a credit signal which will increment the counter at the upstream router. One of the other differences of this architecture with the previous ones is the support of adaptive routing, when the routing logic might choose more than one output port as candidates for sending the flit. Therefore, even though the same FSM-based Round-Robin arbitration logic is used in router Architecture 3, its implementation is in two stages, one for handling multiple requests from an input, and the other for handling multiple requests from different inputs to the same output. The arbitration unit in router Architecture 3 is named *Allocator* (as shown in Figure 2.5 [66]).

One of the advantages of the credit-based router over the handshaking version is its better performance in terms of flow control. As long as the upstream router has valid data to send and it also has the credit, the transmission of flit(s) can continue.

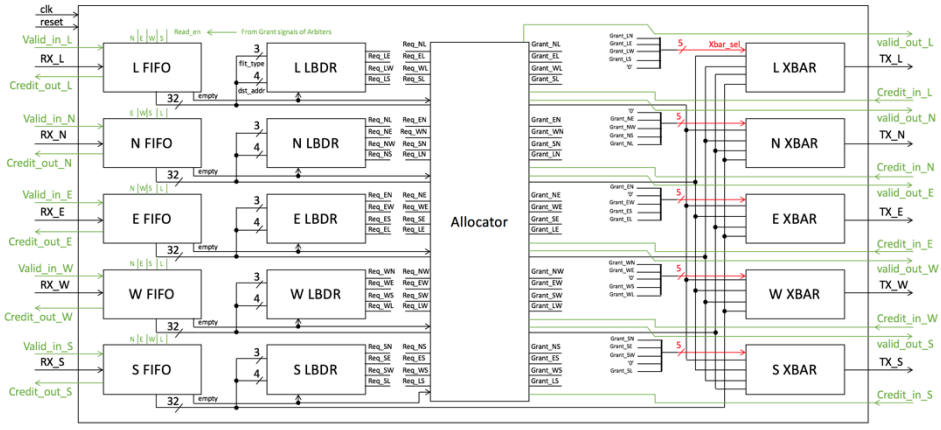


Figure 2.5 High-level overview of Bonfire NoC router with credit-based flow control

2.4 Logic-Based Distributed Routing (LBDR)

The LBDR mechanism [65] has been used as the routing computation component in all three NoC router architectures used in this dissertation. As the mechanism is going to be referred to in the coming chapters, a brief explanation of the mechanism is provided as follows.

LBDR has been introduced as a solution to implement different deadlock-free routing algorithms in 2D NoCs, while overcoming the scalability limitations of routing tables. LBDR removes the need for routing tables at all in NoC routers. It codifies the routing algorithm and topology in form of two sets of configuration bits, *i.e. routing* bits and *connectivity* bits. The former describes the routing algorithm, in form the set of allowed/restricted turns, whereas the latter describes the topology, showing the connection of each router to its possible neighbor(s). As opposed to routing tables, as the network scales up, the routing logic in LBDR is fixed, since it relies on the fixed sets of configuration bits. The logic of LBDR is shown in **Figure 2**.6 [65], [68].

LBDR is distributed, thus, for routing computation it only relies on the current address of the router and the address of the destination node included within the header flit of a packet. This, removes the need for encoding the routing path in the packets (as opposed to source-based routing [69]) at the source nodes. As shown in **Figure 2**.6, the mechanism is composed of two phases. In the first phase, the quadrant or direction in which the destination node is located compared to the current node is computed (N', E', W' and S' signals). In the second phase, using the connectivity and routing bits, the candidate output port(s) is (are) computed. In case of a deterministic routing algorithm, such as XY routing, LBDR always will choose one direction as the candidate one. However, in case of adaptive routing, two output ports can also be selected as candidates for

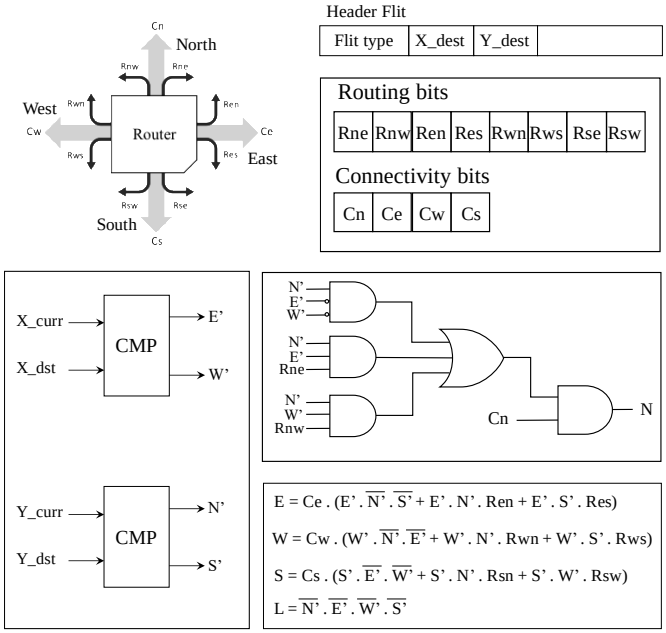


Figure 2.6 Logic of LBDR mechanism (first phase computes the location of destination, and the second phase computes the candidate output port(s).

forwarding the flit(s). It is worth noting that LBDR logic only becomes active when processing the header flit of a packet. Furthermore, when none of the signals in the phase first are active (the packet has reached its destination), the Local (L) output port is chosen as the candidate for forwarding the flit(s) to the Processing Element (PE) connected to the local port of the router. More details regarding how the LBDR mechanism works and implements different turn-model based routing algorithms in 2D NoCs, are provided in [65].

2.4.1 LBDR Extensions

Several extensions to LBDR have been introduced in the literature. In [68], LBDR has been extended to address collective communication (multicast and broadcast) in addition to unicast communication. Authors of [70], [71] have added de-route and fork capability to the mechanism, and proposed uLBDR, aiming to address link faults in 2D Mesh-based networks, and providing non-minimal path support. However, for the mechanism to work efficiently, the switching mechanism must be changed from wormhole to Virtual Cut Through (VCT), thus, imposing additional input buffer overhead to the routers. This issue is overcome by introducing d^2-LBDR mechanism in [72], which uses the wormhole switching, however, the mechanism is still limited regarding the increasing number of faulty links. In [73], one further step is taken and LBDR is augmented with support for irregular topologies in addition to regular 2D Mesh.

The scope of the proposed approach in [73], LBDRhr, is not limited to Mesh based 2D NoCs, but it has also been extended to support topologies with higher radix and routers with higher number of ports. In [27], in addition to tolerating permanent link failures, LBDRhr is also equipped with a set of fault monitors for tackling the detection of transient faults in the routing logic (which is part of the control part of router). These fault monitors are extracted using the Inherent Information Redundancy (IIR) [26] in the routing logic. The fault detectors (also called as checkers) for LBDR are extended in [74], which led to three-fold increase in the fault coverage. There have also been extensions to LBDR for addressing congestion-aware routing in 2D Mesh-based NoCs in [75], [76]. Despite the advantages each of the extensions to LBDR provide, the challenge of implementing fault-tolerant routing in 3D NoCs with partially vertically connected nodes has not been addressed yet using a scalable and re-configurable logic-based routing approach.

2.5 Chapter Summary

This chapter started with a comprehensive literature review, covering the state-of-the-art regarding the topics which are the focus of this dissertation. The objective was to provide an overview of the proposed online fault detection approaches for the control part of NoCs, online fault localization and fault-tolerant routing algorithms for partially vertically connected 3D NoCs, which all correspond to the contributions of this thesis. Moreover, the preliminary materials and terminology which used throughout different parts of the dissertation were explained, including the router architectures used in the thesis as examples for applying the proposed fault detection mechanisms and a background of different variations of logic-based routing in NoCs, related to the third contribution of this thesis.

3 ONLINE DETECTION OF FAULTS IN NETWORK-ON-CHIPS

3.1 Introduction

Online detection of faults in digital systems, including NoCs is important, as transient faults might only manifest themselves during system run-time. Especially, capturing faults in the control part of NoCs via an online detection mechanism is crucial, as such faults can cause mis-routing of packets, data loss or deadlock, leading to the breakdown of the whole system. However, the area overhead of the fault detection circuitry must not be unacceptably high, as the probability of faults occurring in the fault detection logic itself may also increase, which is not desirable. This chapter of this thesis proposes a methodology for devising concurrent online checkers for online detection of faults in control part of NoCs, while providing a trade-off between fault coverage and the incurred area overhead.

First, as a background, the concept of concurrent online checkers is provided, along with the fault model that would be the focus throughout this thesis. The literature review regarding previously proposed online fault detection techniques for control part of NoCs area already covered in Chapter 2. Next, the contribution of this chapter is explained in detail, which is a methodology for devising concurrent online checkers from the control part of a circuit in a systematic way, with the guarantee of single-cycle fault detection latency and minimizing checkers in terms of area while satisfying the target fault coverage. The methodology automates the process of devising two types of checkers, *i.e. structural* and *functional* checkers, which are both elaborated in this chapter. The proposed methodology has been applied to the control part of three different NoC router architectures as examples. The details regarding these NoC architectures are already covered in Chapter 2 as background. Experimental results regarding applying the framework to the control of a NoC router will show the trade-off between checkers' area overhead and fault coverage. Finally, a short summary of the chapter is provided. The contributions of this chapter of the dissertation have led to **publications A, B and D** listed in Chapter 1.

3.2 The Concept of Concurrent Online Checkers

One of the methods used to detect faults online in a circuit is the use of concurrent online checkers [4]. A checker is defined as a module monitoring the *correctness* of a design based on the rules defined for the functionality of that design, taking into account a specific fault model. In this dissertation, the focus is on NoC routers control part checkers.

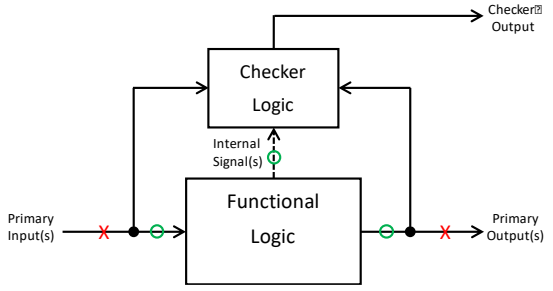


Figure 3.1 The concept of concurrent online checking of faults via checkers

Figure 3.1 presents a checker attached to a functional logic. As it can be seen, in addition to the original circuit (functional logic), a set of checkers (checker logic) are connected to functional inputs/outputs of the circuit. These checkers are derived based on the methodology that will be explained later in this dissertation. One set of checkers are in form of functional assertions obtained from relationships between variables corresponding to inputs and outputs (and possibly internal signals) of the circuit. The other set of checkers are devised in a systematic way by traversing the RTL of the design. The checker logic targets the faults at lines at the inputs of each gate within the functional logic (marked by green circles in **Figure 3**.1). The lines at the functional outputs succeeding the checker inputs (marked by a red cross in **Figure 3**.1) cannot be detected by the checker. In addition, the checkers are not targeting the faults at functional inputs preceding checker inputs, since the checker may not detect that the input value has been altered by a fault (such functional input lines are also marked by a red cross in **Figure 3**.1).

In this dissertation, both transient and permanent faults are modelled as Single Stuck-At-Faults (SAFs) occurring in single clock cycle. This information is used when evaluating checkers for the control part of the NoC router. By means of this fault model, the checkers cover SEUs (in form of transient faults) [77] and permanent faults. As example, the proposed methodology has been applied to the control part of three different NoC router architectures. It is worth noting that in this dissertation, it is already assumed that the data-path of the NoC router is already protected by an error detection/correction technique [15], [78].

3.3 Methodology for Devising, Evaluating and Minimizing Concurrent Online Checkers for Control Part of NoC Routers

This Chapter focuses on the proposed methodology for devising, evaluating and minimizing concurrent online checkers for the control part of circuits.

Figure 3.2 illustrates the proposed flow of the methodology. The flow starts by taking into account the control part of the circuit. Next, it is followed by synthesizing the pseudo-combinational version of the circuit under check and devising the initial set of checkers from a set of combinational assertions. Additional checkers that also describe relations on the pseudo primary inputs/outputs may be added to the checker suite in order to increase the fault coverage. The initial set of checkers includes a set of structural and a set of functional checkers. Subsequently, the checker evaluation environment is created during the environment generation step by generating exhaustive valid set of input stimuli which will serve as the environment for checker evaluation. If there is no bug in the environment and no bug in the checkers, the fault-free simulation step would confirm that. The checkers evaluation is performed afterwards, which leads to measuring the fault detection quality of checkers in terms of metrics, such as CEI (Checkers Efficiency Index), FC (Fault Coverage) and FPR (False Positive Ratio) and the checkers' weight information (number of True Detections) and their corresponding area consumption. The information acquired from this step is used for the minimization process using a greedy heuristic, which provides a trade-off between the fault coverage of the checkers and their area overhead. The final results of the methodology would be

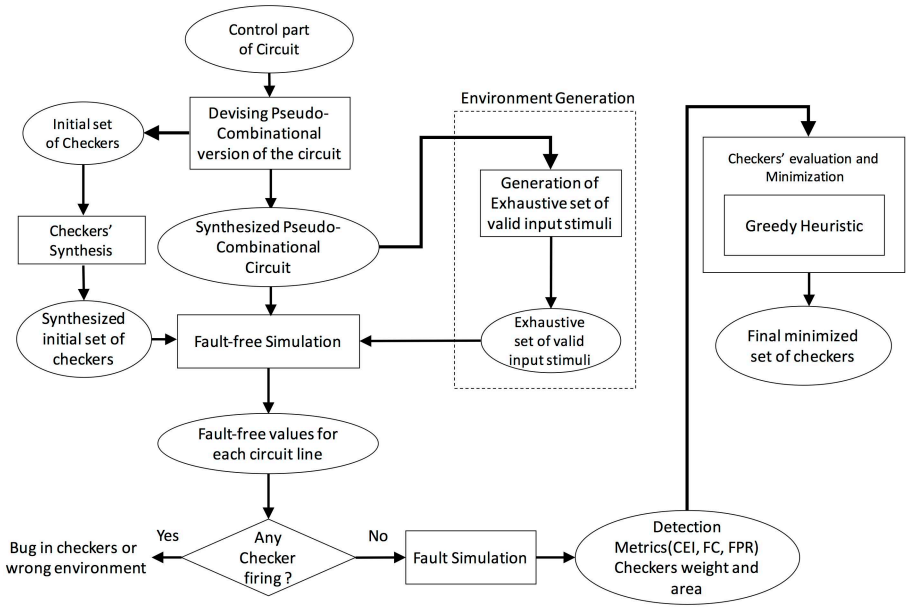


Figure 3.2 Checkers Evaluation and Minimization Flow

the minimized set of checkers in terms of area, meeting specified target fault coverage. In the following sub-sections, each step of the proposed flow is explained in detail.

3.3.1 Devising Pseudo-combinational version of the circuit under check

In order to evaluate the checkers for all possible fault locations and under all possible valid input stimuli, the methodology shown in **Figure 3**.2 first needs some preparation steps. This includes extracting a *pseudo-combinational* equivalent of the module under check

The pseudo-combinational circuit is derived by breaking the Flip-Flops and memory elements (such as registers) and converting them to pseudo-primary inputs and pseudo-primary outputs, as shown in **Figure 3**.3. **Figure 3**.3a illustrates a sequential circuit with its primary inputs and outputs, while **Figure 3**.3b demonstrates its pseudo-combinational equivalent circuit, which has additional pseudo- inputs and outputs. In the pseudo-combinational circuit, the *current state* signals are converted to pseudo-primary inputs and *next state* signals are converted to pseudo-primary outputs (**Figure 3**.3b). This would facilitate the process of evaluating the checkers under all possible valid input stimuli and making it possible to formally prove the presence or absence of cases of True Detection and True Misses (which will be explained later in this chapter).

It should be noted that even though this step of the methodology might lead to creation of additional inputs/outputs, at the end of the proposed flow, the checkers are integrated in the sequential design, therefore, the final structure of the module under check is not altered. Once the design to be checked is prepared, the next part of the flow comes into play, which is devising the concurrent online checkers. It is worth noting that in this dissertation, the control part of a NoC router is used as an example of the circuit under check, from which the pseudo-combinational circuit is extracted.

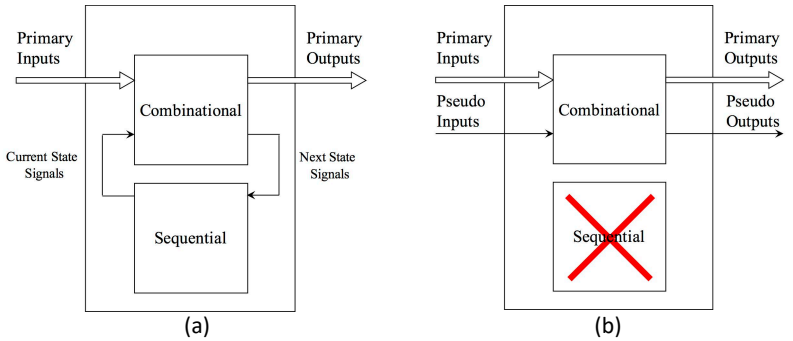


Figure 3.3 a) A sequential circuit and b) its equivalent pseudo-combinational circuit

3.3.2 Devising Initial Set of Checkers

The methodology proposed in this work devises two types of checkers: *functional* and *structural* checkers. The functional checkers take into account the functionality of the module under check. They are not automatically devised and it needs the verification engineer involved in the devising process. To this end the specification of the module under check is also taken into account. On the other hand, structural checkers are devised systematically and in an automated manner by parsing the RTL description of the design. They check all the different paths through which the RTL code can be executed and examine whether based on those paths, the generated output(s) is (are) correct and valid. Examples of devising both types of checkers from the control part a NoC router are provided in Sub-Section 3.4.1 and Appendices A and B.

The functional checkers might have overlaps in terms of the domain of the circuit they are checking. In case of structural checkers, they check distinguished non-overlapping parts of the circuit. It should be noted that after devising both set of functional and structural checkers using the proposed methodology, one cannot necessarily predict which type of checkers would outperform the other. As it will be explained later, considering both checkers when performing the evaluation and minimization heuristics using the proposed flow, can make the search space exploration more efficient, pruning checkers with overlap, but keeping the necessary ones to satisfy the area budget, while still guaranteeing the target fault coverage.

Functional and Structural Checkers

Functional checkers are devised by the verification engineer, from functional assertions. Such assertions are obtained from relationships observed between variables corresponding to inputs and outputs (or possibly the internal signals) of the circuit. Such checkers are not devised by parsing the RTL code of the control circuit. Instead, they are designed by the verification engineer taking into account the specification of the design. *i.e.* checking the rules that must hold in order to confirm that the circuit is working. In the proposed methodology, each checker is evaluated for all possible fault locations in the pseudo-combinational version of the circuit and under all possible values for valid input stimuli.

Structural checkers are extracted from the RTL code of the design. The methodology for devising structural checkers from the control part of the NoC router traverses through all different paths in the RTL code in a systematic way, and devises checkers for each

condition in the pseudo-combinational version of the circuit under check, which would have a relation between an input signal and an internal signal/output signal. It should be noted that structural checkers, examine very specific behaviour(s) of the circuit and usually, they do not check the same part of the circuit.

3.3.3 Environment Generation for Checkers' Evaluation

Before performing any logic simulation or fault simulation for evaluating the checkers, the environment under which the checkers are going to be evaluated must be generated. This is handled by the next step of the proposed flow of the methodology, which is the environment generation section (as shown in **Figure 3**.2). The checker evaluation environment is created by generating exhaustive test stimuli for the extracted pseudo-combinational circuit. These stimuli are fed through a filtering tool that selects only the stimuli that correspond to functionally valid inputs of the pseudo-combinational circuit. It is important to note that the checkers will later be evaluated only under the set of valid input stimuli and not the exhaustive set of all possible input patterns. As a result of this step, the complete valid set of input stimuli that will serve as the environment for checker evaluation, is obtained. In Sub-section **3**.4examples of applying the methodology to the control part of a NoC router will be provided, which will show the constraints that guide the filtering tool to generate the valid set of input stimuli for the modules under check.

3.3.4 Fault-Free Simulation and Debugging Checkers

The obtained environment from the previous step, the pseudo-combinational circuit and the synthesized checkers (structural and functional checkers) are applied to fault free simulation. The simulation calculates fault free values for all the circuit lines. Additionally, if any of the checkers fires during fault-free simulation, it means there is a bug in the checker or the evaluated environment is incorrect. This facilitates the process of debugging the checkers. If none of the checkers fire during fault-free simulation, the checker evaluation step of the proposed methodology flow takes place (as shown in **Figure 3**.2).

3.3.5 Fault Simulation of Checkers

The checker evaluation step is performed using a fault simulator developed as an extension of a freeware test system Turbo Tester [79]. The system applies Structurally Synthesized Binary Decision Diagram (SSBDD) models [80] for circuit modelling. Turbo Tester injects faults to all the lines within the circuit one-by-one and this step is repeated for each input vector. More specifically, faults are considered at the inputs and outputs of all the fan-out free regions within the circuit. It is worth noting that unlike approaches such as NoCAlert [23], [24] which treat the module under check as a black box during high-level evaluation of the checkers, our proposed methodology, along with the fault simulation tool, considers the internal signals of the design for which the checkers are devised as well. As a result, the overall fault detection metrics (discussed later in this chapter) for the set of checkers will be calculated.

What makes this work different from previous approaches regarding online checkers is that all the experiments for evaluating the checkers are based on *fault simulation*. Traditionally, in order to evaluate the fault detection quality of the checkers, *fault injection* has been applied (e.g. the approaches in [23], [24], [81]). Fault injection refers to injecting faults into a circuit at a certain time step and simulating it with the input stimuli to see whether any functional output of the circuit changes and whether any of

the checker outputs fire. Due to the fact that it is generally impossible to inject and simulate all the faults at each circuit line at each time step, a statistically significant sample of random faults would normally be injected and simulated.

However, as mentioned earlier, the proposed methodology in this dissertation is based on automated extraction of a pseudo-combinational circuit out of the original functional logic. Further, an exhaustive test for the extracted circuit is fed through a filtering tool in order to derive the complete valid set of input stimuli which will serve as the environment for checker evaluation. This means that in the proposed flow in this dissertation, full evaluation of the checkers with all the valid stimuli and faults is obtained through fault simulation. The advantage of fault simulation over fault injection is that for considering different fault locations, there is no need for simulating the circuit at all possible time instances. Only one fault simulation run would be sufficient to analyse the effect of the faults [80].

Metrics used for Checkers' Evaluation

After the overall fault simulation step for evaluating the set of checkers, the results of their fault detection quality should be defined. This is what is performed in the next step of the flow of the proposed methodology.

Given a fault at a line within the functional logic and a set of input stimuli, four possible scenarios can occur:

- ***Case 1:*** Fault occurs at an internal line and is visible at functional output(s) and checker logic flags a violation. The term *True Detection* is used to describe this situation, since a critical fault is effectively detected by the checker.

- ***Case 2:*** Fault occurs at an internal line but is not visible at primary output(s). Checker catches the fault and flags a violation. The term *False Positive* is used to describe this situation. False positive is not harmful because an error is flagged which did not have any effect. However, it has negative impact on design's performance because normally it causes re-execution of the task.

- ***Case 3:*** Fault occurs at internal line but is not visible at primary output(s) and the checker logic does not detect the violation. The term *Benign Miss* is used to describe this situation. Benign miss shows correct operation by the checker.

- ***Case 4:*** Fault occurs at internal node and is visible at primary output(s). Checker does not detect violation. The term *True Miss* is used to describe this situation, which is the worst possible case. True miss means that the fault propagates to the functional outputs and onwards to the system. However, the system has no information that a critical fault has occurred.

Let *D* be the number of True Detections, *X* be the number of Benign Misses, *W* be the number of True Misses and *F* be the number of False Positives over all the injection runs. In the proposed flow, the evaluation of the fault detection quality of the checkers based is performed using the following metrics: *Fault Coverage* (*FC*), *Checkers' Efficiency Index* (*CEI*) and *False Positive Ratio* (*FPR*).

One of the contributions of this thesis (as also stated in [3]) is being able to formally prove the presence or absence of True Misses, which has not been addressed in the previous works such as [23], [24].

$$FC = \frac{D + X}{D + X + W} \qquad \textit{(Equation 3.1)}$$

$$CEI = \frac{D}{D + W} \qquad \textit{(Equation 3.2)}$$

$$FPR = \frac{F}{F + X} \qquad \textit{(Equation 3.3)}$$

Here, *FC* shows the probability of the checkers behaving correctly over all possible fault cases (in addition to True Detections and True Misses, it also takes Benign Misses into account), *CEI* shows the probability of checkers' ability to detect critical faults (it covers the cases that the fault is propagated to the output of the circuit, either detected by the checkers or not), whereas *FPR* reports the ratio of false positives over all the cases a fault did not propagate to circuit outputs. It is worth noting that in none of the experiments of this dissertation, checkers resulted in false positive (*FPR* was zero). This is based on our assumption that False Positives do not occur in our experiments and that is why they are excluded from the formulas for *CEI* and *FC* calculation in Equation 3.1 and Equation 3.2. For reference purposes though, the formula used to calculate *FPR* is also provided, in Equation 3.3 [82].

3.3.6 Checkers' Evaluation and Minimization

In this step of the proposed methodology, after the fault simulation, the values for CEI and FC and FPR when considering all the checkers, are calculated. The goal is to reach 100% coverage for SEUs both for CEI and FC. A 100% CEI would mean that there were no cases of True Misses during the fault simulation and thus checkers are able to capture all SEUs at different locations in the design. In addition, each individual checker will be weighted by summing up the total number of True Detections by the checker. This information is used for the next step of the flow, which is the optimization and minimization of checkers in terms of area.

Even though having the full set of checkers, devised for a design, might cover all the faults due to SEUs, integrating all the checkers in the final design can impose significant area overhead to the system. This can be mitigated using a methodology that would analyse each checker one by one finally, choosing only the checkers that are necessary for obtaining the target fault coverage, while consuming less area compared to the initial set of checkers. This would save significant area in case some checkers cover other checkers, *i.e.* one checker captures the same faults another checker can capture and on top of that it detects some additional faults.

The weighting information obtained from the evaluation part of the proposed methodology (the number of True Detections for each checker) will be exploited in minimizing the number of checkers, eventually allowing to outline a trade-off between CEI (and FC) and the area overhead due to the introduction of checker logic. The minimization part of the flow is performed using a greedy heuristic. To this end, the checkers are sorted based on their weight. *i.e.* based on the descending values of True Detections. Then, the checker with the highest weight is chosen and fault simulation is performed and the process is performed by considering each checker with the next

highest weight until the target FC and CEI is reached. In the following sub-sections, some examples of applying the framework to the control part of three NoC routers will be provided, which will show the efficiency of the proposed methodology for devising checkers for the design and the minimizing the checkers in terms of area in order to reach a trade-off between area overhead and CEI (and FC).

3.4 Application of the Proposed Methodology to the Control Part of a NoC Router

As mentioned earlier, the proposed methodology for devising, evaluating and minimizing concurrent online checkers is not limited to the control part of a specific NoC router architecture. To this end, three examples of applying the methodology to the control part of NoC router **Architecture 1**, **Architecture 2** and **Architecture 3** (previously explained in Chapter 2), are provided. However, since the underlying procedure for devising the pseudo-combinational version of the circuit, checkers devising, fault simulation, checkers' evaluation and minimization are similar, only one of the examples is explained in detail in this chapter, *i.e.* NoC router **Architecture 1**. The complete set of devised checkers for the control part of NoC router **Architectures 2 and 3** (Bonfire handshaking and credit-based NoC routers) are listed in Appendices A and B, respectively.

3.4.1 Example: Devising Checkers for the Control Part of NoC Router Architecture 1

In the first example, the proposed methodology is applied to the control part of NoC router Architecture 1 (explained in Chapter 2). The example consists of three experiments. In the first experiment, the control part is only limited to the routing logic (LBDR), more specifically the LBDR of East input port (ELBDR), as shown in **Figure 3**.4. The pseudo-combinational version of ELBDR has the flit type, destination address, empty signal, and the previous values of the output requests as its inputs. The existing output port signals for ELBDR are N, W, S and L (according to **Figure 3**.4).

In the first experiment, LBDR connectivity bits and routing bits and the current address of the router are all hardcoded in the logic, which corresponds to the following scenario: 2D Mesh topology, XY routing algorithm, U-turns not allowed, focus on router with ID 5 in a 4x4 2D Mesh network. This scenario allows minimizing the number of circuit

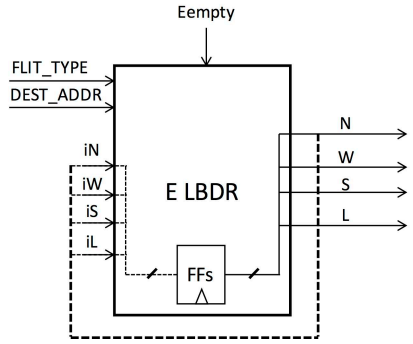


Figure 3.4 The pseudo-combinational circuit for the scenario with LBDR of East port for NoC Router **Architecture 1**

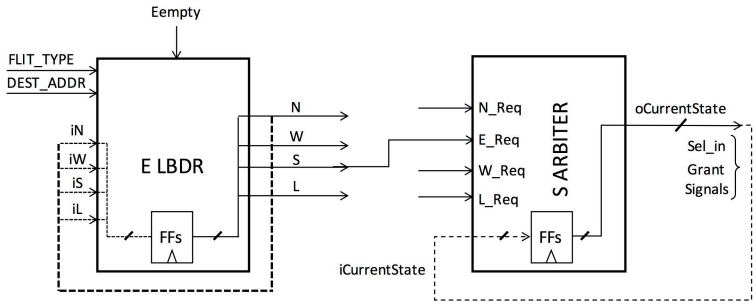


Figure 3.5 The pseudo-combinational circuit for the full scenario of connecting LBDR of East port to Arbiter of South Output port

inputs and previous request values input bits that together form the inputs for the pseudo-combinational circuit of ELBDR.

When only ELBDR is considered, the amount of inputs is limited to 11 bits:

- 2 flit type bits;
- 4 destination address bits;
- 4 ELBDR previous output values bits;
- 1 empty bit (coming from East input buffer (FIFO)).

For the second experiment, the control part is extended to LBDR and the arbitration logic (arbiter), illustrated in **Figure 3**.5 [3]. The modules which have the most number of connected signals are chosen, *i.e.* ELBDR (LBDR for East input) and SArbiter (Arbiter for South output). The output request port signals for ELBDR are the same as the first experiment, and for SArbiter, request and grant signals exist for N, E, W and L (each grant signal corresponds to a request signal). Similar to the previous experiment, the following assumptions have been made: 2D Mesh topology, XY routing algorithm, U-turns not allowed, focus on router with ID 5 in a 4x4 2D Mesh network, and unicast communication.

With the interconnection of ELBDR to SArbiter in the second experiment, the number of input bits is increased to 19:

- 3 SArbiter request signals bits;
- 5 SArbiter previous state bits (*iScurrentState*) (which are used in the internal FSM of SArbiter for prioritizing the input requests).

The reason that the above-mentioned scenarios are chosen for the first and second experiment is that such scenarios provide the case with the most number of connections signals between LBDR and arbiter logic. The checkers that cover faults for such scenario, are symmetrical to the other cases (different connections between each LBDR logic to arbiter logics).

Once the first preparation step for the proposed methodology has taken place and the pseudo-combinational circuit to be studied is extracted, two sets of checkers are devised, one from the functional behaviour of the considered circuit, evaluating the possible implications existing in between input and output signals and the other one the structural checkers, which are devised by traversing all different possible paths in the RTL

of the design under check. It should be noted that *a priori* it may be very difficult to outline the effectiveness of a single checker or the overlap of different checkers in detection (in terms of the domain(s) of the circuit they are checking).

Together with the considered pseudo-combinational circuit and its sets of checkers, a set of input patterns is needed for performing fault simulation. The exhaustive test would require 2^{11}=2,048 and 2^{19}=524,288 input stimuli, for the ELBDR and for the ELBDR + SArbiter control path experiments, respectively. However, in order to minimize the number of stimuli, and more important, to avoid checkers being evaluated in non-realistic conditions, the exhaustive set of stimuli has to be filtered to contain only the functionally feasible values.

The filtering step of the proposed methodology is based on the implemented routing algorithm (i.e. allowed destinations from the current router), restrictions in the routing logic (e.g. no U-turns) and emptiness condition of the input buffer (FIFO) (for the first experiment), as well as invalid conditions for the state variable of the arbiter logic (i.e. violation of one-hot encoding) (for the second experiment). The constraints existing for the inputs of the third experiment are also explained in detail when experiment 3 is described. It is important to stress the fact that none of the checkers fired in fault free simulation with any of the considered input stimuli, in neither of the experiments. The filtering of the exhaustive set of stimuli led to a final set of 1536 vectors and 61440 vectors for the ELBDR and ELBDR+SArbiter scenarios, respectively.

3.4.2 Summary of Experimental Results

Experiment 1: ELBDR Scenario

By applying the proposed methodology to ELBDR of NoC router **Architecture 1**, the initial set of functional and structural checkers was devised for its pseudo-combinational equivalent circuit, as listed in **Table 3**.1.

Table 3.1 Proposed Checkers for LBDR of East Input Port (ELBDR)

Checkers for Routing Logic (LDBR)		
1	Valid LBDR output	If there is a request to the routing logic (the corresponding input buffer is not empty), LBDR has to compute at least one valid output direction (according to XY routing).
2	No LBDR output	If no flit arrives (the corresponding input buffer is empty), all the output port signals of LBDR should remain zero.
3	Single LBDR output	If the corresponding input buffer is not empty (there is a request to LBDR), because of using XY routing, at most only one output port signal of the LBDR logic can become active.
4	Switch LBDR output	If the corresponding input buffer is not empty (there is a request to LBDR) and a non-header flit has arrived, LBDR outputs should remain the same.
5	Local Port output	If the corresponding input buffer is not empty (there is a request to LBDR) and a header flit has arrived, the local output should become active only if the packet has reached its destination.

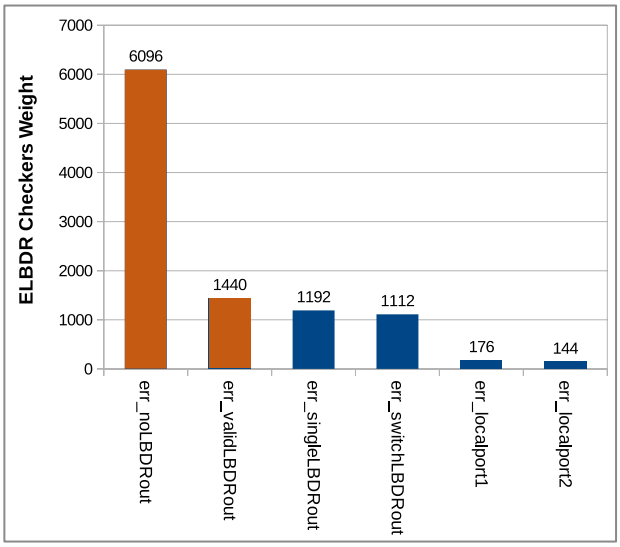


Figure 3.6 Weights of devised checkers (number of True Detections) for EBLDR

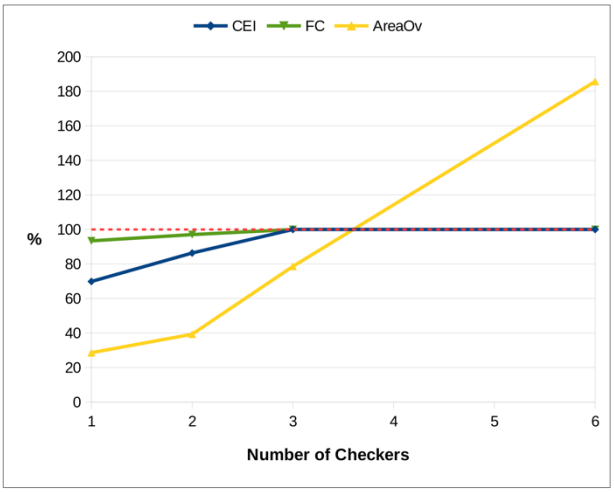


Figure 3.7 ELBDR scenario FC, CEI and area overhead results

In order to evaluate the checkers under all possible values of valid input stimuli, the following situations have been considered in the filtering tool of the proposed methodology:

- If input buffer's empty signal is high, any other input bit is meaningless, and therefore any value is allowed for it.
- If the incoming flit is a header, the destination address has to be valid according to the XY routing and U-turn restrictions.
- If the incoming flit is a body or tail flit, the previous output request values must be valid, and they must follow a one-hot fashion, according to XY routing.

Figure 3.6 displays the weight information output of the checkers' evaluation, corresponding to the initial set of checkers for the ELBDR. The values basically indicate the number of True Detections.

These True Detections quantities were evaluated by iterating the fault simulation, including at each step the next heaviest checker (checker with highest number of True Detections) still not included in the currently considered set of checkers, initialized only with the first heaviest checker. By performing the greedy heuristic on the initial set of 6 ELBDR checkers, it was observed that when the 3 most significant checkers were used (*i.e.* checkers *err_noLBDRout*, *err_validLBDRout* and *err_singleLBDRout* in **Figure 3**.6 [3] with highest number of True Detections), CEI and FC could reach 100% without encountering any cases of True Misses using the valid set of input stimuli. The advantage is that the final set of three checkers impose an area overhead of 78.57%, which is much less than the area overhead of the initial set of checkers (185.71%) (as shown in **Figure 3**.7 [3]). It is worth noting that the results obtained by greedy heuristic is not necessarily the most optimal result always, but, it is a sub-set of sub-optimal results.

Experiment 2: ELBDR and SArbiter Scenario

In this scenario, the LBDR of East input port of the NoC router **Architecture 1** is connected to the arbiter of the South output port (ELBDR is connected to SArbiter), according to **Figure 3**.5. Using the same methodology for devising the initial set of functional and structural checkers, for the scenario of ELBDR and SArbiter, an initial set of 28 checkers was devised only for the SArbiter logic. However, as it will be shown later, for the ELBDR + SArbiter scenario, when evaluating the checkers for both units together, only the 3 checkers for ELBDR chosen previously by the minimization flow (not the total initial set of 6 checkers for ELBDR) are considered along with the 28 checkers of SArbiter. The checkers devised for SArbiter are grouped and listed in **Table 3**.2 [3].

Table 3.2 Proposed Checkers for the Arbiter Logic of South Output Port (SArbiter)

Checkers for the Arbiter logic		
6	Valid Grant output	If there is a request from LBDR, arbiter has to assert at least one of the grant signals for the corresponding output direction.
7	No Grant output	If there is no request to the arbiter, it should not assert any of the grant signals for any direction.
8	Invalid Grant output	Whenever there is a request to the arbiter, the grant signals should go active corresponding to that specific requested direction and invalid direction should not be chosen.
9	Invalid arbiter output state	Output state variable (oScurrentState – which represents the grant signals) in arbiter's pseudo-combinational circuit can not possess invalid values due to the one-hot coding.
10	Invalid IDLE state for arbiter input state	If the input previous state variable (iScurrentstate) is in IDLE state and there is a request for arbitration from LBDR, oScurrentstate should not remain in IDLE state i.e. a grant signal should be asserted.
11	Priority Grant	In case there is one or multiple request(s) to the arbiter, it should follow the correct prioritization (Local, North, East and then West) according to the input previous state variable (iScurrentstate).

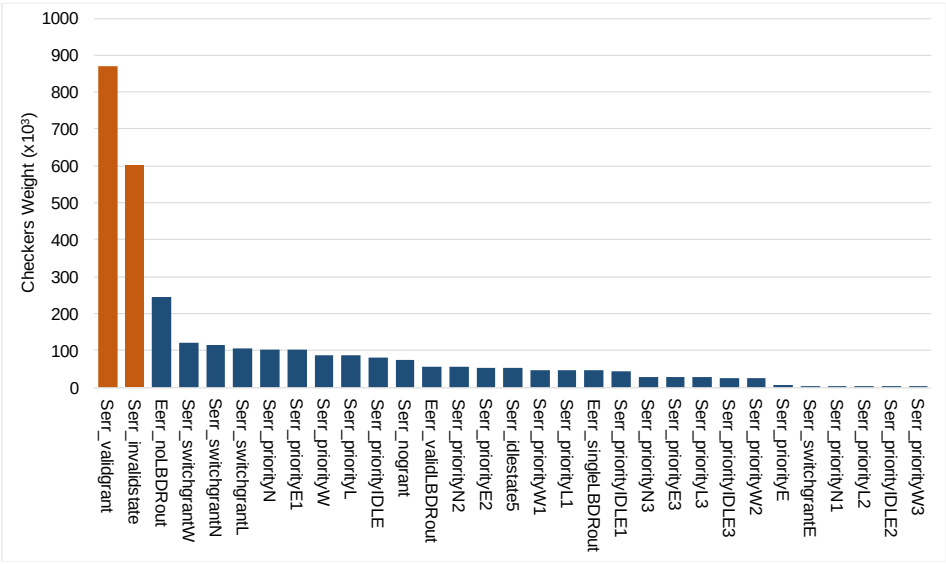


Figure 3.8 Weights of checkers proposed for the EBLDR and SArbiter scenario

In this scenario, in order to evaluate the checkers under valid input stimuli, the considered filtering scheme is an extension of the one used for the ELBDR experiment. In addition to the previous constraints, the new set of input patterns include adding the one-hot encoding constraint to the 5 previous state value bits (*iScurrentstate*) of the SArbiter's pseudo-combinational unit.

First, the evaluation tool was run considering the whole set of checkers for the SArbiter (28 checkers), altogether with only the minimized set of 3 checkers for the ELBDR, which led to a total set of 31 checkers. Similar to the previous experiment, the weights of the checkers (number of True Detections) are provided using the fault simulation part of the proposed methodology and listed in **Figure 3**.8 [3] in descending order. Focusing on the Sarbiter, it is observed that the two checkers monitoring different aspects of the one-hot encoding condition for the arbiter's state variable, have the highest weights.

Applying the greedy heuristic to the initial set of 31 checkers (3 for ELBDR and 28 for SArbiter) led to the final minimized set of 3 checkers for ELBDR and 2 checkers for SArbiter. With this final set of 5 checkers, it is still possible to reach 100% CEI and FC for single stuck-at faults for the ELBDR and SArbiter scenario. In case of SArbiter, the area overhead of the final minimized set of 2 checkers (56.82%) is much less compared to using the whole initial set of 28 checkers which would impose an area overhead of 170.45%. One of the observations that was also has been made during the experiments is that the two set of checkers for the ELBDR and the SArbiter are independent*, i.e.* they cover faults for different and separate parts of the circuit, without any overlap. This observation will be explained later in this sub-section.

It is interesting to note that the minimized set of 5 checkers for the ELBDR and SArbiter scenario corresponds to one-third of the whole 31 checkers set area. **Figure 3**.9 [3] shows the CEI, FC and area overhead results for the experiment in which the checkers for only the SArbiter module are evaluated. As it can be noticed in **Figure 3**.9 [3], with

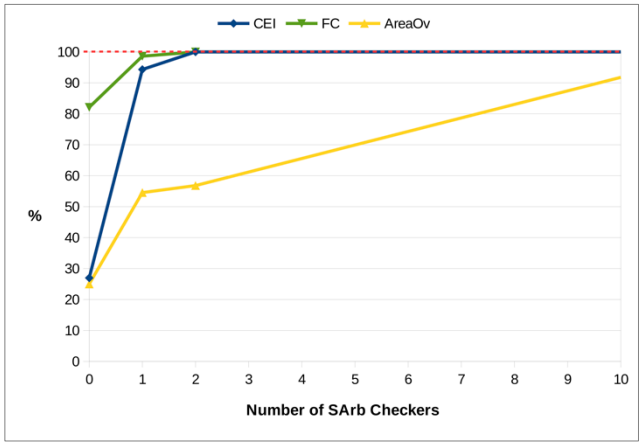


Figure 3.9 SArbiter scenario FC, CEI and area overhead results

Table 3.3 Weights for minimized set of checkers

Checker Name	Weight (No. of True Detections)
Serr_validgrant	871552
Serr_invalidstate	600512
Eerr_noLBDRout	243840
Eerr_validLBDRout	57600
Eerr_singleLBDRout	47680

only 2 SArbiter checkers (out of 28 checkers) which have the highest number of True Detection, it is possible to reach 100% CEI and FC, while imposing less than 60% area overhead.

The final minimized set of 5 checkers along with their corresponding weights are listed in **Table 3**.3 [3].

Impact of clustering the faults for ELBDR and SArbiter scenario: One of the observations that was also made during the first and second experiment was that the two set of checkers for the ELBDR and the SArbiter are independent*, i.e.* they cover faults for different and separate parts of the circuit, without any overlap. Therefore, for the ELBDR and SArbiter scenario, even though the control part consists of a path from ELBDR to SArbiter, 100% fault coverage for SArbiter does not necessarily mean that they have also covered all the faults occurring in ELBDR. For this reason, the minimized set of ELBDR checkers is used, and the previously introduced weight-based greedy minimization heuristic is applied to the SArbiter checkers set for the ELBDR + SArbiter scenario.

Assuming that there was no information of the overlap of faults detected by the checkers for ELBDR and SArbiter, the weight-based greedy heuristic, starting from the heaviest checker, would add at each step the next heaviest checker (from the whole set of 31 checkers) still not considered in the current set of checkers, based on the weight information displayed in **Figure 3**.8 [3].

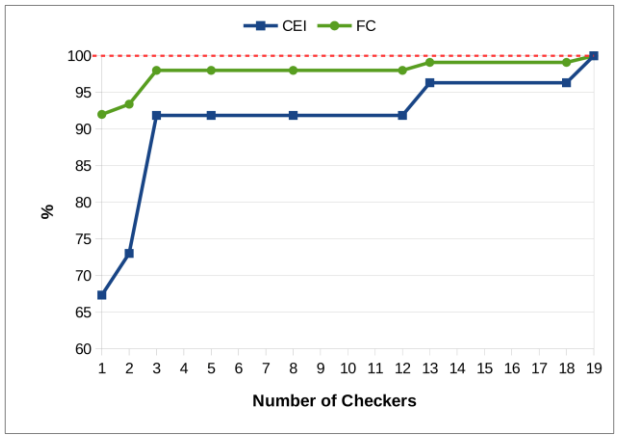


Figure 3.10 CEI and FC results without considering independent clusters

Figure 3.10 [3] shows the inefficiency of the heuristic approach caused by the lack of the clustering information. The number of steps in the greedy procedure is heavily increased, and only after 19 steps, when the *Eerr_singleLBDRout* checker for ELBDR is considered, the 100% upper bound for CEI and FC is reached with large area overhead.

However, when partitioning the fault set to clusters is taken into account and minimization is performed on the clusters separately, then total of 5 checkers are needed. **Table 3**.3 [3] illustrates the importance of considering the clustering information. As it can be observed in the table, the weights of the ELBDR checkers are far less than those of the SArbiter, but they are still needed to achieve full coverage for the considered design.

Experiment 3: FIFO Control Part Scenario

Fault Injection Experiments for the FIFO: Using the methodology proposed in this dissertation, the experiments were extended to the full control part of the router **Architecture 1**, adding control part of FIFO to the circuits under check (this experiment is included in **publication B** [82] mentioned in Chapter 2, included to this thesis).

For the FIFO's control part, an initial set of 8 checkers (functional and structural) were devised from the verification assertions. These checkers are grouped and listed in **Table 3**.4 [82]. As mentioned in [82], it should be noted that additional checkers are devised from temporal assertions for modules that do not achieve 100% fault detection. For these checkers the formal qualification step described in the proposed flow of methodology in this dissertation was not possible at the moment of writing the paper and thus, traditional fault injection experiments were carried out by a sequential fault simulation tool included to the methodology flow.

However, in experiments 1 and 2 and also in the experiments for the control part of Bonfire NoC router Architecture 2 (included in Appendix A and also published in **publication D**), the checkers' evaluation process is carried out using fault simulation.

For evaluating the FIFO control part checkers in Experiment 3, a set of input stimuli for the FIFO was devised, aiming to cover all the possible situations for the control logic.

Table 3.4 Proposed Checkers for control part of FIFO

Checkers for FIFO control part		
1	Reset checker	Whenever reset goes high, at the next clock cycle empty flag should be high (reading and writing pointer are reset to the same value).
2	Flags checkers	Empty and full flags should never be high at the same time. Whenever the defining condition occurs, the corresponding flag should go high at the next clock cycle.
3	One-hot pointers checkers	Reading and writing pointers have to respect one-hot encoding.
4	Registers enable DMR checker	Duplication and comparison for the logic enabling the writing operation in data registers.
5	Reading pointer update checker 1	Whenever read enable is high and the FIFO is not empty, at the next clock cycle the reading pointer should be updated.
6	Reading pointer update checker 2	If either read enable is low or the FIFO is empty, at the next clock cycle the reading pointer should preserve its value.
7	Writing pointer update checker 1	Whenever write enable is high and the fifo is not full, at the next clock cycle the writing pointer should be updated.
8	Writing pointer update checker 2	If either write enable is low or the fifo is full, at the next clock cycle the writing pointer should preserve its value.

Table 3.5 Proposed Checkers for FIFO's Control Part Infrastructure

Control Part Infrastructure Checkers		
1	FIFOs read enable DMR checker	Logic producing read enable signals for the FIFOs (5 OR gates) is duplicated, then real and duplicated outputs are compared.
2	Output registers enable DMR checker	Logic producing enable signals for the output registers (5 OR gates) is duplicated, then real and duplicated outputs are compared.
3	Flit type LBDR error	Flit type field of a flit has to respect one-hot encoding.

The following conditions were considered in the pattern generation procedure:

- Reset condition;
- Filling the FIFO, followed by reading from it until it becomes empty;
- Smooth traffic condition, *i.e.* concurrent writing and reading operations, but avoiding the FIFO to get full;
- Idle condition, *i.e.* write and read enable signals low, during reading and writing operations, in different conditions of fulfilment of the FIFO.

Using fault injection experiments with the checkers listed in **Table 3**.4 [82], 100% FC and CEI is obtained for the control part of the FIFO, considering the patterns derived from the previously listed conditions. Similar to previous experiments, no false positives were encountered in this experiment.

However, as mentioned earlier, achieving 100% FC and CEI became possible with the addition of new checkers obtained from the fault injection experiments for the

control part of FIFO. This was performed in order to identify uncovered faults in the interconnections of control part modules (as stated in [82]). The 3 additional checkers proposed for the infrastructure of FIFO's control part are listed and grouped in **Table 3**.5 [82]. Full details regarding the third experiment and also the impact of the router's data-width on the checker's area overhead are reported in **publication B** (mentioned in the list of publications in Chapter 1).

3.5 Applicability of the Proposed Methodology to Control Part of Any NoC Router Architecture

One of the targets of the first contribution of this thesis is to keep the proposed methodology as generic as possible. Of course, if the router architecture under check changes, depending on the structure of the control part modules and their RTL code and the specification, the checkers devised for that router would change. However, the principles and the basis of the methodology would still remain the same, which includes identifying the control part modules, extracting the pseudo-combinational version of the module under check and providing the appropriate environment as inputs and devising the two sets of structural and functional checkers, and finally evaluating the checkers in terms of CEI and FC and minimize in terms of area, in an automated manner.

The complexity of the design can indeed affect the process of minimization when performing the greedy heuristic. As some further examples of providing proof of applicability of the proposed methodology to control part of routers, two other architectures are studied. To this end, the methodology is applied to the control part of Bonfire handshaking and credit-based router architectures, explained earlier in Chapter 2. However, for the sake of repetition, the set of checkers for **Architectures 2** and **3** are listed in Appendices A and B, devised using the same methodology proposed in this dissertation.

3.6 Chapter Summary

In this chapter, the first contribution of this dissertation was proposed and explained in detail, which was a methodology for devising concurrent online checkers for the control part of a NoC router (regardless of its architecture). The proposed methodology has been applied to the control part of three different NoC routers. The proposed methodology is able to reach 100% coverage for online detection of faults caused by SEUs and single stuck-at faults in the control part of the NoC router for all three considered examples.

Moreover, the two sets of checkers devised for the circuit, *i.e.* functional and structural checkers, guarantee single cycle fault detection latency, along with formal proof of True Misses. In addition, the automated minimization part of the proposed methodology uses greedy heuristics, providing the opportunity to reach a trade-off between area overhead of the checkers and the target fault coverage with the minimized set of checkers. The methodology proposed in this chapter has led to **publications A and B** [3], [82]. Also, the same methodology was applied to the control part of Bonfire handshaking NoC router, which led to **publication D** [83].

As a conclusion, the final area overhead of the minimized set of checkers conforms to the statement mentioned in [4]: "In practice, a method of concurrent checking is of interest if the necessary area is considerably smaller than the 220–250% of the area of the functional circuit needed for duplication and comparison, and if the probability of detecting errors due to single stuck-at faults is about 90%+x."

4 FAULT LOCALIZATION AND ABSTRACTION IN NETWORK-ON-CHIPS

4.1 Introduction

In addition to the online detection of faults in control part of NoCs, the localization of faults and also abstracting the fault information is of high value and must be performed with lowest possible latency. Especially, in case of NoC routers, such abstraction can be utilized in order to model faulty components of the routers or model fault in turns, which can be further used for re-configuration of the system. The system fault manager is in charge of this re-configuration, which has a holistic view of the healthy/faulty turns in the network.

This chapter covers the second contribution of this dissertation, which makes use of the information provided by the concurrent online checkers in the control part of a NoC router for fault localization and abstraction. The literature review regarding fault localization approaches in the control part of NoCs has already been covered in Chapter 2. In this chapter, first, it is explained how the information acquired from online checkers is interpreted and abstracted to meaningful data for higher levels in the system, such as the application layer. To this end, the checker outputs (acquired using the proposed methodology in Chapter 3) are fed to a fault localization module (developed by the author of this dissertation) in the Bonfire router **Architecture 3**, making it possible to find the location of faults in the control part of the router at different granularity levels, *i.e.* router-level, component-level and input/output port level (which is used for modelling *turn faults)*. Especially, the third level of granularity will be explained in detail, which is the contribution of this thesis and used by the system fault manager. However, it is worth noting that the implementation details of the system fault manager is not in the scope of this dissertation and the focus of this chapter is on the fault localization module and compression of fault information via abstraction. The contribution of this chapter has led to **publication E** [67].

4.2 Fault Localization and Fault Information Abstraction for Control Part of NoC Routers

Two of the main aspects of fault diagnosis in NoCs are fault detection and fault localization [7]. In this thesis, the former is performed via the concurrent online checkers, integrated at each control part module of a NoC router, whereas the latter is performed via a fully combinational logic integrated in the router to compress the fault information acquired from checkers and model turn faults in routers (introduced in this chapter).

The accuracy and granularity level of fault localization in NoC routers is important. Depending on the level of abstraction required by the system fault manager, the fault localization granularity can be adapted. It should be noted that this dissertation covers localization of faults and providing compressed information from the checker outputs, which would be transmitted to the system fault manager. However, the implementation details of the fault manager and how this information is transmitted, is out of the scope of this work.

As an example, all of the proposed mechanisms in this chapter are implemented in the Bonfire NoC router **Architecture 3** (introduced in Chapter 2). The fault localization

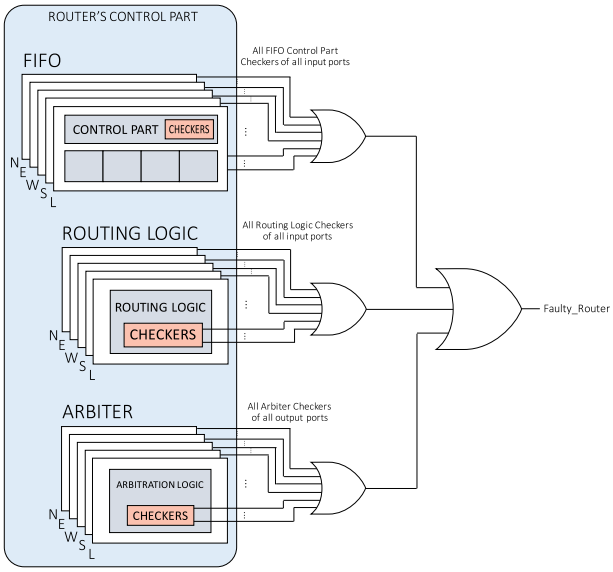


Figure 4.1 Router-level fault localization for control part of NoC router by means of concurrent online checker outputs.

and abstraction module has been developed and integrated within the router design by the author of this thesis.

The highest level of abstraction supported is router-level fault localization (**Figure 4**.1). This is achieved by ORing all the checker outputs for the control part components of the router. In such a case, regardless of the fault location in the control part of the router, the signal resulting from ORing all the checker outputs, indicates that the router is faulty. However, this level of coarse granularity suffers from low fault localization accuracy, since a fault in a single component results in the whole router rendered as faulty, whereas some intact parts of the router could have been usable.

The next level of fault localization granularity is router control part module-level fault localization (**Figure 4**.2). To this end, for each control part module (FIFO control part, routing computation unit and arbitration unit), the corresponding checker outputs for each module are ORed together and they form an error signal. This would help distinguish faults occurring in different modules, for instance, if the control part of FIFO for the North input port of a router becomes faulty, only the checkers corresponding to that module which are ORed together, will fire. The advantage of this level of fault localization granularity is that, for instance, by using resource-sharing based techniques (such as [84]), the faulty component can be isolated and the router can still function with remaining intact components, but at the price of gracefully degraded performance.

The third level of abstraction considered for localization of faults in the Bonfire NoC router **Architecture 3** takes into account the control part checker outputs in order to model *turn faults* (**Figure 4**.3), which is the contribution and focus of this chapter. NoC router **Architecture 3** has been chosen due to its higher performance compared to **Architecture 2** , because of using credit-based flow control (which is already explained in Chapter 2).

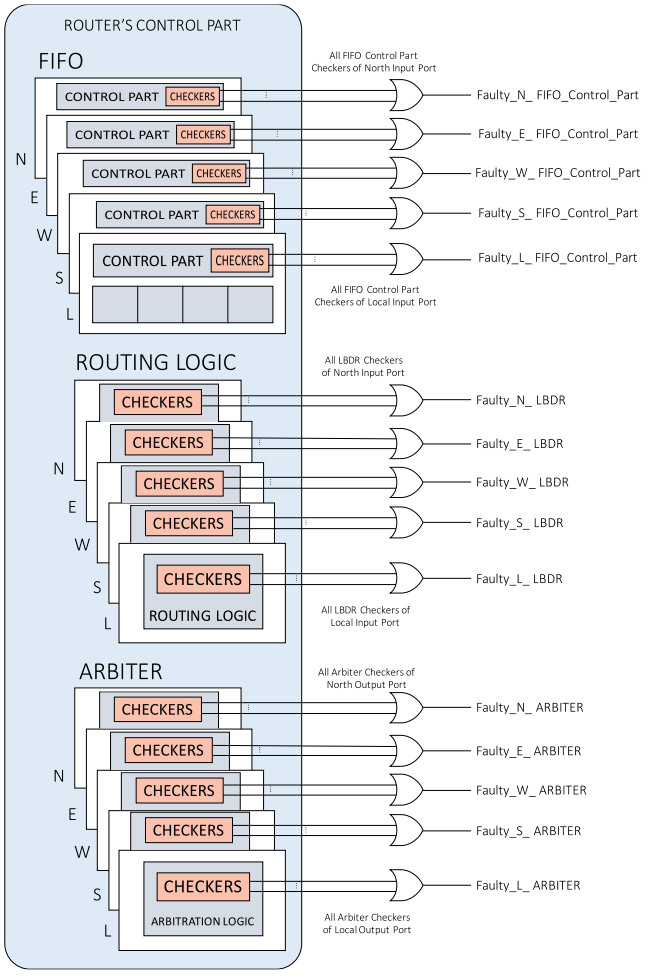


Figure 4.2 Component-level fault localization for control part of NoC router by means of concurrent online checker outputs.

A *turn fault* is specified as a fault present in a path from an input port to an output port. For instance, in a 2D Mesh-based NoC router, a West to North turn fault (shown as W2S turn fault) denotes the existence of a fault in either of the following modules: control part of FIFO of West input port, routing computation unit of West input port, or the arbitration logic related to West input and North output port. Of course, the fault can also be in a combination of these locations or in all of them.

In either case, the faulty scenario is interpreted as a West to North (W2N) turn fault. Such level of abstraction of checkers' fault information facilitates the process of reconfiguring the routing algorithm by the system fault manager. Especially, if LBDR is used to implement the routing logic (which is the case in all router architectures discussed in this dissertation), the set of allowed and disallowed turns shown in form of the routing bits can be re-configured by the system fault manager, using the information acquired from the turn faults at each router.

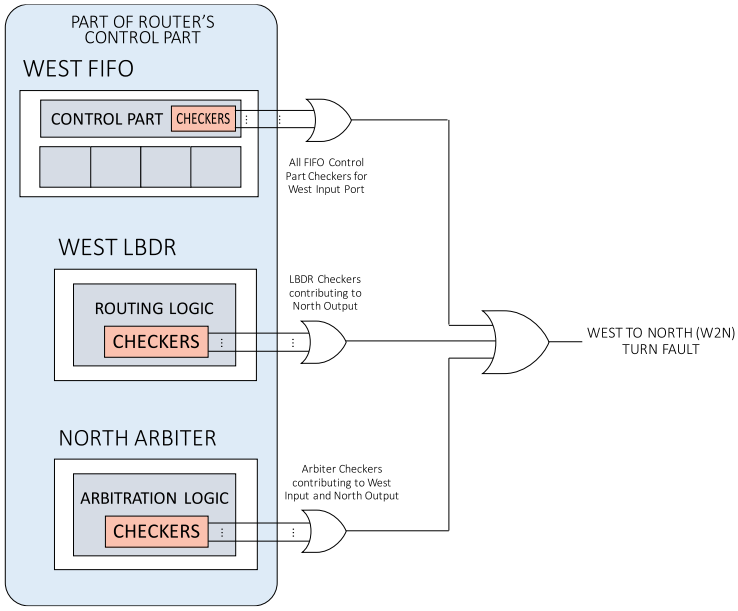


Figure 4.3 Combining concurrent online checker outputs and generating the abstracted Turn Fault (West-to-North (W2N) turn fault shown as an example).

In addition to the 8 turns that denote a 90 degree change of direction in the router (i.e. N2E, N2W, E2N, E2S, W2N, W2S, S2E and S2W), there are 4 straight paths (i.e. N2S, S2N, E2W and W2E), and 8 paths/turns related to the local (L) port of the router (4 starting from the local port to the other output ports and 4 starting from the other ports and leading to local port) (i.e. L2N, L2E, L2W, L2S, N2L, E2L, W2L and S2L), thus making in total 20 different turns in a router. Therefore, the fault localization module proposed in this dissertation, generates the values of these 20 turn faults based on the information acquired from the checkers.

Example: In order to clarify how a turn fault is modelled using the proposed fault localization module, the logic generating the *West to North (W2N) turn fault* in is explained in details. This example corresponds to the control part of Bonfire credit-based router. Recalling from **Table B. 1** in Appendix B, which shows all the concurrent online checkers devised for the control part Bonfire credit-based router, in order to model the W2N turn fault, the following checkers from each control module are taken into account in the fault localization unit (which is fully combinational):

- All checker outputs for the control part of **FIFO for the West input** port -(Checkers **1-110** from the table in Appendix B for W FIFO)- are ORed together, since the FIFO of West input port contributes to all turns deriving from the West input (including W2N turn).
- The next component that contributes to any turn stemmed from the West input would be the **LBDR (routing computation unit) for the West input**. However, for the case of West LBDR, only the checkers that check part of the logic related to North output request generation are considered and ORed together. The rest of the checkers are excluded from the logic for W2N as they do not contribute

to it (Checkers **1-7, 8, 9, 18-20, 21, 25-29, 145-154, 155-161** from the table in Appendix B for W LBDR are included).

- Finally, as the last control part module, the Allocator (arbitration) unit of the router is taken into account. The Allocator is composed of an internal logic which handles the credit counters and the flow control signals, plus 5 Arbiter_in modules for handling requests from inputs to multiple output directions (in case of using an adaptive routing algorithm), and 5 Arbiter_out modules which handle the arbitration for multiple requests for the same output port, giving grant to only one of the requests (as explained in Chapter 2). Since the focus is on W2N turn fault, thus, the fault localization unit should only consider the **Arbiter_in for West input** checkers and **Arbiter_out module for North output** checkers for modelling such turn fault. In addition, all the **Allocator internal logic checkers** that contribute to the W2N turn fault are also considered. Finally, all the considered checker outputs are ORed together (Checkers **5, 6, 51, 52, 61, 62-67** for Allocator internal logic and credit counter handling logic, Checkers **1, 2-5, 24, 25, 34, 35, 44, 45, 54, 55, 62-64, 65** for West Arbiter_in, and Checkers **1, 9-11, 20, 24, 25, 35, 40, 42-45, 46, 51** for North Arbiter_out, from the table in Appendix B).
- As the last step, all the checkers ORed from the previous steps are ORed together to create the final West to North (W2N) turn fault signal, which is one of the 20 turn faults information generated by the fault localization module. Similar deductions can be inferred to form the logic for localizing the remaining 19 turn faults in the router.

It is worth noting that in the implementation of the fault localization module in the Bonfire credit-based router, the third level of abstraction (modelling turn faults) has been chosen, however, the architecture supports all three above-mentioned levels of granularity for fault localization. The growing number of checkers for a complex design would make the fault detection information generated by the checkers quite large (in terms of the number of bits). This can, in turn, make it infeasible to transmit all fault information from the checkers to the system fault manager, which keeps a holistic view of the health status of the components of the network. This is one of the motivations behind introduction of the fault localization module in this dissertation, which would help reduce the total of more than 1000 control part checker outputs (more than 1000 bits) for Bonfire credit-based router to a final set of only 20 bits (representing 20 the turn faults).

Table 4.1 Area Overhead Analysis of the proposed Fault Localization Unit for modelling turn faults

	Baseline Router	Fault Localization Module	Fault-Tolerant Router
Area (μm^2)	92800	5314	193568
Area Overhead (%)	---	---	107.3 %
Critical Path Delay (ns)	7.69	2.42	7.82
Critical Path Delay Overhead (%)	---	---	1.69 %

The 20-bit turn faults obtained by the fault localization module could also be further classified based on their frequency of occurrence, as transient, intermittent and permanent, for instance using the approach proposed in [85]. This would also, in turn, help the system fault manager make decisions about which resources to use or not use when monitoring the health status of the system. In addition, how the classified fault information is propagated to the fault manager is of utmost importance, for instance, the main NoC can be used for this purpose (*e.g.* [67]), or a dual network could be used (*e.g.* [86]). The latter imposes more area overhead though. The author would like to emphasize that the details of both topics of fault classification and propagation of classified fault information to the system fault manager are out of the scope of this dissertation.

4.3 Hardware Overhead Analysis of Fault Localization Module for Modelling Turn Faults

As mentioned earlier, the proposed fault localization module with the capability of modelling turn faults is integrated in the Bonfire credit-based flow control NoC router. Of course, beforehand, using the proposed methodology in this dissertation, the full set of structural and functional checkers were devised for the control part of the router (comprised of FIFO control part, LBDR and Allocator). The checker outputs are fed to the fault localization module, integrated in the router. It is worth noting that similar to [24], only one fault localization module exists per router and takes into account the checker outputs from the current router and does not depend on the neighbour(s) (unlike [28]).

Table 4.1 has summarized the area overhead of the fault localization module with respect to the whole router. The area results are synthesized using AMS 0.18 µm CMOS technology library [87] and by means of Synopsys Design Compiler [88]. As it can be seen, the fault-tolerant router (Bonfire NoC router **Architecture 3)** with all the checkers, fault localization module and all fault-tolerance mechanism integrated, incurs 107.3% area overhead compared to the baseline non-fault-tolerant router. However, the fault-localization module only takes 2.76% of the area of the fault-tolerant router. This is less than the amount reported for the fault localization unit proposed for *NoCAlert* [24] (4.4% when considering the input/output port granularity level for localization with assertion vector compaction), while the proposed approach in this thesis not only covers faults in the control part modules related to input and output ports with single cycle latency, but it also performs the compression of the fault information and models the turn faults, that has not been addressed in the previous works. As mentioned earlier, such information can further be used by the system fault manager, in charge of computing a new routing algorithm to handle the faulty topology.

It is also worth noting that according to **Table 4**.1, the critical path delay of the fault localization module is 2.42 ns (with a constraint of clock period set as 3 ns in the Synthesis tool). The fault-tolerant router with all the fault detection and localization mechanisms incurs a critical path delay of about 1.69% compared to the baseline router (without any checkers and fault-tolerance mechanism).

4.4 Chapter Summary

This chapter covered the second contribution of this dissertation, which is proposing a fault localization module that takes into account the checker outputs and provides abstract and compressed fault information to be used by higher levels of abstraction (*e.g.* the application layer). The proposed mechanism has also been integrated into the Bonfire credit-based NoC router. Using the proposed fully combinational fault localization module, it was possible to compress more than 1000 checker outputs per router to a final meaningful set of only 20 bits, representing different *turn faults* in the router. The modelling of turn faults facilitates the process of routing re-configuration when a system fault manager deals with the faulty topology, taking into account the fault/health status of the routers. Synthesis results showed that the fault localization module only takes 2.76% of the fault-tolerant router with all the fault detection and localization mechanism integrated, which is still a lower amount compared to the state-of-the-art. The fault localization and abstraction approach proposed in this chapter as the second contribution of this dissertation has led to **publication E** [67].

5 LOGIC-BASED MECHANISM FOR IMPLEMENTATION OF FAULT-TOLERANT ROUTING IN 3D NETWORK-ON-CHIPS

5.1 Introduction

In an on-chip network, processing cores communicate with each other on one layer and they might also need access to their memory blocks at the same time, therefore one approach can be placing the memory blocks on an adjacent layer in a 3D NoC architecture. Different research works have focused on the topic of 3D integration of NoCs by using stacked layers [89]. As the number of vertical links is reduced in a 3D NoC - thus, transforming them into vertically partially connected 3D NoCs [51], [59] - the utilization of the remaining vertical links increases, therefore creating a communication bottleneck. These missing vertical links can be either the result of faults, such as wear-out, or they can be related to saving area due to the on-chip area constraints. Therefore, in order to run an application on such NoCs, a mechanism for implementing routing algorithms which would be both fault-tolerant and adaptive, would help mitigate the issue by uniformly distributing packets on the communication links and bypassing the faulty links, while being re-configurable at the same time. This has been the focus of the third contribution of this dissertation, explained in this chapter.

This chapter proposes a mechanism for implementing fault-tolerant routing algorithms in 3D Mesh-based Network-on-Chips with partially connected vertical links. The proposed mechanism removes the need for routing tables at routers, thus, making it a scalable solution for large network sizes. In addition, it does not rely on the location and number of faulty vertical links. Moreover, it does not augment the packets with any additional information overhead when transmitting them across the layers of the 3D NoC.

The literature review regarding the previously proposed fault-tolerant routing algorithms and mechanisms for 3D NoCs and also the background covering the pre-requisites for the baseline mechanism which the proposal of this chapter is based on, are all provided in Chapter 2. Therefore, the chapter starts with the description of the mechanism, named Logic-Based Distributed Routing for 3D NoCs (LBDR3D), which is an extension to LBDR, and follows with an example scenario to show how the mechanism handles routing in a 3D NoC with faulty vertical links. Afterwards, a summary of the experimental results is provided, emphasizing the scalability of the proposed approach. Finally, the chapter is concluded and a summary is provided, remarking the theoretical novelties. The contribution of this chapter led to **publication C** [90], included in the list of publications in Chapter 1.

5.2 LBDR3D Mechanism

One approach to address implementation of routing algorithms in Network-on-Chips (NoCs) is by means of routing tables. They make it possible to implement any routing algorithm for any type of topology [65]. However, they tend to grow with the increasing size of the network (number of nodes), thus, facing the challenge of scalability. On the contrary, implementing routing algorithms using a logical circuit distributed at each router in the network, can overcome this scalability issue. To this end, in [65], a logic-based approach named as LBDR was proposed which made it possible to implement any dead-lock free routing algorithm for NoCs with 2D Mesh topology and topologies derived

from the 2D Mesh. The mechanism basically describes the topology and routing algorithm using two fixed sets of configuration bits, called connectivity and routing bits. This brings the advantage of keeping the mechanism scalable, as it does not depend on the size of the network. Moreover, it provides the possibility of re-configuration - for instance, to address adaptation of the network to a situation with faulty links - by only modifying a few set of bits at each router.

LBDR3D is an extension to the previously introduced LBDR mechanism. The mechanism is inspired by the idea that for 3D NoCs with faulty vertical links, whenever a cross-layer communication is going needed, data should be transmitted one step closer to nodes with vertical links to eventually reach the corresponding destination layer and destination node. In contrast to previous approaches such as Elevator-first [51], LBDR3D does not need each router to store the location address of the nodes with vertical links, making it a more scalable solution. Instead, it utilizes a new set of bits, called the *vertical bits* (explained later in this chapter), which only indicate the existence of a node with vertical link. Moreover, in LBDR3D, the packet information is not augmented with any additional overhead, when transmitting data across the layers of the 3D NoC. More importantly, unlike approaches such as [59]–[61], [91], LBDR3D does not depend on the number and location of the faulty vertical links, and does not depend on the existence of any *pillars* in the network. With regards to faults on the horizontal links in each layer, LBDR3D supports the same number of faults as the baseline LBDR does (which is 2D Mesh topology in each layer and topologies derived from the 2D Mesh, as stated in [65]).

5.2.1 The Foundations for LBDR3D logic

The terminology "*the Foundations*" for the LBDR mechanism has been introduced in [92], which includes the configuration bits based on which LBDR would be able to implement the routing algorithm. The configuration bits include: routing bits and connectivity bits for the baseline LBDR mechanism. In the proposed LBDR3D mechanism, a new set of *vertical bits* is also added, which will be explained shortly.

As stated earlier, LBDR3D is an extension to LBDR [65]. In order to add support for 3D NoCs, the connectivity bits (C_x) of the logic are extended to cover *Up* and *Down* directions in the 3D domain, in addition to the existing directions for 4 cardinal 2D directions (North, East, West and South), therefore, leading to six connectivity bits per router, as follows:

$$C_x : C_n , C_e , C_w , C_s , C_u , C_d$$

LBDR3D uses the same number of routing bits (R_{xy}) as LBDR for implementing the routing algorithm in each layer, as follows:

$$R_{xy} : R_{ne} , R_{nw} , R_{en} , R_{es} , R_{wn} , R_{ws} , R_{se} , R_{sw}$$

One of the new additions to the mechanism is a new set consisting of 8 bits per router, named as *vertical bits*, based on which the logic can determine whether there is at least one node with up and/or down vertical link(s) in the corresponding direction or not (4 bits for up and 4 bits for down links). This reduces the area overhead compared to approaches such as Elevator-First [51], because the location address of the nodes with vertical links does not need to be stored at every router and only a fixed set of bits indicates the existence of at least one such node in a specific direction or quadrant with respect to each router. The vertical bits for LBDR3D are defined as follows:

$$N_u , E_u , W_u , S_u , N_d , E_d , W_d , S_d$$

The bits ending with *u* indicate that there is at least one vertical node with up link in the corresponding direction. The same applies to the bits ending with *d*, but for down links. In order to cover the situations in which the vertical node is located on a quadrant with respect to the current node, both the corresponding bits are set. For instance, if a router has a node on the North-East quadrant with the up vertical link, both N_u and E_u bits at the current router are set.

One important issue is the approach taken to compute the values of the vertical bits at each router, which is addressed in the sub-section 5.2.3 of this chapter. This is performed via the proposed offline algorithm that calculates the vertical bits at each router at the same time when connectivity and routing bits are initialized. The re-configuration process of these bits is performed using the OSR-Lite framework [8] in a transparent way, without imposing significant run-time latency and affecting normal operation of the network. Details regarding the re-configuration process are however, out of the scope of this dissertation.

5.2.2 LBDR3D Logic Description

The logic of LBDR3D is proposed based on the principle that packets should be steered towards a node with vertical link when having cross-layer traffic, making the packet getting closer to its destination eventually, but it should not wander between different nodes with vertical links in one layer, since in that case, it can lead to live-lock and affect performance. Also, the underlying routing algorithm in each layer of the 3D NoC must be deadlock-free for the mechanism to guarantee deadlock freeness. The complete logic of

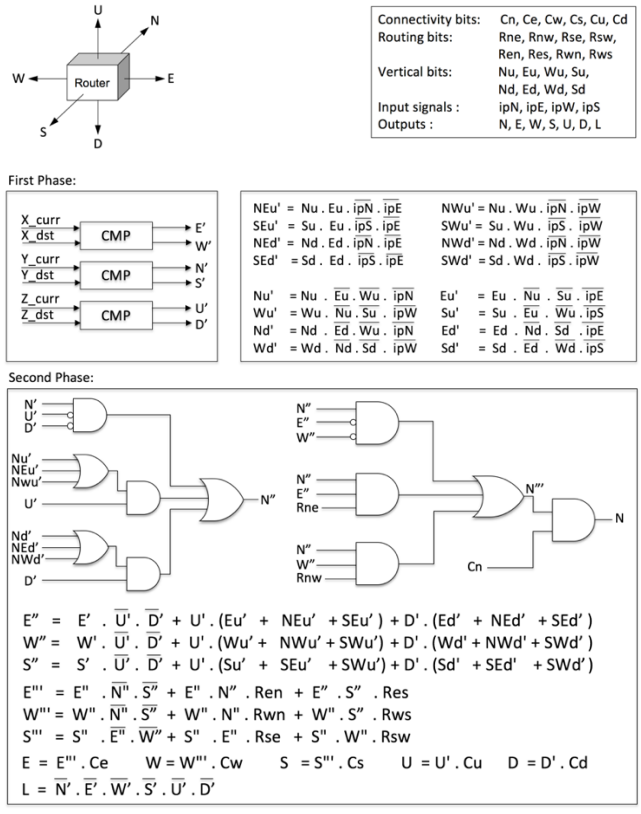


Figure 5.1 Proposed logic of LBDR3D mechanism

LBDR3D mechanism is shown in **Figure 5**.*1* [90] (**publication C**).

In the first phase, the direction signals are computed by comparing the current address of the packet (stored in the current router) and the destination address of the packet (extracted from the header flit of the packet), *i.e.* signals *N', E', W', S', U'* and *D'* are computed. Also, in this phase, first the quadrants or directions that the packet cannot traverse are filtered out temporarily for the packet.

In order to prevent a packet from fluctuating between two vertical nodes (which guarantees live-lock freeness), four additional signals have been introduced and utilized which are fed from the 2D input ports, as follows:

ipX : ipN, ipE, ipW, ipS

7For instance, if a packet comes from the North input port, *ipN* signal is set to one. As the packet should not go back to the North direction again (avoiding U-turns), it must not be possible for the packet to be steered towards North (N) direction in search of a vertical link.

Next, the directions that the packet may take, are computed, that means the packet is transmitted on the plane using any kind of deadlock-free turn model routing algorithm that can already be implemented using LBDR on a 2D NoC. In order to explain the logic of LBDR3D, the focus is on one output port, for instance the North (N) output port logic.

For the North port to be selected for forwarding the packet, one of the following conditions must hold: (1) The packet's destination is located on the same layer as the current node and it is located towards the North direction (the term ***N'. U'. D'*** in **Figure 5**.*1*), or (2) The current node is not a vertical node, but there exists at least one up/down vertical node on the same layer as the current node towards the North direction (*i.e.* on North direction or on North-East or North-West quadrant) (the term ***U'. (N_u' + NE_u' + NW_u') + D'. (Nd' + NE_d' +NW_d')*** in **Figure 5**.*1* [90]).

In the second phase of the logic, for instance, in case of the North output logic, if one of the above-mentioned conditions hold, the North output port can be selected if either (1) the destination is located on the same column as the current node in the North direction or (2) it is located on the North-East (NE) or (3) North-West (NW) quadrant and the turn at the next router along North direction allows the packet to take the North to East (R_{ne} = 1) or North to West turn (R_{nw} = 1), respectively. Finally, for the North port (N) to be considered as the output port for transmitting the packet, the corresponding connectivity bit of North port (C_n) should also be set to one. Therefore, in the end, the packet will be forwarded to the North output port (if North is also chosen by the arbitration unit) and it will either reach its final destination (if destination is on the same layer as current node) or it will reach the nearest node with up/down vertical link, depending on whether it needs to go upwards or downwards (when destination is not on the same layer as current node).

Similar logic can be deduced for the E, W and S output ports. The output port signals that have slightly different logics are U (*Up*) and D (*Down*) and the L (*Local*) output port signals. If a packet reaches a vertical node and has to be steered upwards or downwards, only U or D output port can become active, respectively, and other output port signals are automatically set to zero (based on the logic's behaviour and because the offline algorithm will compute all the corresponding vertical bits as zero for a node with vertical link, as will be explained later in this chapter). It should be noted that depending on the

topology, and based on the nearest vertical node, the vertical bits for a node might change during the life-time of the system, if re-configuration would be necessary.

Also, regarding the *Local* output port (L), it is activated only when the packet has reached its destination (all the direction signals *N', E', W', S', U'* and *D'* are zero). In such case, since the current address of the router is the same as the destination address of the packet, the flits of the packet are forwarded to the Processing Element (PE) connected to the router's *Local* port.

It is worth noting that in order to avoid the occurrence of deadlock when cross-layer traffic transmission is performed by LBDR3D, two Virtual Channels (VCs) are used per router, which separate the traffic going upwards from the one going downwards. The VC of a packet is chosen at the source node, based on either the destination is on a higher or lower layer. If the destination is on the same layer as the source, one of the VCs is chosen randomly. It is worth noting that once a packet is injected into the network, it can never change its VC, as otherwise, it would introduce possibility of deadlock.

5.2.3 Offline Algorithm for Computation of Vertical Bits

The next contribution of this chapter is the offline algorithm introduced for calculation of vertical bits (shown in Algorithm 5.1 [90]) based on which LBDR3D performs routing decisions, including cross-layer transmission of packets in the 3D NoC.

For each router (node), first, it is checked whether it is a vertical node itself (lines **5-6** and **14-15** of Algorithm 5.1). In that case, all the corresponding vertical bits are set to zero (if the node is an up vertical node, all the 4 up vertical bits are set to zero, and similarly the same approach is done for down vertical nodes). If the node is not a vertical node, the node in the same layer with the shortest *Manhattan distance* to the current node (explained in Algorithm 5.1) that has a vertical link is searched and based on the location of that node, the corresponding vertical bits are set in the current node. If two nodes exist with the same Manhattan distance from the current node, one is chosen *randomly* to break the tie. The procedure is once performed for calculation of up vertical bits (lines **8-13** of Algorithm 5.1 [90]) and the other time for the calculate of down vertical bits (lines **17-22** of Algorithm 5.1 [90]). The outputs of the algorithm are the final set of vertical bits for all routers of the network, which would serve as part of "*the Foundations*" for LBDR3D and they are fed in a transparent manner via the OSR-Lite [86], [93], [94] reconfiguration mechanism to the LBDR3D logic at system start-up. The mechanism guarantees negligible re-configuration latency and deadlock freeness for the system, when changing from routing algorithm to another.

Moreover, as proven in [90], as long as the routing algorithm in each layer of the 3D NoC is dead-lock free and also faults do not disconnect the network nodes completely from each other, LBDR3D guarantees deadlock-freeness, live-lock freeness and connectivity. Details regarding proof of deadlock and live-lock freeness and connectivity of LBDR3D are published in **publication C** [90].

Example: To further clarify the computation of the set of vertical bits for LBDR3D using the offline algorithm, an example scenario with a 4×4×4 3D Mesh-based NoC with 88% faulty vertical links (as shown in **Figure 5**.2 [90]) is explained in the following.

Algorithm 5.1: Offline algorithm for calculating vertical bits at routers in each layer

```
1  forall the nodes router[curr] ∈ NoC do
2      DVL = ∅ // Down Vertical Nodes List
3      UVL = ∅ // Up Vertical Nodes List
4      HTV = 0 // hops to Vertical node
5      if Cu[router[curr]] = 1 then
6          SetVerticalBits("U", router[curr], 0)
7      else
8          forall the router[i] ∈ Layer[curr] do
9              if router[i] ≠ router[curr] then
                   // for each router[i] in current
                      layer except the current node
10                 if Cu [router[i]] = 1 then
                       // router[i] is an up vertical
                          node
11                     HTV = Distance (router[i],router[curr])
12                     append HTV to UVL
13         SetVerticalBits("U",router[curr],FindMin(UVL))
14     if Cd[router[curr]] = 1 then
15         SetVerticalBits("D", router[curr], 0)
16     else
17         forall the router[i] ∈ Layer[curr] do
18             if router[i] ≠ router[curr] then
                   // for each router[i] in current
                      layer except the current node
19                 if Cd [router[i]] = 1 then
                       // router[i] is a down vertical
                          node
20                     HTV = Distance (router[i],router[curr])
21                     append HTV to DVL
22         SetVerticalBits("D",router[curr],FindMin(DVL))

// Functions Description:
// 1. Distance (router[i],router[curr]) returns
      |Δx + Δy|= |x_curr − x_router[i]|+ |y_curr − y_router[i]|
// 2. FindMin(VerticalNodesSet) returns the
      vertical node in set VerticalNodesSet with
      shortest distance to current node,ties are
      broken randomly
// 3. SetVerticalBits(Dir,router[curr],VNode)
      sets the vertical bits of router[curr] in
      direction Dir("U"=up,"D"=down), to vertical
      node VNode
```

In the scenario shown in **Figure 5**.2 [90], if node **53** wants to send a packet to node **37**, it has 3 choices for choosing a node on the current layer as an up vertical node (nodes **51**, **60** and **63**). A vertical node is defined as a node with vertical link. As it can be seen in **Figure 5**.2, it cannot be necessarily guaranteed that the total path the packet takes to reach its destination will be the

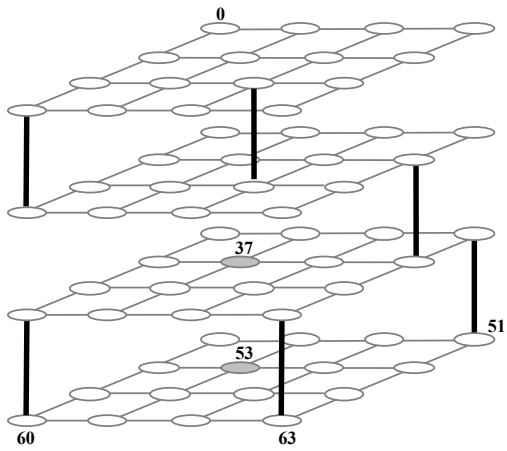


Figure 5.2 A 4×4×4 3D Mesh-based NoC with 88% faulty vertical links

minimal path. Instead of trying to take the minimal possible path from source to destination, the offline algorithm calculates the values of vertical bits at each router based on its *Manhattan distance* to a vertical node (as shown in Algorithm 5.1). According to **Figure 5**.2, two nodes with shortest Manhattan distance of 3 with respect to node **53** can be chosen as candidates as up vertical nodes (*i.e.* nodes 51 and 60). In this case, the tie is broken by randomly choosing one of the possible candidates, for instance, node **60** is chosen. Therefore, since node **60** is located on the South-West quadrant of node **53**, the values of up vertical bits at node **53** will be set as follows by the offline algorithm:

$N_u = 0$, $E_u = 0$, $W_u = 1$, $S_u = 1$

Also, since node **53** is located at the bottom-most layer of the 3D NoC of Figure 5.2, all the down vertical bits for this node are set to zero, as follows:

$N_u = 0$, $E_u = 0$, $W_u = 0$, $S_u = 0$

The configuration bits of LBDR3D are calculated offline and fed to the logic at system start-up. Thereafter, the algorithm for computation of vertical bits will only be executed if a new fault occurs in the network and there is a need for re-configuration of the vertical bits.

5.2.4 Example Scenario of Fault-Tolerant Routing Using LBDR3D

The functionality of LBDR3D logic is shown with an example scenario, demonstrated in **Figure 5**.3.

The source of the communication is node **35** and the destination is node 2. In such scenario, the destination node is on a different layer than the source node, therefore, the part of LBDR3D logic in charge of transmitting the packet to the node with vertical links and the values of pre-computed vertical bits also plays an important role in routing. The routing algorithm in each layer is considered to be the North-Last deadlock-free turn model [95] (as shown in **Figure 5**.3), which provides partial adaptivity. It is assumed that the offline algorithm has already been applied to the faulty

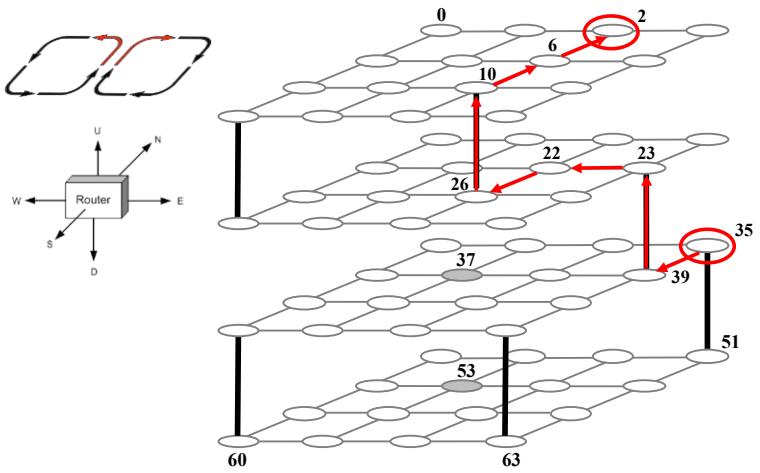


Figure 5.3 An example scenario of packet routing using a fault-tolerant routing algorithm implemented with LBDR3D

topology of **Figure 5**.3 and the values of the vertical bits are sets at all nodes. Therefore, at the source node (node **35**) the vertical bits would be as follows:

N_u = 0, E_u = 0, W_u = 0, S_u = 1

N_d = 0, E_d = 0, W_d = 0, S_d = 0

This would mean that the up vertical node with the shortest possible Manhattan distance with respect to node **35** is located on the South direction of it. Since node **35** is a down vertical node itself, all the corresponding vertical bits related to the down direction are set to zero at this node. However, since the flit must be sent upwards to reach its destination, the down vertical bits do not play a role in this routing procedure. According to the logic of LBDR3D, at this step, the flit is forwarded to node **39** on the South direction of node **35**. Node **39** is an up vertical node itself, therefore, all the corresponding up vertical bits are set to zero at that node and LBDR3D gives the priority to up direction. Thus, the flit is forwarded to node 23 in the upper layer. At node **23**, the values of the up vertical bits are set as follows:

N_u = 0, E_u = 0, W_u = 1, S_u = 1

This indicates that there exists at least one node with up vertical link on the South-West quadrant of node **23**. According to the North-Last turn model, both West and South output ports can be taken. It is assumed that the routing logic gives the priority to the West output. Thus, the flit is forwarded to node **22** (as shown in **Figure 5**.3 with the path shown by red arrows).

At node 22, the value of S_u vertical bit is set to 1 and the other vertical bits for up direction are set to zero. Even though there exists both nodes 28 on South-West quadrant and node 26 on South direction of node 22 for sending the flit upwards, the priority is given to node 26 by the offline algorithm. The reason is that the *Manhattan Distance* of node 26 with respect to node **22** is shorter. Thus, the flit is forwarded to the South direction, reaching node **26**. Node **26** is an up vertical node, which would forward the flit directly upwards to node **10**. Currently, the flit is in its destination layer.

It should also be noted that since in this scenario the packet is going only upwards to reach its destination, it is assigned to one of the VCs and therefore, it can never change its VC. The existence of two VCs per each input port would guarantee that cross-layer traffic transmission would not lead to deadlock during routing packets by LBDR.

The rest of the routing path would be the same as the way LBDR mechanism would make decisions for routing in 2D NoCs. Since the destination node (node **2**) is on the same column as node **10**, the flit is forwarded to North output port to node **6** and finally to node **2** and it reaches its destination (as shown in **Figure 5**.3 with the red arrows). As this example shows, LBDR3D is able to route the flit to its destination despite the faulty topology with 88% faulty vertical links. As long as faults do not disconnect the network, LBDR3D guarantees the connectivity between all source-destination pairs (more detailed information is provided in **publication C** [90]).

5.3 Summary of Experimental Results

This sub-section is dedicated to the experimental results, first comparing the proposed LBDR3D mechanism with state-of-the-art (the approaches [51], [59], [61] from the ones reviewed in Chapter 2) in terms of performance (average packet latency). Afterwards, the area consumption of LBDR3D are compared with the other fault-tolerant mechanisms for 3D NoCs, showing the scalability of the proposed mechanism.

5.3.1 Performance Analysis

In [90], LBDR3D is compared with other state-of-the-art approaches proposed for fault-tolerant routing in 3D NoCs with partially faulty vertical links. However, as LBDR3D does not rely on the existence of a pillar in the network and does not rely on the number and

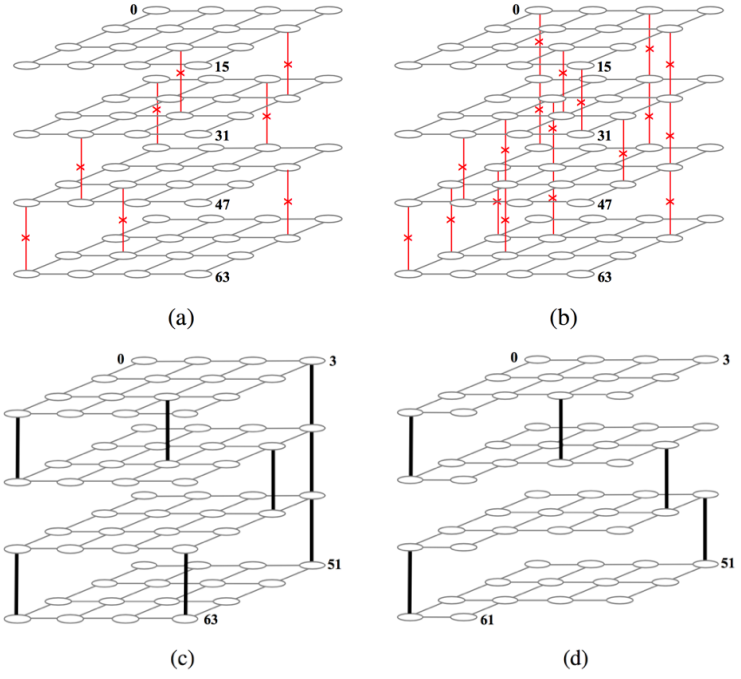


Figure 5.4 Different considered scenarios: with (a) 20%, (b) 40%, (c) 84% faulty vertical links, and (d) 88% faulty vertical links with some faulty horizontal links

location of faulty vertical links, experimental results are only considered for LBDR3D and Elevator-First [51]. The other two mechanisms, named NETZ [61] and ETW [59] depend on existence of a pillar in the network and for instance, they do not support some of the topologies shown in **Figure 5**.4, thus, they are only considered in the area overhead comparison experiments.

The scenarios in which adaptive routing algorithms have been implemented using LBDR3D mechanism and compared against Elevator-First, are shown in **Figure 5**.4 [90], covering from scenarios with 20% (**Figure 5**.4a), 40% (**Figure 5**.4b), and 84% (**Figure 5**.4c) faulty vertical links and a scenario with 88% (**Figure 5**.4d) faulty vertical links and some faulty horizontal links. It should be noted that in **Figure 5**.4a and **Figure 5**.4b, the red vertical links are the faulty and the vertical links that are not shown are the healthy ones. Whereas, in figure **Figure 5**.4c and **Figure 5**.4d, only the healthy vertical links are shown for the sake of figure's simplicity and the faulty ones are not shown.

As reported in [90], the experiments have been performed by an extension of the open-source NoC simulator, Noxim [96], [97] with 3D NoC support. The parameters used in the experiments and the traffic scenarios considered (synthetic traffic patterns) for simulations are summarized in **Table 5**.1 [90].

The performance (average packet latency) results for different simulation scenarios are detailed in [90]. One of the observations that has been made in different fault scenarios (ranging from 20% to 88% faulty vertical links), LBDR3D performs similar or slightly better than Elevator-First when programmed to XY routing and also turn-model based adaptive routing. Even thought, performance results might be similar, the advantages of LBDR3D over Elevator-First are two-fold: (1) no extra information is added to the packet when transmitting data from one layer to another through a node with vertical link, and (2) There is no need to store the location address of the node(s) with vertical link (up and/or down) at any node, and instead only the fixed set of 8 vertical bits are set (calculated using the offline algorithm proposed in this dissertation) per each router, making LBDR3D scalable, especially in large network sizes (beyond 25×25×25). Moreover, in [51], the routing algorithm in each layer of the 3D NoCs has only been considered as the deterministic XY (X-First as stated in [51]) routing. Whereas, in this dissertation and in [90], an adaptive routing algorithm is used (such as the well-known West-First and North-Last [95] turn models).

In addition, a scenario with 88% faulty vertical links and some horizontal faulty links is considered (as shown in **Figure 5**.4d), which is still supported by LBDR3D when using West-First turn-model routing in each layer of the 3D NoC.

Table 5.1 Considered Scenarios and simulation parameters

Routing Algorithm	LBDR3D XY, Elevator-First X-First (XY), LBDR3D West-First, LBDR3D North-Last
Network Topology	4×4×4 3D Mesh with 20%,40%, 84% faulty vertical links and 88% faulty vertical links with some faulty horizontal links
Number of VCs	2 (per each router input port)
VC depth	4 flits
Network Frequency	1 GHz
Simulation Time	10000 cycles (1 cycle = 1ns)
Warm-up time	Warm-up time 1000 cycles (1 cycle = 1ns)
Traffic patterns	Random Uniform, Bit-Reversal and Transpose

5.3.2 Area Consumption and scalability Analysis

In the experiments, the area consumption of LBDR3D has been compared with other fault-tolerant routing mechanisms for partially vertically connected 3D NoCs, *i.e.* Elevator- First, NETZ and ETW. To this end, the RTL logic of LBDR3D for a 4×4×4 and 10×10×10 3D Mesh, along with the logic of ZXY routing (which is a non-fault-tolerant mechanism), Elevator- First, NETZ and ETW are described in Verilog, and synthesized using Synopsys Design Compiler[3] [88]. The results showed 34.6% increase in area for a 4×4×4 3D Mesh and 11.7% increment in area for a 10×10×10 3D Mesh, when comparing LBDR3D logic to Elevator-First using XY routing. The decrease in the area overhead can be a proof of the scalability of LBDR3D as it does not store the location of nodes with vertical links in each layer. In addition, when comparing LBDR3D programmed to XY routing (LBDR3D XY) with NETZ and ETW for the case of a 4 × 4 × 4 3D Mesh network, the area overhead was only around 5.02% and 5.1%, respectively. Another explanation for the area overhead of LBDR3D compared to ZXY, LBDR, Elevator-First, NETZ and ETW would be the additional set of vertical bits and the new logic for supporting 3D NoC topologies, but at the same time it brings the advantage of providing flexibility and not relying on existence of any pillars in the topology.

Figure 5.5 and Figure 5.6 [90] summarize the area consumption results for the compared mechanisms both for a 4×4×4 and a 10×10×10 3D Mesh-based NoC. In addition, the area of LBDR3D has been compared with ZXY routing, which is the one of the simplest non-fault-tolerant routing algorithm for 3D NoCs and also with its 2D

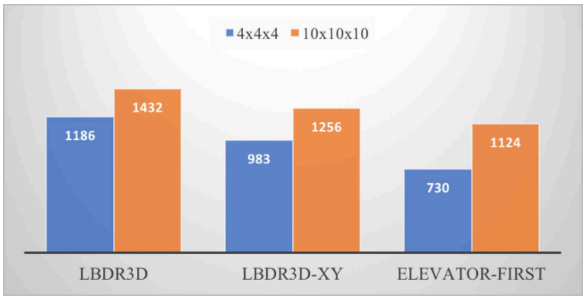


Figure 5.5 Area consumption (in µm^2) for different compared routing mechanisms for showing scalability of LBDR3D over Elevator-First.

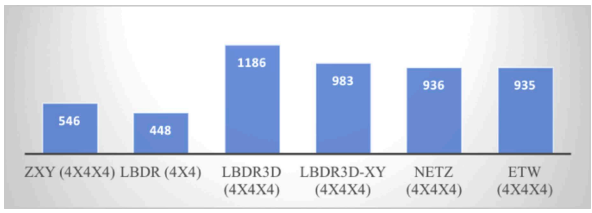


Figure 5.6 Area consumption (in µm^2) for different compared routing mechanisms for comparison of LBDR3D to ZXY, original LBDR, NETZ and ETW.

[3] Synopsys Design Compiler: http://www.synopsys.com

counterpart, the original LBDR mechanism. The area results are obtained using NanGate Open Cell 45 nm Library[4] [98] and synthesis of the RTL of the designs is performed using Synopsys Design Compiler [88].

5.4 Chapter Summary

This chapter covered the third contribution of this dissertation, focusing on implementation of fault-tolerant routing in 3D Mesh-based NoCs with partially faulty vertical links. The proposed mechanism is a logic-based technique that makes it possible to implement any deadlock-free turn model based routing algorithm in such topologies. The mechanism is scalable and depends only on a fixed number of configuration bits, *i.e.* connectivity, routing and vertical bits. The vertical bits are used to define the existence of at least one node with a vertical link in a specific direction for each router, thus removing the need for storing the location address of the node with vertical link at every router. Moreover, using only two Virtual Channels (VC), the proposed approach guarantees deadlock freeness for cross-layer communication. It also guarantees live-lock freeness and connectivity, as long as faults do not disconnect the network.

What also makes LBDR3D different from the previous works is that it does not rely on the existence of a pillar in the network and does not rely on the location and number of faulty vertical links. Moreover, it does not augment packets with additional information when being sent across layers of the 3D NoC. Performance and area overhead results showed the advantages of the proposed mechanism, reaching similar or better average packet latency compared to state-of-the-art, and being scalable for large network sizes. The proposed mechanism in this chapter as the third contribution of this dissertation has led to **publication C** [90], included in the list of publications in Chapter 1.

[4] NanGate 45nm Open Cell Library: http://www.nangate.com/?page id=2325

CONCLUSIONS

The trend in moving from computation-centric to communication-centric systems and the integration of more processing elements on the same chip, has increased the value of Network-on-Chips (NoCs) as a scalable interconnection paradigm. However, the miniaturization of semi-conductor technologies beyond the sub-micron domain jeopardizes the reliability of on-chip components, including NoC routers. Transient and permanent faults can cause serious problems such as mis-routing of packets, corruption of data and eventually deadlock and breakdown of the whole system. This dissertation focused on (1) online detection of faults in control part of NoCs, while providing a trade-off between fault coverage and area overhead, (2) localization of faults in NoC routers and abstraction of fault information, and (3) a generic scalable approach for implementing fault-tolerant routing algorithms in NoCs.

The first contribution of this dissertation was proposing a methodology for devising concurrent online checkers for online detection of faults in control part of a circuit. As an example, the proposed methodology was applied to the control part of a NoC router. The proposed methodology allows extracting two sets of checkers, *i.e.* structural and functional. Moreover, the methodology provides automated evaluation of the checkers and minimization in terms of area overhead, while meeting the target fault coverage. The checkers guarantee single cycle fault detection latency for Single Event Upsets (SEUs) in the control part of NoC routers. Moreover, the proposed methodology is capable of formally proving the existence or absence of True Misses when evaluating the checkers under the exhaustive set of valid input stimuli. The additional area for the checkers imposed to the circuit under check comes with the advantage of providing fault localization capability, as opposed to approaches such as Duplication With Comparison (DMR) and TMR.

In general, the number of functional checkers might be less than structural ones, but they can still cover a larger part of the circuit. However, they cannot always guarantee reaching 100% coverage for SEUs. On the other hand, structural checkers will always guarantee reaching 100% coverage of SEUs, but at the price of duplicating every part of the circuit, as each part of the RTL code is checked for the occurrence of SEUs. The advantage of structural checkers compared to functional ones would be remarkable when high fault localization accuracy is of utmost importance, as they can pinpoint in which part of the logic the fault has occurred, whereas the functional checkers are more abstract. The minimization part of the proposed methodology provides a final trade-off between reaching the target coverage while still meeting the area budget provided by the user. However, in case the highest level of fault localization accuracy is required, it would be recommended to consider the full set of structural checkers, whereas where area constraints are stringent and fault localization accuracy is not a major issue, having the final minimized set of checkers (including a combination of structural and functional checkers) would also be acceptable, provided by the proposed methodology. Therefore, at a first glance, no specific final decision can be made whether only structural or functional checkers are sufficient and depending on the verification engineer's goals and available area budget, they can be chosen accordingly.

The second contribution of this dissertation was addressing the problem of big data acquired by the checker outputs when the design is complex. To this end, a fault localization and abstraction module was introduced. This unit allows abstracting fault

information, which compresses and translates the fault data to meaningful information (i.e. turn faults) for higher layers of abstraction (such as application layer). As an example of integrating the fault localization module in the control part of Bonfire router, a final set of more than 1000 checkers were compressed to a fixed set of 20 bits, denoting 20 turn faults in the router. Such compact information can be further used by the global fault manager in the system in charge of keeping a holistic view of the health status of the components, also facilitating the routing algorithm re-configuration process.

The fault localization latency is also as important as fault detection latency. As opposed to the state-of-the-art which captures 97% of the stuck-at faults during the first cycle, the proposed checkers and fault localization module in this thesis are able to detect and localize 100% of the SEUs in maximum one clock cycle of their occurrence. Thus, the abstracted turn faults (in form of the compressed 20 bits) are also provided in the same clock cycle as fault localization, which can be further utilized by the system fault manager to take actions upon re-configuration of the system. As the fault localization module is fully combinational, it can contribute to the increase of the critical path delay of the circuit, and one might argue that one solution to this would be to store all the checker outputs in a set of Flip-Flops. However, this would impose a significant area overhead to the system, as storing more than 1000 checker outputs in Flip-Flops would translate into memory elements when synthesized. Therefore, in this dissertation, it has been chosen to keep the fault localization and abstraction module fully combinational, and not storing any of the checker outputs in a memory-based element. With regards to the information provided by the abstraction of checker outputs to a final set of 20 turn faults, one step that can be taken as future research would be to classify the occurrence of these turn faults in terms of their frequency (as transient, intermittent or permanent). However, the approach used to implement the classification logic must be designed with care, as, for example, using counter-threshold based approaches can introduce additional area overheads to the system.

The third contribution of this thesis addresses the problem of implementing fault-tolerant routing algorithms in NoCs. As a solution, a scalable and re-configurable mechanism (LBDR3D) was proposed. The mechanism allows implementation of any dead-lock free turn model-based routing algorithm in 3D NoCs with faulty vertical links. The proposed mechanism removes the need of routing tables at all in the routers. Also, it guarantees live-lock and deadlock freeness and connectivity both in case of intra- and inter-layer traffic, while it also does not depend on the location and number of faulty vertical links. In order to codify the topology, routing algorithm and location of nodes with vertical links, a fixed set of configuration bits are implemented. This would guarantee scalability of the mechanism for larger network sizes, and also remove the need to store any location address of nodes with vertical link at every router, and also avoids incurring overhead to the transmitted packet information.

The motivation behind LBDR3D was to provide a scalable and flexible solution for implementing different routing algorithms, however, this might not necessarily mean that the performance of LBDR3D is always better than all other approaches proposed in the literature for partially vertically connected 3D NoCs. For instance, performance-wise (in terms of average packet latency), ETW performs better than LBDR3D (programmed to turn model-based routing) under most traffic patterns. Despite this, the remarkable feature of LBDR3D becomes noticeable when the focus is on generality of the mechanism, *i.e.* not being dependent on the location of faulty vertical links. For ETW and

NETZ, they depend on existence of vertical links in some locations in the network or require the existence of a pillar, respectively. However, in reality, faults can occur on any of the links in the network. Both Elevator-First and LBDR3D take this into account and regardless of the location of the faulty link, as long as faults do not disconnect the network layers completely, they both provide a valid path for every source-destination pair. With regards to area overhead, LBDR3D would be more compact than Elevator-First, as it does not need to store the location address of nodes with vertical links at each router in the network. This has been codified instead (in LBDR3D) by just a few sets of bits, showing the existence of such nodes (with vertical link). The same advantage exists for LBDR3D when transmitting packets between layers, as there is no need to store the address of intermediate nodes with vertical link in the header, whereas Elevator-First imposes such packet information overhead.

As future work, the following works can be pursued:

- With regards to the first and second contributions of this thesis, the re-action to the faults and how the system could recover based on the detected faults and their location can be further explored, for instance augmenting the checkers with correction capability as well. Also, taking into account the classification of faults based on the frequency of this occurrence would be important. Furthermore, considering the security aspects in addition to fault detection when devising checkers for the digital circuit would be of research value.
- Regarding the third contribution of this dissertation, extending the mechanism to support any possible combination of faults on the horizontal links in each layer of the 3D NoC (in addition to the faults in the vertical links) would be important. Also, adding support for irregular topologies derived from 3D Mesh-based NoCs is a feature that can be further explored and added to the proposed mechanism for implementing routing algorithms in such topologies.

Abbreviations

BICST	Built-In Concurrent Self-Test
BIST	Built-In Self-Test
CEI	Checkers' Efficiency Index
CMP	Chip Multi-Processor
CTS	Clear To Send
DCTS	Detect Clear To Send
DQP	Dynamic Quadrant Partitioning
DRTS	Detect Request To Send
DWC	Duplication With Comparison
EM	Electro-Migration
ETW	East-Then-West
FC	Fault Coverage
FIFO	First In First Out
FIFO	First-In-First-Out
FPR	False Positive Ratio
FSM	Finite State Machine
HBH	Hop-By-Hop
IIR	Inherent Information Redundancy
LBDR	Logic-Based Distributed Routing
NETZ	North-East To Z
NI	Network Interface
NoC	Network-on-Chip
PE	Processing Element
ROWR	Reduced Observation Width Replication
RR	Round-Robin
RTL	Register Transfer Level
RTS	Request To Send
SEU	Single Event Upset
SMC	Secure Model Checker
SoC	System-on-Chip
SR	Segment-based Routing
SSBDD	Structurally Synthesized Binary Decision Diagram

TMR	Triple Modular Redundancy
TSV	Through-Silicon Via
VC	Virtual Channel
VCT	Virtual Cut-Through

List of Figures

List of Tables

References

[1] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.

[2] L. Benini and G. De Micheli, 'Networks on Chips: A New SoC Paradigm', *Computer (Long. Beach. Calif).*, vol. 35, no. 1, pp. 70–78, Jan. 2002.

[3] P. Saltarelli, B. Niazmand, R. Hariharan, J. Raik, G. Jervan, and T. Hollstein, 'Automated minimization of concurrent online checkers for Network-on-Chips', in *2015 10th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, 2015, pp. 1–8.

[4] M. Gössel, *New methods of concurrent checking*, vol. 42. Dordrecht: Springer Netherlands, 2008.

[5] L. Benini and D. Bertozzi, 'Network-on-chip architectures and design methods', *IEE Proc. - Comput. Digit. Tech.*, vol. 152, no. 2, p. 261, 2005.

[6] G. De Micheli and L. Benini, 'Networks on Chips: 15 Years Later', *Computer (Long. Beach. Calif).*, vol. 50, no. 5, pp. 10–11, 2017.

[7] A. Dalirsani, 'Self-diagnosis in Network-on-Chips', Jan. 2015.

[8] M. Gössel, V. Ocheretny, E. Sogomonyan, and D. Marienfeld, *New methods of concurrent checking*, vol. 42. Dordrecht: Springer Netherlands, 2008.

[9] S. Werner, J. Navaridas, and M. Luján, 'A Survey on Design Approaches to Circumvent Permanent Faults in Networks-on-Chip', *ACM Comput. Surv.*, vol. 48, no. 4, pp. 1–36, 2016.

[10] E. Dubrova, *Fault-Tolerant Design*. Springer Publishing Company, Incorporated, 2013.

[11] P. K. Samudrala, J. Ramos, and S. Katkoori, 'Selective triple Modular redundancy (STMR) based single-event upset (SEU) tolerant synthesis for FPGAs', *IEEE Trans. Nucl. Sci.*, vol. 51, no. 5, pp. 2957–2969, Oct. 2004.

[12] J. M. Berger, 'A note on error detection codes for asymmetric channels', *Inf. Control*, vol. 4, no. 1, pp. 68–73, 1961.

[13] D. Das and N. A. Touba, 'Synthesis of Circuits with Low-Cost Concurrent Error Detection Based on Bose-Lin Codes', *J. Electron. Test.*, vol. 15, no. 1, pp. 145–155, 1999.

[14] K. Mohanram, E. S. Sogomonyan, M. Gossel, and N. A. Touba, 'Synthesis of low-cost parity-based partially self-checking circuits', in *9th IEEE On-Line Testing Symposium, 2003. IOLTS 2003.*, 2003, pp. 35–40.

[15] S. Ghosh, S. Basu, and N. A. Touba, 'Synthesis of Low Power CED Circuits Based on Parity Codes', in *23rd IEEE VLSI Test Symposium (VTS'05)*, 2005, pp. 315–320.

[16] R. Sharma and K. K. Saluja, 'An implementation and analysis of a concurrent built-in self-test technique', in *[1988] The Eighteenth International Symposium on Fault-Tolerant Computing. Digest of Papers*, 1988, pp. 164–169.

[17] P. Drineas and Y. Makris, 'Concurrent fault detection in random combinational logic', in *Fourth International Symposium on Quality Electronic Design, 2003. Proceedings.*, 2003, pp. 425–430.

[18] K. Nepal, N. Alves, J. Dworak, and R. I. Bahar, 'Using Implications for Online Error Detection', *2008 IEEE International Test Conference*. pp. 1–10, 2008.

[19] N. Alves, Y. Shi, J. Dworak, R. I. Bahar, and K. Nepal, 'Enhancing online error detection through area-efficient multi-site implications', *29th VLSI Test Symposium*. pp. 241–246, 2011.

[20] M. Boule, J. S. Chenard, and Z. Zilic, 'Assertion Checkers in Verification, Silicon Debug and In-Field Diagnosis', *8th International Symposium on Quality Electronic Design (ISQED'07)*. pp. 613–620, 2007.

[21] C. Grecu, A. Ivanov, R. Saleh, E. S. Sogomonyan, and P. P. Pande, 'On-line Fault Detection and Location for NoC Interconnects', in *12th IEEE International On-Line Testing Symposium (IOLTS'06)*, 2006, pp. 145–150.

[22] D. Park, C. Nicopoulos, J. Kim, N. Vijaykrishnan, and C. R. Das, 'Exploring fault-tolerant network-on-chip architectures', in *Proceedings of the International Conference on Dependable Systems and Networks*, 2006, vol. 2006, pp. 93–104.

[23] A. Prodromou, A. Panteli, C. Nicopoulos, and Y. Sazeides, 'NoCAlert: An On-Line and Real-Time Fault Detection Mechanism for Network-on-Chip Architectures', *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*. pp. 60–71, 2012.

[24] K. Chrysanthou *et al.*, 'An Online and Real-Time Fault Detection and Localization Mechanism for Network-on-Chip Architectures', *ACM Trans. Archit. Code Optim.*, vol. 13, no. 2, pp. 1–26, Jun. 2016.

[25] R. Parikh and V. Bertacco, 'ForEVeR: A Complementary Formal and Runtime Verification Approach to Correct NoC Functionality', *ACM Trans. Embed. Comput. Syst.*, vol. 13, no. 3s, p. 104:1--104:30, Mar. 2014.

[26] Q. Yu, M. Zhang, and P. Ampadu, 'Exploiting inherent information redundancy to manage transient errors in NoC routing arbitration', *Proceedings of the Fifth ACM/IEEE International Symposium*. pp. 105–112, 2011.

[27] Q. Yu, J. Cano, J. Flich, and P. Ampadu, 'Transient and Permanent Error Control for High-End Multiprocessor Systems-on-Chip', *2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip*. pp. 169–176, 2012.

[28] L. Huang, X. Zhang, M. Ebrahimi, and G. Li, 'Tolerating transient illegal turn faults in NoCs', *Microprocess. Microsyst.*, vol. 43, pp. 104–115, Jun. 2016.

[29] R. Abdel-Khalek, R. Parikh, A. DeOrio, and V. Bertacco, 'Functional correctness for CMP interconnects', *2011 IEEE 29th International Conference on Computer Design (ICCD)*. pp. 352–359, 2011.

[30] B. Niazmand, R. Hariharan, V. Govind, G. Jervan, T. Hollstein, and J. Raik, 'Extended checkers for Logic-Based Distributed Routing in Network-on-Chips', *2014 14th Biennial Baltic Electronic Conference (BEC)*. pp. 77–80, 2014.

[31] A. Alaghi, N. Karimi, M. Sedghi, and Z. Navabi, 'Online NoC Switch Fault Detection and Diagnosis Using a High Level Fault Model', *22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT 2007)*. pp. 21–29, 2007.

[32] T. Boraten, D. DiTomaso, and A. K. Kodi, 'Secure Model Checkers for Network-on-Chip (NoC) Architectures', *Proc. 26th Ed. Gt. Lakes Symp. VLSI - GLSVLSI '16*, pp. 45–50, 2016.

[33] G. Dimitrakopoulos and E. Kalligeros, 'Low-cost fault-tolerant switch allocator for network-on-chip routers', in *Proceedings of the 2012 Interconnection Network Architecture on On-Chip, Multi-Chip Workshop - INA-OCMC '12*, 2012, pp. 25–28.

[34] A. Strano, C. Gómez, D. Ludovici, M. Favalli, M. E. Gómez, and D. Bertozzi, 'Exploiting Network-on-Chip structural redundancy for a cooperative and scalable built-in self-test architecture', in *2011 Design, Automation & Test in Europe*, 2011, pp. 1–6.

[35] K. Petersén and J. Öberg, 'Utilizing NoC Switches as BIST Structures in 2D-Mesh

Network-on-Chips', 2006.
[36] K. Petersen and J. Oberg, 'Toward a Scalable Test Methodology for 2D-mesh Network-on-Chips', in *2007 Design, Automation & Test in Europe Conference & Exhibition*, 2007, pp. 1–6.
[37] M. Hosseinabady, A. Dalirsani, and Z. Navabi, 'Using the inter- and intra-switch regularity in NoC switch testing', in *Proceedings -Design, Automation and Test in Europe, DATE*, 2007, pp. 361–366.
[38] A. Pellegrini and V. Bertacco, 'Cardio: CMP Adaptation for Reliability Through Dynamic Introspective Operation', *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 33, no. 2, pp. 265–278, Feb. 2014.
[39] G. Schley, A. Dalirsani, M. Eggenberger, N. Hatami, H.-J. J. Wunderlich, and M. Radetzki, 'Multi-Layer Diagnosis for Fault-Tolerant Networks-on-Chip', *IEEE Trans. Comput.*, vol. 66, no. 5, pp. 848–861, May 2017.
[40] Y. Jojima and M. Fukushi, 'A fault-tolerant routing method for 2D-mesh Network-on-Chips based on components of a router', in *2016 IEEE 5th Global Conference on Consumer Electronics*, 2016, pp. 1–2.
[41] B. Aghaei, A. Khademzadeh, M. Reshadi, and K. Badie, 'Link Testing: a Survey of Current Trends in Network on Chip', *J. Electron. Test.*, vol. 33, no. 2, pp. 209–225, Apr. 2017.
[42] M. Radetzki, C. Feng, X. Zhao, and A. Jantsch, 'Methods for fault tolerance in networks-on-chip', *ACM Comput. Surv.*, vol. 46, no. 1, pp. 1–38, 2013.
[43] D. Fick, A. DeOrio, J. Hu, V. Bertacco, D. Blaauw, and D. Sylvester, 'Vicis: A reliable network for unreliable silicon', *2009 46th ACM/IEEE Design Automation Conference*. pp. 812–817, 2009.
[44] M. R. Kakoee, V. Bertacco, and L. Benini, 'A distributed and topology-agnostic approach for on-line NoC testing', in *Proceedings of the Fifth ACM/IEEE International Symposium on Networks-on-Chip - NOCS '11*, 2011, p. 113.
[45] M. R. Kakoee, V. Bertacco, and L. Benini, 'At-Speed Distributed Functional Testing to Detect Logic and Delay Faults in NoCs', *IEEE Trans. Comput.*, vol. 63, no. 3, pp. 703–717, Mar. 2014.
[46] E. A. Rambo, C. Seitz, S. Saidi, and R. Ernst, 'Designing Networks-on-Chip for High Assurance Real-Time Systems', *2017 IEEE 22nd Pacific Rim Int. Symp. Dependable Comput.*, pp. 185–194, Jan. 2017.
[47] P. Bahrebar, A. Jalalvand, and D. Stroobandt, 'Dynamically Reconfigurable Architecture for Fault-Tolerant 2D Networks-on-Chip', in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, 2017, pp. 1–7.
[48] M. Ebrahimi, M. Daneshtalab, J. Plosila, and H. Tenhunen, 'Minimal-path fault-tolerant approach using connection-retaining structure in networks-on-chip', in *2013 7th IEEE/ACM International Symposium on Networks-on-Chip, NoCS 2013*, 2013, pp. 1–8.
[49] A. Ghofrani, R. Parikh, S. Shamshiri, A. DeOrio, K.-T. Cheng, and V. Bertacco, 'Comprehensive online defect diagnosis in on-chip networks', in *2012 IEEE 30th VLSI Test Symposium (VTS)*, 2012, pp. 44–49.
[50] C. Killian, C. Tanougast, F. Monteiro, and A. Dandache, 'Smart reliable network-on-chip', *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 22, no. 2, pp. 242–255, Feb. 2014.
[51] F. Dubois, A. Sheibanyrad, F. Petrot, and M. Bahmani, 'Elevator-First: A Deadlock-Free Distributed Routing Algorithm for Vertically Partially Connected 3D-NoCs',

IEEE Trans. Comput., vol. 62, no. 3, pp. 609–615, Mar. 2013.

[52] S. Pasricha and Y. Zou, 'A low overhead fault tolerant routing scheme for 3D Networks-on-Chip', in *2011 12th International Symposium on Quality Electronic Design*, 2011, pp. 1–8.

[53] C. Feng, M. Zhang, J. Li, J. Jiang, Z. Lu, and A. Jantsch, 'A Low-Overhead Fault-Aware Deflection Routing Algorithm for 3D Network-on-Chip', in *2011 IEEE Computer Society Annual Symposium on VLSI*, 2011, pp. 19–24.

[54] S. Akbari, A. Shafiee, M. Fathy, and R. Berangi, 'AFRA: A low cost high performance reliable routing for 3D mesh NoCs', in *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2012, pp. 332–337.

[55] M. Ebrahimi, M. Daneshtalab, and J. Plosila, 'Fault-tolerant routing algorithm for 3D NoC using hamiltonian path strategy', *2013 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. pp. 1601–1604, 2013.

[56] X. Jiang and T. Watanabe, 'A novel fully adaptive fault-tolerant routing algorithm for 3D Network-on-Chip', in *2013 IEEE International Conference of IEEE Region 10 (TENCON 2013)*, 2013, pp. 1–4.

[57] J. Zhou, H. Li, Y. Fang, T. Wang, Y. Cheng, and X. Li, 'HARS: A High-Performance Reliable Routing Scheme for 3D NoCs', in *2014 IEEE Computer Society Annual Symposium on VLSI*, 2014, pp. 392–397.

[58] P. Mitra, 'TARAS: A topology agnostic routing algorithm using segmentation strategy for 3D NoC', in *Proceeding - IEEE International Conference on Computing, Communication and Automation, ICCCA 2016*, 2017, pp. 1606–1611.

[59] R. Salamat, M. Ebrahimi, and N. Bagherzadeh, 'An Adaptive, Low Restrictive and Fault Resilient Routing Algorithm for 3D Network-on-Chip', in *2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, 2015, pp. 392–395.

[60] R. Salamat, M. Khayambashi, M. Ebrahimi, and N. Bagherzadeh, 'A Resilient Routing Algorithm with Formal Reliability Analysis for Partially Connected 3D-NoCs', *IEEE Transactions on Computers*, vol. 65, no. 11. pp. 3265–3279, 2016.

[61] H. Ying, K. Hofmann, and T. Hollstein, 'Dynamic quadrant partitioning adaptive routing algorithm for irregular reduced vertical link density topology 3-Dimensional Network-on-Chips', in *2014 International Conference on High Performance Computing & Simulation (HPCS)*, 2014, pp. 516–522.

[62] E. Taheri, M. Isakov, A. Patooghy, and M. A. Kinsy, 'Advertiser elevator: A fault tolerant routing algorithm for partially connected 3D Network-on-Chips', in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2017, pp. 136–139.

[63] P. Mitra, 'LBDR3D: Fault tolerant routing scheme for 3D NoCs', in *2015 Annual IEEE India Conference (INDICON)*, 2015, pp. 1–6.

[64] A. Prodromou, A. Panteli, C. Nicopoulos, and Y. Sazeides, 'NoCAlert: An On-Line and Real-Time Fault Detection Mechanism for Network-on-Chip Architectures', *2012 45th Annu. IEEE/ACM Int. Symp. Microarchitecture*, vol. 13, no. 2, pp. 60–71, 2012.

[65] S. Rodrigo, S. Medardoni, J. Flich, D. Bertozzi, and J. Duato, 'Efficient implementation of distributed routing algorithms for NoCs', *IET Comput. Digit. Tech.*, vol. 3, no. 5, p. 460, 2009.

[66] 'Project Bonfire Network-on-Chip'. 2015.

[67] S. P. Azad, *B. Niazmand, K.* Janson, N. George, A. S. Oyeniran, T. Putkaradze, A.

Kaur, J. Raik, G. Jervan, R. Ubar, and T. Hollstein, 'From online fault detection to fault management in Network-on-Chips: A ground-up approach', in *2017 IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, 2017, pp. 48–53.

[68] S. Rodrigo, J. Flich, J. Duatc, and M. Hummel, 'Efficient unicast and multicast support for cmps', in *Proceedings of the Annual International Symposium on Microarchitecture, MICRO*, 2008, no. 2008 PROCEEDINGS, pp. 364–375.

[69] J. Duato, S. Yalamanchili, and L. M. Ni, *Interconnection Networks: An Engineering Approach*. Morgan Kaufmann, 2003.

[70] S. Rodrigo *et al.*, 'Addressing manufacturing challenges with cost-efficient fault tolerant routing', in *NOCS 2010 - The 4th ACM/IEEE International Symposium on Networks-on-Chip*, 2010, pp. 25–32.

[71] S. Rodrigo *et al.*, 'Cost-efficient on-chip routing implementations for CMP and MPSoC systems', in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2011, vol. 30, no. 4, pp. 534–547.

[72] R. Bishnoi, V. Laxmi, M. S. Gaur, and J. Flich, 'd^2-LBDR: Distance-driven routing to handle permanent failures in 2D mesh NoCs', *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. pp. 800–805, 2015.

[73] J. Cano, J. Flich, A. Roca, J. Duato, M. Coppola, and R. Locatelli, 'Efficient routing in heterogeneous SoC designs with small implementation overhead', *IEEE Trans. Comput.*, vol. 63, no. 3, pp. 557–569, Mar. 2014.

[74] B. Niazmand, R. Hariharan, V. Govind, G. Jervan, T. Hollstein, and J. Raik, 'Extended checkers for Logic-Based Distributed Routing in Network-on-Chips', in *Proceedings of the Biennial Baltic Electronics Conference, BEC*, 2014, vol. 2015–Novem.

[75] N. Gupta, M. Kumar, V. Laxmi, M. S. Gaur, and M. Zwolinski, 'σLBDR: Congestion-aware logic based distributed routing for 2D NoC', in *2015 19th International Symposium on VLSI Design and Test*, 2015, pp. 1–6.

[76] N. Gupta, A. Sharma, V. Laxmi, M. S. Gaur, M. Zwolinski, and R. Bishnoi, 'σ^nLBDR: generic congestion handling routing implementation for two-dimensional mesh network-on-chip', *IET Computers & Digital Techniques*, vol. 10, no. 5. pp. 226–232, 2016.

[77] E. A. Rambo, C. Seitz, S. Saidi, and R. Ernst, 'Bridging the Gap between Resilient Networks-on-Chip and Real-Time Systems', *IEEE Trans. Emerg. Top. Comput.*, vol. VV, no. SEPTEMBER, 2017.

[78] T. Lehtonen, D. Wolpert, P. Liljeberg, J. Plosila, and P. Ampadu, 'Self-adaptive system for addressing permanent errors in on-chip interconnects', *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 18, no. 4, pp. 527–540, Apr. 2010.

[79] A. I. Jutman *et al.*, 'Turbo Tester - Diagnostic Package for Research and Training', *Tallinn Univ. Technol. Dep. Comput. Eng. Raja 15, Tallinn, Est. Turbo Tester*, vol. 3, pp. 69–73.

[80] R. Ubar, S. Devadze, J. Raik, and A. Jutman, 'Ultra fast parallel fault analysis on structurally synthesized BDDs', in *Proceedings - 12th IEEE European Test Symposium, ETS 2007*, 2007, pp. 131–136.

[81] C. Draft *et al.*, 'A Comprehensive Reliability Assessment of Fault-Resilient Network-on-Chip Using Analytical Model', *IEEE Trans. Very Large Scale Integr. Syst.*, vol. PP, no. 99, pp. 1–14, 2017.

[82] P. Saltarelli *et al.*, 'A framework for combining concurrent checking and on-line

embedded test for low-latency fault detection in NoC routers', in *Proceedings - 2015 9th IEEE/ACM International Symposium on Networks-on-Chip, NOCS 2015*, 2015.

[83] S. P. S. P. Azad, B. Niazmand, A. K. A. K. Sandhu, J. Raik, G. Jervan, and T. Hollstein, 'Automated area and coverage optimization of minimal latency checkers', in *2017 22nd IEEE European Test Symposium (ETS)*, 2017, pp. 1–2.

[84] T. Putkaradze, S. P. Azad, B. Niazmand, J. Raik, and G. Jervan, 'Fault-resilient NoC router with transparent resource allocation', in *12th International Symposium on Reconfigurable Communication-Centric Systems-on-Chip, ReCoSoC 2017 - Proceedings*, 2017, pp. 1–8.

[85] J. Silveira, C. C. C. Marcon, P. Cortez, G. Barroso, J. M. J. M. Ferreira, and R. Mota, 'Scenario preprocessing approach for the reconfiguration of fault-tolerant NoC-based MPSoCs', *Microprocess. Microsyst.*, vol. 40, pp. 137–153, Feb. 2016.

[86] A. Strano, D. Bertozzi, F. Trivino, J. L. Sanchez, F. J. Alfaro, and J. Flich, 'OSR-Lite: Fast and deadlock-free NoC reconfiguration framework', in *2012 International Conference on Embedded Computer Systems (SAMOS)*, 2012, pp. 86–95.

[87] 'AMS 0.18um CMOS process'. 2016.

[88] 'Synopsys Design Compiler'. 2015.

[89] J. Flich and D. Bertozzi, *Designing network on-chip architectures in the nanoscale era*. Chapman and Hall/CRC, 2011.

[90] B. Niazmand, S. P. Azad, J. Flich, J. Raik, G. Jervan, and T. Hollstein, 'Logic-based implementation of fault-tolerant routing in 3D network-on-chips', in *2016 10th IEEE/ACM International Symposium on Networks-on-Chip, NOCS 2016*, 2016.

[91] A. Charif, N.-E. Zergainoh, A. Coelho, and M. Nicolaidis, 'Rout3D: A lightweight adaptive routing algorithm for tolerating faulty vertical links in 3D-NoCs', in *2017 22nd IEEE European Test Symposium (ETS)*, 2017, pp. 1–6.

[92] S. R. Mocholi, 'Cost Effective Routing Implementations for On-Chip Networks', Universidad Politécnica de Valencia, 2010.

[93] M. Balboni, J. Flich, and D. Bertozzi, 'Synergistic use of multiple on-chip networks for ultra-low latency and scalable distributed routing reconfiguration', *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015*. IEEE Conference Publications, New Jersey, pp. 806–811, 2015.

[94] M. Balboni and F. Trivi, 'Optimizing the Overhead for Network-on-Chip Routing Reconfiguration in Parallel Multi-Core Platforms', *2013 Int. Symp. Syst. Chip*, pp. 1–6, Oct. 2013.

[95] C. J. Glass and L. M. Ni, 'The Turn Model for Adaptive Routing', in *[1992] Proceedings the 19th Annual International Symposium on Computer Architecture*, 1992, vol. pages, no. 7, pp. 278–287.

[96] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti, 'Cycle-Accurate Network on Chip Simulation with Noxim', *ACM Trans. Model. Comput. Simul.*, vol. 27, no. 1, pp. 1–25, 2016.

[97] 'Noxim with LBDR3D support'. .

[98] 'NanGate, Inc. NanGate 45nm Open Cell Library'. .

Acknowledgements

Firstly, I would like to express my gratitude to my supervisor Professor Gert Jervan and my co-supervisor Professor Jaan Raik, for their tremendous support regarding my PhD studies, research, publications and motivation for the work we invested to focus on during the last 4 years. Without their guidance and enlightening the research path, writing this dissertation and the publications would not be possible. Working with my supervisor and co-supervisor has also taught me the lesson that during PhD studies, one cannot always have prejudice on one topic or specific field to work on, but better to actually have a wider and more open-minded view on solving problems and thinking of solutions to a problem with different angles.

I would also like to thank the thesis reviewing committee at Tallinn University of Technology for their constructive and informative feedback and comments on the dissertation. Moreover, I would like to appreciate the pursuing of Ms. Katri Kadakas and Ms. Anu Johannes for pursing all the administrative and paperwork during my PhD studies, related to Dean's office and initial registration at university.

I would see it necessary to appreciate Prof. José Flich, who was a co-author of one of the publications which is part of the contribution to this dissertation. His guidance and help with this research path has also given me clear ideas regarding future works in the fields relevant to this dissertation.

Also, I would like to thank my colleagues and friends, whom without their help, many of the publications and collaborative works we had would not be possible, especially Siavoosh Payandeh Azad, Karl Janson, Hannes Kinks, Pietro Saltarelli and Ranganathan Hariharan. Also, I would like to especially thank my other friends and relatives who encouraged me to pursue my studies at PhD level and kept me motivated constantly.

Last but not certainly least, I would like to express my deep gratitude to my family, my mother and my aunt who have been there for me since the first day I started my education. Their constant support during the years made it possible for me to make progress in my education and be able to seek the path I was very motivated in, *i.e.* Computer Engineering.

Finally, I would also like to give my acknowledgment to Tallinn University of Technology for accepting me as a PhD student and providing the facilities and environment to experience collaborative research, work with academic tools and software. I would also like to acknowledge that my research work would have not been possible without the financial supports from (1) EU's H2020 RIA IMMORTAL, (2) Estonian institutional research grant IUT 19-1, (3) the Estonian Center of Excellence in IT EXCITE funded by the European Regional Development Fund, (4) Estonian IT Academy program and EU's Twinning Action TUTORIAL project.

Lühikokkuvõte
Töökindluse parandamine kiipvõrkudel põhinevatel süsteemides

Pooljuhttehnoloogia mõõtmete vähenedes integreeritakse ühele kiibile üha rohkem arvutustuumasid, mistõttu muutub kiipsüsteemide jõudluse kitsaskohaks tuumadevaheline ühendustaristu. Harilikud, siinipõhised ühendused ei suuda tuumade arvu kasvades piisavat jõudlust pakkuda. Nende puuduste lahendamiseks on alternatiivse kiipsüsteemi ühendustaristuna välja pakutud kiipvõrgud.

Paraku mõjutab praegune suundumus kahandada transistoride mõõtmeid pooljuhttehnoloogial põhinevate seadmete, kaasaarvatud kiipvõrkude töökindlust. Kuigi püsivad rikked on tihtipeale võimalik avastada tehases toote testimise käigus, tuleb ikkagi tegeleda normaalse kulumise ja vananemise tulemusena tekkinud rikete ja süsteemi eluajal esinevate mööduvate vigadega. Seetõttu on vaja lähenemist, mis suudaks ilma tööd katkestamata vigu tuvastada ja vajadusel neile võimalikult kiiresti reageerida. Üks kirjanduses välja pakutud lahendusest kiipsüsteemimarsruuterite juhtosas tööajal avalduvate vigade tuvastamiseks on süsteemiga paralleelselt töötavad rikkemonitorid. Juhul, kui sellised rikkemonitorid ei ole piisavalt hästi disainitud, on nende peamine puudus suur pindala kiibil. See omakorda tekitab vajaduse luua metoodika, mida saaks kasutada kiipsüsteemide juhtosale selliste rikkemonitoride loomiseks, mis suudaks võimalikult väikese kiibipindala juures tagada soovitud veakatvusprotsendi. Selle väitekirja esimene panus ongi sellise metoodika väljatöötamine. Rikkemonitoride hindamiseks ja minimeerimismetoodika analüüsiks korraldatud eksperimentide tulemuste analüüs näitab, et minimeeritud komplekt väljatöötatud kiipsüsteemimarsruuteri juhtosa rikkemonitoridest garanteerib sajaprotsendilise üksikute konstantsete rikete katvuse, tagades samal ajal peaaegu kohest vigade tuvastamist. Rikkemonitoride arvelt disainile lisanduv kiibipindala ei ületa pindala, mis oleks vaja kolmekordse liiasuse (TMR) saavutamiseks.

Disaini keerukuse kasvades suureneb rikkemonitoride arv märgatavalt, mistõttu kasvab süsteemi rikkehaldurile saadetav andmete kogus liiga suureks. Süsteemi rikkehaldur on eraldi moodul, mis võib olla implementeeritud nii tarkvaras, riistvaras kui ka kombinatsioonina neist kahest. Veahaldur omab informatsiooni kiipvõrgu komponentide nagu marsruuterite, nende vaheliste ühenduste ja ka marsruuterites paiknevate pöörete veaoleku kohta. Rikkehaldur saab seda informatsiooni kasutada rikkehaldusmehanismide tarbeks, näiteks marsruutimisalgoritmi ümberseadistamiseks komponendi- või pöördevea tuvastamise korral. Selleks et rikke tuvastamiseks kasutatava teabe üldistustase vastaks veahalduris kasutatava teabe üldistustasemele, on vaja tehnikat rikkemonitoridelt saadava informatsiooni üldistamiseks ja tihendamiseks. Lisaks vea tuvastamisele on oluline ka vea asukoha kindlaks määramine, sest see võimaldab tagada süsteemi töö rikete korral, kasutades vaid töötavaid komponente ja minnes mööda rikkis komponentidest. Seetõttu ongi selle väitekirja teiseks panuseks rikete lokaliseerimiseks ja veainformatsiooni üldistamiseks kasutatava mehanismi väljatöötamine. Väljatöötatud rikete lokaliseerimis- ja veainformatsiooni üldistamismooduli sünteesitulemused näitavad, et võrreldes teiste moodsate lahendustega, kasutab väljatöötatud moodul vähem kiibiala, garanteerides samas madala hilistumise.

Kuna võrgusõlmedevaheliste ühenduste rikked kiipvõrkudes võivad mõjutada kogu võrgu jõudlust, on oluline käsitleda ka marsruutimisalgoritmi mõjutavaid võrgukihirikkeid. Lisaks peab veakindla marsruutimisalgoritmi implementeerimiseks kasutatav mehhanism olema skaleeritav ja taasseadistatav. Samuti peab see olema paindlik, et rikkehalduril oleks võimalik marsruutimisalgoritmi vajadusel muuta. Eelmainitud mehhanism ei tohi sõltuda vigaste võrgusõlmede asukohast ega arvust. Selleks on selle väitekirja kolmanda panusena välja töötatud loogikapõhine jaotatud marsruutimismehhanism, mis on võrreldes marsruutimistabelitega palju skaleeritavam, kuna selle pindala kiibil ei sõltu kiipvõrgu sõlmede arvust. Selle mehhanismi töö sõltub ainult fikseeritud seadistusbittidest (mille väärtused arvutatakse selles väitekirjas toodud algoritmiga süsteemi töö väliselt), omades samas nii kahe- kui ka kolmemõõtmeliste kiipvõrkude tuge ning garanteerides tupikude ja nõiaringide puudumise kiipvõrgus. Väljapakutud mehhanismiga tehtud katsete tulemused tõestavad selle skaleerimisvõimekust võrreldes teiste tänapäevaste alternatiividega, mõjutamata tuntavalt kiipvõrkude jõudlust. Seetõttu on see mehhanism suurte kiipvõrkude puhul soositud lahendus.

Abstract
Dependability Improvements of NoC-based Systems

The trend in shrinking size of semiconductor technology beyond the sub-micron domain, and the need for integrating more Processing Elements (PEs) on the same chip would render the underlying interconnection infrastructure as a bottle-neck. For instance, the traditional shared-medium bus-based architecture cannot catch up with the growing number of Intellectual Property (IP) cores, due to performance and scalability limitations. Network-on-Chip (NoC) has emerged as an interconnection infrastructure paradigm to address the parallelism and performance limitation of conventional bus-based architectures [5], and handle communication-centric Systems-on-Chips (SoCs) with large number of communicating PEs.

Unfortunately, the current trend in miniaturization of transistors, affects the reliability of devices based on semiconductor technology, including NoCs. Even if permanent faults are captured using manufacturing testing, the circuits being susceptible to run-time faults (caused by phenomena such as wear-out and aging) and transient faults during system's lifetime must still be addressed. There is a need for an online approach, which would instantaneously detect faults at run-time, concurrent with the system operation and would react rapidly to them. Concurrent online checkers have been one of the approaches introduced in the literature for handling run-time faults online in control part of NoCs. However, the area overhead of the fault detection circuitry would become a concern if not envisioned properly. This necessitates a methodology for devising checkers for the control part of NoCs, while addressing both, fault detection quality of the checkers and minimization of checkers in terms of area, while guaranteeing the target fault coverage. Proposing such methodology has been the focus of the first contribution of this dissertation. Experimental results for checkers' evaluation and minimization methodology show that the minimized set of the devised structural and functional checkers guarantee 100% single stuck-at fault coverage in the control part modules of a NoC router, while providing near-instantaneous (single-cycle) fault detection latency and formal proof of presence/absence of True Misses, and in worst case, an area overhead between duplication and triplication-based approaches, such as Duplication With Comparison (DWC) and Triple Modular Redundancy (TMR).

As the designs become more complex, the number of concurrent online checkers tends to grow significantly, therefore, when transmitting the fault information to the system fault manager, it would generate excessive amount of data. The system fault manager is a module in the system, which can be a separate block implemented in hardware or software or combination of both, having information of fault/health status of network components (including routers, links and turns in the routers). The system fault manager can make use of the detected fault information in order to re-configure the routing algorithm, for instance in case of a fault in a component or a turn fault. In order to match the abstraction of fault detection information to the information used by the system fault manager, a technique would be needed to make such information compact and compressed. Moreover, in addition to detection of faults, finding the location of the fault is important, for instance, in order to make use of the healthy parts of the device, while bypassing the faulty component(s). Therefore, to achieve the best of both worlds, a fault localization and abstraction mechanism would be required, which is the focus of the second contribution of this dissertation. Synthesis results for the fault

localization and abstraction module, proposed in this thesis show that compared to the state-of-the-art, lower area overhead is achieved, while performing its operation in a single clock cycle.

As the faults on the NoC links can affect the network performance, it is crucial to handle network-layer faults affecting the routing algorithm. Moreover, the mechanism used for implementation of the fault-tolerant routing algorithm must be scalable and re-configurable. The mechanism must also be flexible, so that it would allow changing the routing algorithm from one regime to another by the system fault manager. Furthermore, the mechanism must not depend on the location and number of faulty links in the network. This has been the focus of the third contribution of this dissertation. To this end, a logic-based distributed routing mechanism is developed, which is scalable solution compared to routing tables, thus not growing in size with the increasing number of network nodes. The mechanism relies only on a fixed set of configuration bits (computed offline via an algorithm proposed in this dissertation), while having support both for 2D and 3D NoCs and allows deadlock and live-lock-free implementation of turn model-based routing algorithms in such networks. Experimental results for the proposed mechanism for implementing fault-tolerant routing algorithms confirms the scalability of the proposed mechanism compared to the state-of-the-art, making it a viable solution for large network sizes, while not affecting the performance (average packet latency) significantly compared to other approaches.

Appendix A

This Appendix includes the second example in which the proposed methodology for devising and evaluating and minimizing concurrent online checkers is applied to the full control part of the Bonfire handshaking router. This can serve as supplementary information for checkers' experiments, related to Chapter **3** of this dissertation. The Appendix covers how functional and structural checkers are devised for Bonfire handshaking router using the proposed methodology in this dissertation, a published in **publication D** [83]. It is worth noting that in this example, similar to the first one, single stuck-at fault has been considered as the fault model. Moreover, the data-path is assumed to be already protected using an Error Detection/Correction Coding technique.

Functional Checkers for Control Part of Bonfire Handshaking Router

For better clarification regarding how functional checkers are devised, the control part of the Bonfire handshaking NoC router (**Architecture 2**) [66] has been chosen as the second example. Functional checkers are devised for *FIFO control part*, routing logic (*LBDR* [65]) and arbitration logic (*arbiter*), as explained in the following.

Bonfire handshaking NoC router FIFO control part functional checkers: Based on the rules existing for the control part of FIFO implemented in Bonfire handshaking flow control router, which is a circular buffer, the following properties must always hold:

- The FIFO cannot be full and empty at the same time.
- According to the design of Bonfire's FIFO, the read pointer and write pointer must always follow the one-hot fashion (since in the router design the read and write pointer are encoded as one-hot). The choice of one-hot encoding is for providing better fault detection capability regarding SEUs and single stuck-at faults.
- It is not possible to read from an empty FIFO or write to a full FIFO.

Bonfire handshaking NoC router routing logic (LBDR) functional checkers: Based on the rules existing for the routing logic, implemented using LBDR, the following properties must always hold:

- When LBDR is configured to the deterministic XY routing algorithm, during the processing of header flit, the output request signals must always follow the one-hot fashion.
- Since the baseline LBDR supports only minimal paths, during routing computation, opposite direction output requests cannot become active at the same time (*e.g.* the request for East and West output cannot be active simultaneously).
- If there is an approved request to LBDR for routing, the output request signals cannot be all zero.
- If the tail flit of the packet is processed by LBDR, all the output request signals must become zero.

For the case of LBDR module in the Bonfire handshaking NoC router (**Architecture 2**), by taking into account the properties shown in the flowchart of **Figure A. 1**, the following higher level (functional) checkers are devised:

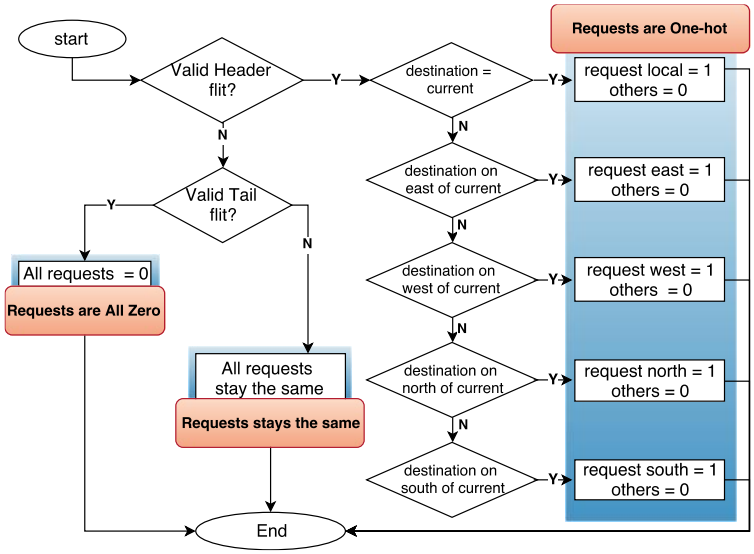


Figure A. 1 Functional checkers devised for the routing logic (LBDR) of Bonfire handshaking NoC router using the proposed.

- If the flit type is header and the corresponding input FIFO is not empty:
  - The output requests must follow the one-hot fashion (when LBDR is configured to the deterministic XY routing algorithm).
  - If the destination is on the North side of the current node (on the same column), then the output request for North port (Req_N_in) must be set to 1 and the other requests must be set to 0. Similar deduction can be inferred for the output requests for other directions (*i.e.* East, West and South).

These checkers are still more abstract than the structural checkers, which will be explained later in this chapter.

Bonfire handshaking NoC router arbitration logic (arbiter) functional checkers: Based on the rules existing for the arbitration logic of Bonfire handshaking router, which is implemented as an FSM-based Round-Robin (RR) prioritization logic, the following properties must always hold:

- The arbiter states must always follow the one-hot fashion (both current and previous values of arbiter states). This is considered in the specification in order to increase the fault detection capability of arbiter against single stuck-at faults and SEUs.
- It would not be possible for arbiter to give grant to a request that is not active. Thus, if a request is zero, its corresponding grant signal must also be zero.
- Since by specification, the Bonfire router only supports unicast communication, it is not possible to send data from an input port to multiple output ports at the same time. Therefore, no matter how many requests from the routing modules (LBDR modules) come to the arbiter of an output port, arbiter must always generate the grant signals following the one-hot fashion, or in case no request

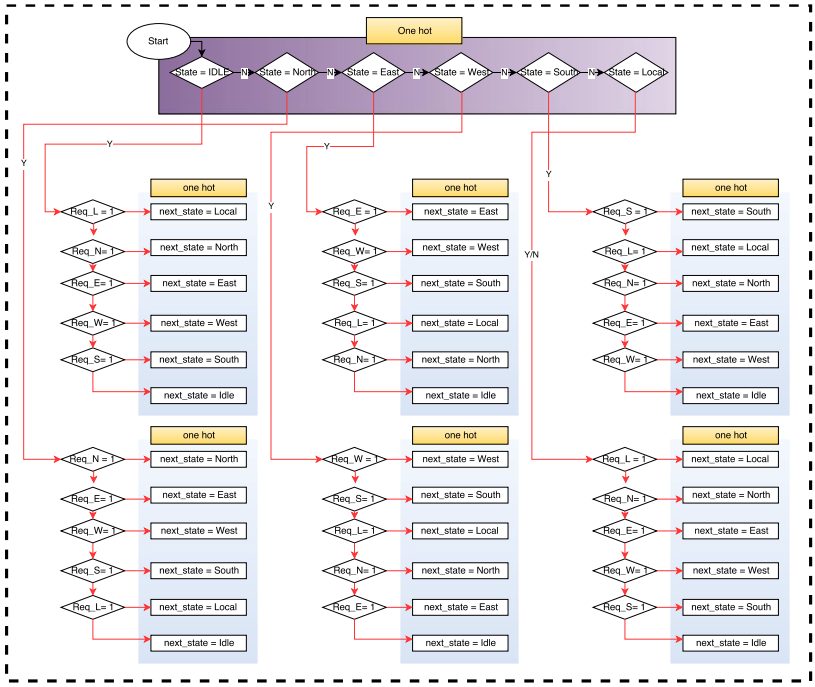


Figure A. 2 Flowchart of applying the proposed methodology for devising checkers from arbiter of Bonfire handshaking flow control router (checkers for arbiter's FSM)

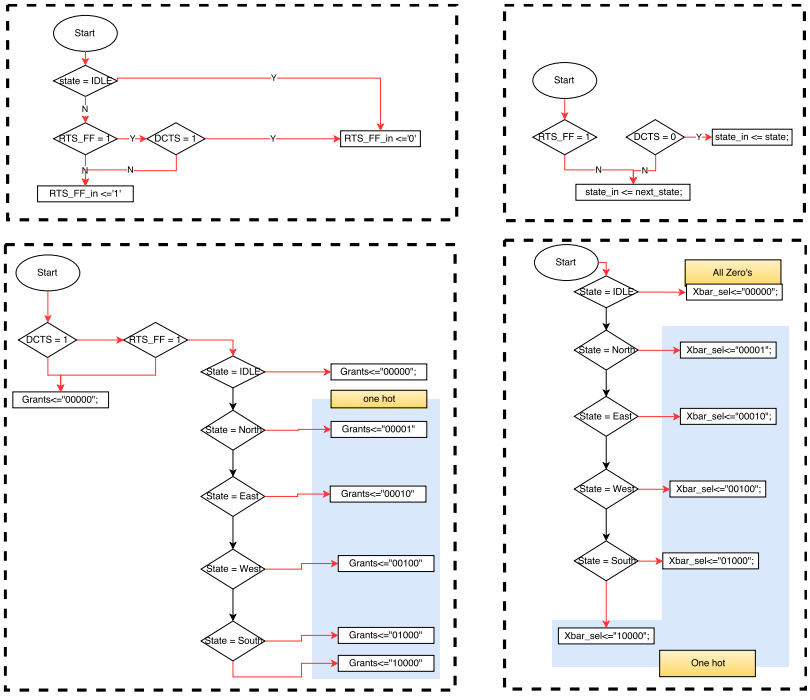


Figure A. 3 Flowchart of applying the proposed methodology for devising checkers from arbiter of Bonfire handshaking router (checkers for arbiter's handshaking signals, grant signals and crossbar select lines)

is granted, all grant signals must remain zero (grant signals can only be one-hot or all zero).

- Since arbiter is also in charge of selecting the crossbar to allow the flits be forwarded to the granted output, the select lines of crossbar switch are handled by arbiter. In Bonfire handshaking router, the select lines of crossbar switches are encoded as one-hot, for higher fault detection capability in case of SEUs and single stuck-at faults.

The yellow rectangles in **Figure A. 2** and **Figure A. 3** demonstrate the higher level (functional) checkers devised for the arbiter logic of Bonfire handshaking router.

Structural Checkers for Control Part of Bonfire Handshaking Router

For the example of Bonfire handshaking NoC router, in addition to high-level (functional) checkers, the methodology for devising structural checkers is also applied to the pseudo-combinational version of each control part module. The flowcharts representing the devised structural checkers for FIFO's control part, routing logic (LBDR) and arbiter Bonfire handshaking router are explained in the following.

Structural checkers for the routing logic (LBDR) of Bonfire handshaking NoC router: As it can be seen in the flowchart of **Figure A. 4**, the following cases are possible to occur in the RTL code of routing logic (LBDR) of Bonfire handshaking router:

- If the flit type is header and the corresponding input FIFO is empty, then all request must keep their previous value.
- If the flit type is body (or invalid), then all requests must keep their previous values.
- If the flit type is tail, then all requests must be zero.
- 8 checkers can be devised that check the properties of N1, E1, W1 and S1 (shown with green and red arrows in **Figure A. 4**). These signals show the direction or quadrant on which the destination node is located with respect to the current node:
    - If the destination node is located on the North side with respect to the current node, then N1 must be 1.
    - If the destination node is not located on the North side with respect to the current node, then N1 must be 0.
    - If the destination node is located on the East side with respect to the current node, then E1 must be 1.
    - If the destination node is not located on the East side with respect to the current node, then E1 must be 0.
    - If the destination node is located on the West side with respect to the current node, then W1 must be 1.
    - If the destination node is not located on the West side with respect to the current node, then W1 must be 0.
    - If the destination node is located on the South side with respect to the current node, then S1 must be 1.
    - If the destination node is not located on the South side with respect to the current node, then S1 must be 0.

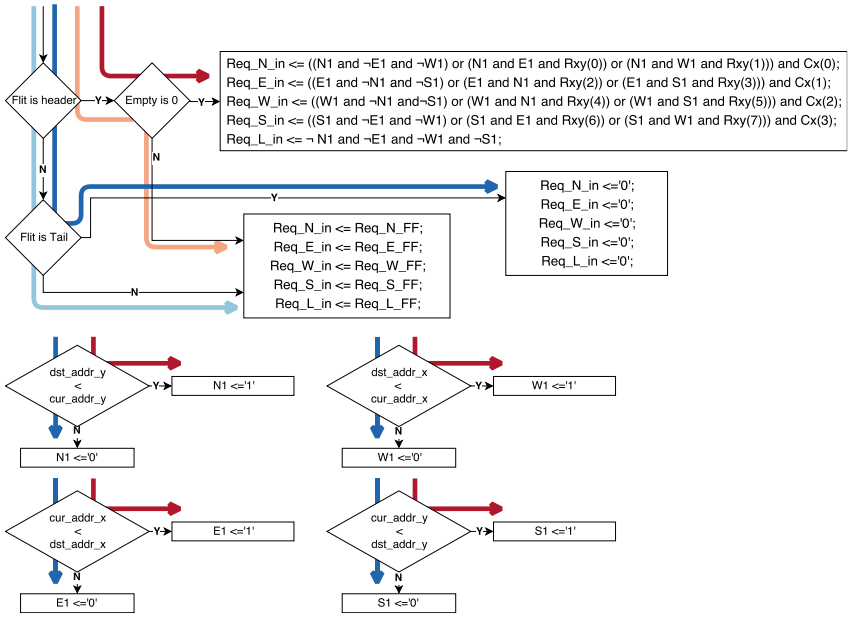


Figure A. 4 Flowchart of applying the proposed methodology for devising structural checkers from LBDR (routing) logic of Bonfire handshking flow control router

- If the flit type is header and the corresponding FIFO connected to the routing logic is not empty, then output request for North port (Req_N_in) must be correctly set according to the logic of LBDR (the routing algorithm is assumed to be XY routing).
- If the flit type is header and the corresponding FIFO connected to the routing logic is not empty, then output request for East port (Req_E_in) must be correctly set according to the logic of LBDR (the routing algorithm is assumed to be XY routing).
- If the flit type is header and the corresponding FIFO connected to the routing logic is not empty, then output request for West port (Req_W_in) must be correctly set according to the logic of LBDR (the routing algorithm is assumed to be XY routing).
- If the flit type is header and the corresponding FIFO connected to the routing logic is not empty, then output request for South port (Req_S_in) must be correctly set according to the logic of LBDR (the routing algorithm is assumed to be XY routing).
- If the flit type is header and the corresponding FIFO connected to the routing logic is not empty, then output request for Local port (Req_L_in) must be correctly set according to the logic of LBDR. This case would occur when the current node is the destination node and therefore, all the N1, E1, W1 and S1 signals are zero.

Structural checkers for the arbitration logic (arbiter) of Bonfire handshaking NoC router: The flowcharts in **Figure A. 2** and **Figure A. 3** demonstrate the different possible paths in the RTL code of the Round-Robin (RR) arbiter for the Bonfire handshaking router. The arbiter has an internal Finite State Machine (FSM) with five different states, encoded

as one-hot: IDLE, North, East, West, South and Local. Each state denotes that the arbiter is serving the corresponding input port. For the case of IDLE, it means there is no request for arbitration from the inputs. The order in which the arbiter in Bonfire handshaking router serves the requests from inputs (LBDR logics) is from highest to lowest: North, East, West, South, Local and again North and so on (in a circular manner). Based on this prioritization, as it can be seen in **Figure A. 2**, For example, when the current state of the arbiter is IDLE (no requests to arbitrate), first the request from Local input (Req_L) is checked and if there is such a request, the state variable of the FSM changes to Local at the next clock cycle (meaning that it will be serving Local input).

The same applies to other states in the order of L, N, E, W and S. Thus, the structural checkers devised for checking the correct order of state variable of the arbiter can be extracted using the proposed methodology from the RTL code the pseudo-combinational version of the FSM of the arbiter.

Similar approach has been followed, parsing the other sections of the Bonfire handshaking arbiter's RTL code, including the code sections in charge of the computation of the handshaking signals, *i.e.* RTS and DCTS (as shown in **Figure A. 3**). RTS is in charge of generating request from current router to the next router or Network Interface, if the data on the corresponding output port is valid. DCTS is used for receiving the signal from the next router or Network Interface that there are enough free buffer slots in the downstream side to receive data from the current router.

In addition, as shown in **Figure A. 3**, part of the structural checkers is checking the values generated for the grant signals based on the current state of arbiter. Each grant signal corresponds to a request from an input direction. Finally, since arbiter is also in charge of activating the correct path from an input to the granted output and selecting the correct crossbar switch, the final set of checkers check the logic used to generate select signals (which are also encoded as one-hot in the arbiter of Bonfire handshaking router). The values of the select lines are checked by taking into account the current state of the arbiter's FSM (as dictated by its RTL code and demonstrated in **Figure A. 3**).

Bonfire handshaking NoC router FIFO control part structural checkers: As it can be seen in **Figure A. 5**, based on the RTL code of the FIFO of Bonfire handshaking router, the control part consists of a write pointer and read pointer which are both encoded as one-hot for improving the fault detection capability. The first two parts of **Figure A. 5**, parse the part of the code related to the values of read and write pointers. The write pointer only gets updated when there is a request for writing to a FIFO slot. Similarly, the read pointer gets updated only when there is a request for reading from a FIFO slot and the corresponding input buffer is not empty.

In addition to arbiter, FIFO is also charge of handling part of the handshaking signals, *i.e.* generating CTS and interpreting DRTS (both connected to the previous router or Network Interface (NI)). CTS informs the previous router or Network Interface that the FIFO of the current router has at least one free slot and therefore, it is not full. DRTS examines the RTS signal from the previous router or Network Interface, which denotes when the input data on the links are valid for the current router to read. The checkers related to DRTS and CTS are shown in **Figure A. 5**, which are also connected to a compact FSM consisting of two states in the FIFO's control part, *i.e.* IDLE and READ_DATA states. The FIFO's control part will only go to the READ_DATA state if CTS has been zero in the previous cycle and there is an active input request from the

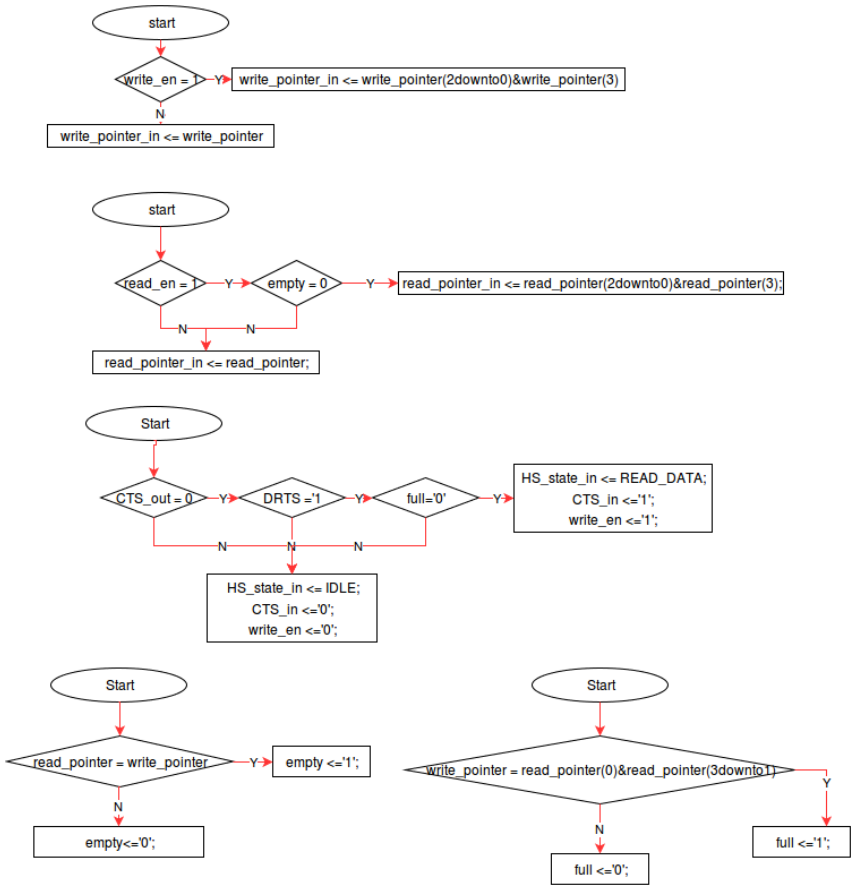


Figure A. 5 Flowchart of applying the proposed methodology for devising structural checkers from control part of FIFO in Bonfire handshaking flow control router

previous router/Network Interface (DRTS is one), and also the buffer of the current router is not full. The structural checkers related to this part of logic, which are devised from the RTL code of the control part of the FIFO using the proposed methodology, are shown in **Figure A. 5**.

Finally, since the full and empty signals of the FIFO get their values based on the positions where the read and write pointers point to in the buffer, their correct values must also be checked. This is done by taking into account the RTL code of FIFO's control part in charge of generating the full and empty signals.

It is worth noting that none of the devised structural checkers are inferred by the functionality of the FIFO's control part, but they are all rather devised by traversing all possible paths in the RTL code of the pseudo-combinational version of FIFO's control part, checking each condition and the corresponding outputs of that condition (which would be a relation between an input signal and an internal signal/output signal). That is one reason why it is mentioned earlier in this thesis that as opposed to the functional checkers, the structural checkers examine different specific parts of the same circuit which do not have any overlaps.

Full Set of Devised Checkers for Control Part of Bonfire Handshaking Router

Table A. 1 lists the initial set of checkers for control part modules of Bonfire handshaking router, devised using the methodology proposed in this dissertation.

It is worth noting that the each of the control part modules with all the checkers integrated (without any minimization) impose area overhead to the control part module as follows: LBDR, Arbiter and FIFO control part with full set of checkers incur an overhead of 159%, 193% and 96%, respectively to their corresponding non-fault-tolerant circuits (without any checkers).

Table A. 1 The complete list of devised functional and structural checkers for the control part of Bonfire handshaking NoC router

FIFO Control Part Logic Checkers	
Checker Number(s)	Checker(s) description
1, 2	Depending on the value of the write enable signal, the write pointer of FIFO's control part must update accordingly (one-hot).
3, 4	The value of empty signal should be set based on the values of read pointer and write pointer.
5, 6	The value of full signal should be set based on the values of read pointer and write pointer.
7, 8	Depending on the value of the read enable and empty signals, the read pointer of FIFO's control part must update accordingly (one-hot).
9-12	Depending on the previous value the handshaking signals and also the full signal, the current value of handshaking signals and write enable signals of FIFO's control part must have the correct values.
13	If FIFO is not empty and at least one of the read enable signals is active, the read enable signal generated inside FIFO's control part must be set to one.
Routing Logic (LBDR) Checkers	
1	If the flit type is header and input FIFO is not empty, current values of output requests of LBDR must be one-hot.
2, 4	If the flit type is header or body and input FIFO is empty, the output requests of LBDR must preserve their previous values.
3	If the flit type is tail, the current values of LBDR output requests must be all zero (there should be no request generated).
5, 6, 7, 8, 9, 10, 11, 12	Based on the location of the destination node with respect to the current node, the correct corresponding internal signal of LBDR, related to each cardinal direction (North, East, West or South) should get activated.
13, 14	If the flit type is header and the input FIFO is not empty, when all the internal signals of LBDR corresponding to the cardinal directions are zero, only the request for Local (L) output port can be activated. Also, when the destination address of the header flit is not the same as the current address of the router (node), the Local (L) output request of LBDR must not go high.
15, 16, 17, 18	If the flit type is header (routing computation must be performed on it) and the input FIFO is not empty, the output requests of LBDR for the cardinal directions (North, East, West and South) must go active according to calculated internal signals in one-hot fashion (due to XY routing).

Arbitration Logic (Arbiter) Checkers	
1, 2	If the FSM of Arbiter is in IDLE state, the select lines for XBAR (Crossbar Switch) must correspond to it. Also, if it is not in IDLE state, the XBAR select lines must always follow the one-hot encoding.
3	If Arbiter's FSM is in IDLE state, the current value of RTS handshaking signal must be zero.
4-6	If Arbiter is not in IDLE state then corresponding handshaking signals must have correct value.
7, 8, 9	Depending on the values of the handshaking signals, the previous and current values for Arbiter's FSM state variable must be set accordingly.
10-14	Depending on the values of the handshaking signals and state of Arbiter's FSM, the output grant signals of Arbiter must have the correct value and a one-hot grant should be issued.
15-44	Depending on the previous state of Arbiter's FSM and the request signals from LBDR modules, the correct order of prioritization must always be followed in Arbiter's FSM in a circular way (Local, North, East, West and then South and then back to Local) and also the state of the FSM must be updated accordingly.
45, 46	The current and next values of Arbiter's FSM state variable must always follow the one-hot encoding.
47-51	If the handshaking signals are high, depending on the state variable of the Arbiter's FSM, the grant signal should also be generated correctly.
52-56	The value of the XBAR select lines must correspond to the state that Arbiter's FSM is in it.

APPENDIX B

This Appendix is dedicated to the third example for applying the proposed methodology for devising, evaluating and minimizing checkers to the control part of the Bonfire credit-based router.

Example Three: Devising checkers for the Control Part of Bonfire Credit-based NoC Router

In the third example, since the fault detection information of all checkers was necessary for the fault localization module in order to model turn faults and compress the big data obtained from the checkers (explained in Chapter 4 of this dissertation), minimization part of the proposed methodology is not used and all the devised set of checkers are maintained for maximum fault localization accuracy.

The final set of checkers, along with the fault localization module for modelling turn faults have been integrated in the Bonfire credit-based router. As it was already explained in more detail in Chapter 4, the fault localization and abstraction module compresses the checker outputs to a final set of only 20 bits, representing 20 turn faults in the router.

Table B. 1 lists the full set of devised checkers for the control part of Bonfire credit-based router. The checkers are grouped based on the property and/or part of the module they are checking. Also, for each checker it is marked whether it is structural (marked as *S* in the table) or functional (marked as *F* in the table).

Table B. 1 Full set of devised functional and structural checkers for Bonfire credit-based NoC router

FIFO Control Part Logic Checkers			
Checker Number(s)	Checker Type		Checker(s) description
	F	S	
1	✓		FIFO cannot be empty and full at the same time.
2	✓		Reading from an empty FIFO is not possible.
3	✓		Writing to a full FIFO is not possible.
4	✓		The states of the packet dropping FSM of FIFO must always be one-hot.
5	✓		Read pointer of FIFO must follow the one-hot fashion.
6	✓		Write pointer of FIFO must follow the one-hot fashion.
7, 8		✓	Checkers related to the logic of FIFO write pointer value update: Write pointer must get updated according to the one-hot encoding, when there is a request for writing to the FIFO (the FIFO is circular).
9, 10		✓	Checkers related to the logic of empty signal in FIFO: Only when read pointer and write pointer are pointing to the same location, the empty signal should go high (the FIFO is circular).
11, 12		✓	Checkers related to the logic of full signal in FIFO: Only when read pointer is pointing to the immediate location after where write pointer is pointing to, the full signal should go high (the FIFO is circular).
13, 14		✓	Checkers related to the logic of FIFO read pointer value update: Read pointer must get updated according to the one-hot encoding, when there is a request for reading from the FIFO (the FIFO is circular).

15-18		✓	Checkers contributing to the logic of write enable signal, which is used for flagging a write request to FIFO.
19, 20		✓	Checkers contributing to the logic of read enable signal, which is used for flagging a read request from FIFO.
21-25		✓	Checkers related to the fake credit counter update logic in FIFO. This is used in the packet dropping process to manipulate the previous router or NI by generating a fake credit out.
26-28		✓	Checkers contributing to the logic of credit out signal, which is used to signal the previous router or Network Interface (NI) that the current router has enough free FIFO slots for storing a flit.
29-110		✓	Checkers related to the packet dropping FSM logic of FIFO.
Routing Logic (LBDR) Checkers			
1-4		✓	Checkers related to the generated Requests by LBDR (Requests are generated based on the routing algorithm and the destination address of the packet).
5	✓		Checker related to the grants signal received from the allocator corresponding to different output directions. If there is at least one grant signal from one of the output directions, the grants signal cannot be zero.
6		✓	Checker related to the grants signal received from the allocator corresponding to different output directions. If there are no active grant signals, the grants signal cannot go high.
7		✓	Checkers related to the generated Requests by LBDR (Requests are generated based on the routing algorithm and the destination address of the packet). These checkers check the previous and current values of the Requests generated by LBDR logic.
8, 9		✓	Checkers contributing to the first phase of LBDR logic, which generates the signals indicating that the destination node is towards to the North direction or a quadrant related to North direction (North-East or North-West quadrant).
10, 11		✓	Checkers contributing to the first phase of LBDR logic, which generates the signals indicating that the destination node is towards to the East direction or a quadrant related to East direction (North-East or South-East quadrant).
12, 13		✓	Checkers contributing to the first phase of LBDR logic, which generates the signals indicating that the destination node is towards to the West direction or a quadrant related to West direction (North-West or South-West quadrant).
14, 15		✓	Checkers contributing to the first phase of LBDR logic, which generates the signals indicating that the destination node is towards to the South direction or a quadrant related to South direction (South-East or South-West).
16, 17		✓	Checkers contributing to the logic for generating Local output request. If the packet has reached its destination, the Local output request must go high Also, if the packet has not reached its destination, the Local output request cannot go active.
18, 19		✓	Checkers contributing to the packet dropping request generated by LBDR module in case of detection of a faulty flit of a packet. This is used for packet dropping in case a flit's contents get damaged after read from FIFO and entered the LBDR logic.

20		✓	Checker related to the generated Requests by LBDR (Requests are generated based on the routing algorithm and the destination address of the packet).
21		✓	Checker contributing to the second phase of LBDR logic, generating the request for North output port.
22		✓	Checker contributing to the second phase of LBDR logic, generating the request for East output port.
23		✓	Checker contributing to the second phase of LBDR logic, generating the request for West output port.
24		✓	Checker contributing to the second phase of LBDR logic, generating the request for South output port.
25-29		✓	Checkers contributing to the packet dropping request generated by LBDR module in case of detection of a faulty flit of a packet. This is used for packet dropping in case a flit's contents get damaged after read from FIFO and entered the LBDR logic.
30-39		✓	Checkers contributing to the reconfiguration of the connectivity bits (4 bits per router).
40-46		✓	Checkers contributing to the reconfiguration of the routing bits (8 bits per router).
Arbitration Logic (Allocator) Checkers			
Allocator Internal Logic and Credit Counter Logic Checkers			
1-10		✓	Checkers related to the logic generating internal grant signals for North output port based on requests from different input ports.
11-20		✓	Checkers related to the logic generating internal grant signals for East output port based on requests from different input ports.
21-30		✓	Checkers related to the logic generating internal grant signals for West output port based on requests from different input ports.
31-40		✓	Checkers related to the logic generating internal grant signals for South output port based on requests from different input ports.
41-50		✓	Checkers related to the logic generating internal grant signals for Local output port based on requests from different input ports.
51, 52		✓	Checkers contributing to final grant signal related to North output port.
53, 54		✓	Checkers contributing to final grant signal related to East output port.
55, 56		✓	Checkers contributing to final grant signal related to West output port.
57, 58		✓	Checkers contributing to final grant signal related to South output port.
59, 60		✓	Checkers contributing to final grant signal related to Local output port.
61	✓		This checker makes sure the valid out signal generated by the Allocator matches the grant signal generated (each valid out signal for a specific output direction corresponds to the grant signal for that direction).
62-67		✓	Checkers contributing to the credit counters related to North output port.
68-73		✓	Checkers contributing to the credit counters related to East output port.
74-79		✓	Checkers contributing to the credit counters related to West output port.

80-85		✓	Checkers contributing to the credit counters related to South output port.
86-91		✓	Checkers contributing to the credit counters related to Local output port.
Allocator Arbiter_in Checkers (5 Arbiter_in modules per Allocator)			
1	✓		If there are no requests from the LBDR modules to Arbiter_in of North input port, the FSM state variable of the arbiter must keep its previous value.
2-61		✓	Checkers contributing to checking the prioritizing algorithm of the Round-Robin arbiter. The priority of the requests from inputs from highest to lowest are as follows: North, East, West, South, Local and then again North, and so on (in circular manner).
62	✓		The FSM state variable of Arbiter_in must follow the one-hot encoding.
63	✓		If there are no requests from the LBDR modules to Arbiter_in of North input port, all grant signals must remain low.
64	✓		If there is at least one requests from the LBDR modules to Arbiter_in of North input port, the grant signals cannot be all zero.
65-69	✓		A grant for North input port to an output port cannot be generated if there is no request generated for it by LBDR.
Allocator Arbiter_out Checkers (5 Arbiter_out modules per Allocator)			
1	✓		If there are no requests from the LBDR modules to Arbiter_in of North input port, the FSM state variable of the Arbiter_out must stay in IDLE state.
20-41		✓	Checkers contributing to checking the prioritizing algorithm of the Round-Robin arbiter. The priority of the requests from inputs from highest to lowest are as follows: North, East, West, South, Local and then again North, and so on (in circular manner).
42	✓		The FSM state variable of Arbiter_out must follow the one-hot encoding.
43	✓		If there are no requests from the Arbiter_in modules to Arbiter_out of a specific output port, all grant signals must remain low.
44		✓	If there is at least one requests from the Arbiter_in modules to Arbiter_out of a specific output port port, the state variable of Arbiter_out cannot be IDLE.
45		✓	If the Arbiter_out FSM is in IDLE state, there must be a generated grant and grants cannot be all zero.
46-50		✓	Checkers that make sure the generated grant corresponds to the state that Arbiter_out FSM is currently in. For example, it would be impossible that Arbiter_out FSM is in North state, but the grant signal for another direction except North output goes high.
51	✓		Arbiter_out follows the one-hot fashion for the grants, therefore, since the router does not support multi-casting or broadcasting of packets, grant signals must always be one-hot or all zeros, no other possible combination for them is allowed.

Table B. 2 Total number of functional and structural checkers for Bonfire credit-based NoC router

	Total No. of checkers	No. of Functional checkers	No. of Structural checkers
FIFO Control Part	110	4	104
LBDR	46	0	46
Allocator (*Arbiter_in*)	345	15	330
Allocator (*Arbiter_out*)	255	15	240
Allocator	691	30	661

A summary of the number of functional and structural checkers for each control part module of Bonfire credit-based router is provided in **Table B. 2**.

It is worth noting that the each of the control part modules of Bonfire router **Architecture 3** with all the checkers integrated (without any minimization) imposes area overhead to the control part module as follows: LBDR, Allocator and FIFO control part with full set of checkers incur an overhead of 324.33%, 196.47% and 36.17%, respectively to their corresponding non-fault-tolerant circuits (without any checkers).

APPENDIX C

P. Saltarelli, B. Niazmand, R. Hariharan, J. Raik, G. Jervan and T. Hollstein, "Automated Minimization of Concurrent Online Checkers for Network-on-Chips", 10th International Symposium on Re-configurable Communication-centric Systems-on-Chip (ReCoSoC 2015), June 29- July 1, 2015, Bremen, Germany.

Automated Minimization of Concurrent Online Checkers for Network-on-Chips

Pietro Saltarelli[1,2], Behrad Niazmand[1], Ranganathan Hariharan[1], Jaan Raik[1], Gert Jervan[1], Thomas Hollstein[1]
[1]Tallinn University of Technology, Estonia
[2]Università degli Studi di Ferrara, Italy
pietro.saltarelli@student.unife.it
{bniazmand, jaan, ranga, gert, thomas}@ati.ttu.ee

***Abstract*— The paper introduces automated minimization of a set of concurrent online checkers for Network-on-Chips (NoCs) under given fault detection quality constraints. The proposed framework allows accurate and complete evaluation of the fault detection capabilities of checkers, which in turn enables finding seamless trade-offs between the overhead area of the checkers and the fault detection quality. The features of the automated minimization approach include formal proof for the absence or presence of true misses in checkers and a minimal fault detection latency. The minimization technique is based on a divide-and-conquer approach of partitioning the checkers' fault table into independent clusters. The checkers within the cluster are weighted and the set of checkers is minimized based on a heuristic method. Experiments on the control part (routing and arbitration) of an NoC router show that 100% fault coverage with very low overhead area will be achieved by the proposed minimization approach.**

Keywords—Network-on-Chip, routing logic, arbitration, concurrent online checking.

I. Introduction

Network-on-Chip (NoC) has been introduced as a solution to overcome the scalability and performance constraints of previous on-chip communication architectures such as bus-based networks. One of the challenges in the design of NoC routers is that as more cores get integrated on the same die and nanometer technologies get extremely scaled down, the probability of vulnerability of the components to wear-out and environmental effects increases. These are effects occurring during the life time of the system and cannot be filtered out by manufacturing testing. Thus, concurrent online fault monitors (i.e. *checkers*) for detecting faults during circuit's life time are needed. These checkers would report errors within routers and would allow reconfiguration of the routing infrastructure.

In this paper, we introduce an automated tool flow for obtaining a minimized list of checkers for checking on-chip communication architectures. The flow is based on accurate, automated evaluation of concurrent online checkers. The methodology includes preparation of the checkers in the form of verification assertions (or reuse of existing assertions), creation of a pseudocombinational version of the circuit under test and specifying the environment in terms of valid input stimuli for it. Subsequently, the set of the fault detection characteristics for the checkers, together with the stimuli and the circuit are applied to accurate evaluation. As a result, weights for individual checkers belonging to the set are obtained.

Finally, the number of checkers within the set will be minimized. The minimization technique is based on a divide-and-conquer approach of partitioning the checkers' fault table into independent clusters. Further, weight information of the checkers within the cluster is applied in a heuristic minimization method. The ultimate result will be a minimal selection of checkers to achieve a target fault coverage level.

The underlying approach is complete, i.e. it allows proving the absence or presence of true misses by the checkers. In addition, it provides minimal fault detection latency due to the fact that the circuit is transformed into a pseudo-combinational one and therefore only checkers with a single clock cycle latency are considered. Experiments on the control part (routing and arbitration) of a Network-on-Chip (NoC) router show that 100% fault coverage with very low overhead area will be achieved by the proposed minimization approach.

The paper is organized as follows. Section 2 provides an overview of related works in concurrent online testing. Section 3 explains the concurrent online checking concept. In Section 4, the automated tool flow and the corresponding methodology for checkers' minimization are presented. Section 5 presents the target architecture of the control part of an NoC router. Section 6 discusses application of the checker evaluation and minimization framework to the NoC Router design. Section 7 provides the checkers' evaluation and minimization experiments. Finally, Section 8 concludes the paper.

II. Related Works

Online detection of errors in logic is a thoroughly studied research area. Traditional Triple-Modular Redundancy (TMR) and duplication based approaches are too costly in terms of multiplying the area and correspondingly the power consumption. An alternative to minimize this overhead is the selective TMR that identifies Single Event Upset (SEU) sensitive sub-circuits that are to be protected [1].

In addition, there exists a variety of solutions based on coding techniques such as Berger [2] or Bose-Lin [3] codes. In many works the coding techniques are combined with synthesis [4,5]. The approaches suffer from significant area overhead as well as require alteration of the original circuit in order to generate the codes.

Concurrent on-line built-in self-test techniques such as Built-In Concurrent Self-Test (BICST) [6] and Reduced Observation Width Replication (ROWR) [7] provide high fault coverage at low area overhead but only consider a limited subset of pre-

computed test vectors. Hence these approaches are likely to miss faults occurring in a normal circuit operation.

Several alternatives based on checkers that do not require modification of the circuit under test have been developed. Creating checkers automatically based on logic implications derived from the circuit structure [8] is feasible but suffers from low fault coverage and high area overhead, often exceeding the duplex solutions. On the other hand, deriving checkers from functional assertions, or reusing verification assertions, is similarly known to yield low coverage of structural faults as it is difficult to correlate functional coverage to structural one [9].

Many previous works have focused on addressing faults in the control logic of NoC routers. In [15], Yu et al. have addressed fault tolerance for NoC topologies and proposed an error control method for detecting transient errors in routing logic implemented using Logic-Based Distributed Routing (LBDR) mechanism and its extension for high-radix topologies, LBDRhr. The proposed error control method utilizes the inherent information redundancy (IIR) to reduce the error control overhead. However, the method does not guarantee full fault coverage.

Authors of [16] have presented a method for online error detection and diagnosis of NoC switches. The proposed method deals with routing faults that cause NoC packets to be forwarded to output ports that are not intended to. Regarding modeling routing faults in switches, a high-level fault model has been introduced in this work. The fault coverage is measured only at the functional level and there is no estimate of correlation to gate-level fault coverage.

Parikh et al. have proposed ForEVeR [13], where in order to deliver correctness guarantees for the complete network, a network-level detection and recovery solution is devised that monitors the traffic in the NoC and protects it against functional bugs that were not detected during design time. To this end, ForEVeR augments the baseline NoC with a lightweight checker network that alerts destination nodes of incoming packets ahead of time and is used for the recovery process. The approach suffers from extremely high latency. Only 30% of the faults will be detected during the first clock cycle by the approach.

[14] proposes checkers synthesized from a set of 32 verification assertions. The checkers detect most of the injected faults. The faults that are not covered correspond to non-catastrophic failures. The work proposed in [14] lacks the completeness and minimization aspects present in the current paper.

This paper exceeds the existing state-of-the-art in concurrent online checking by proposing a tool flow for automated evaluation and minimization of the verification checkers. We show that starting from a realistic set of verification assertions a minimal set of checkers will be synthesized that provide 100% fault coverage at a low area overhead and the minimum fault detection latency of a single clock-cycle. The latter is especially crucial for enabling rapid fault recovery in reliable real-time systems.

An additional feature of the proposed approach is that it allows formally proving the absence or presence of true misses over all possible valid inputs for a checker, whereas in the case of traditional fault injection only statistical probabilities can be calculated without providing the user with full confidence of fault detection capabilities.

The formal proof as well as the minimal fault detection latency will be guaranteed by reasoning on a pseudo-combinational version of the circuit and by the application of exhaustive valid set of input stimuli as the verification environment.

III. The Concept of Concurrent Checkers

Fig. 1 presents the role of concurrent on-line checkers in detecting faults within a circuit. In addition to the original circuit (functional logic), a set of checkers (checker logic) will be connected to functional inputs/outputs of the circuit. These checkers are derived based on functional assertions obtained from relationships between variables corresponding to inputs and outputs of the circuit. The checker logic targets the faults at lines at the inputs of each gate within the functional logic (marked by green circles). The lines at the functional outputs succeeding the checker inputs (marked by a red cross) cannot be detected by the checker. In addition, the checkers are not targeting the faults at functional inputs preceding checker inputs, since the checker may not detect that the input value has been altered by a fault (such functional input lines are also marked by a red cross in Fig. 1). In this paper, we consider the single stuck-at fault model. However, due to the fact that concurrent checkers are implemented and a single time-frame is targeted, the model also covers timing related faults.

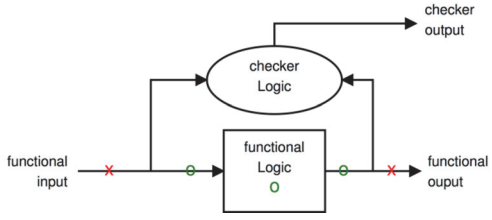


Fig. 1. The concept of concurrent checking

Given a fault at a line within the functional logic and a set of input stimuli, four possible scenarios may occur:

- ***Case 1***: Fault occurs at an internal line and is visible at functional output(s) and checker logic flags a violation. The term *True Detection* is used to describe this situation, since a critical fault is effectively detected by the checker.

- ***Case 2***: Fault occurs at an internal line but is not visible at primary output(s). Checker catches the fault and flags a violation. The term *False Positive* is used to describe this situation. False positive is not harmful because an error is flagged which did not have any effect. However, it has negative impact on design's performance because normally it causes re-execution of the task. In the experiments in this paper we did not encounter any cases of false positives.

- ***Case 3***: Fault occurs at internal line but is not visible at primary output(s) and the checker logic does not detect the violation. The term *Benign Miss* is used to describe this situation. Benign miss shows correct operation by the checker.

- ***Case 4***: Fault occurs at internal node and is visible at primary output(s). Checker does not detect violation. The term *True Miss* is used to describe this situation, which is the worst possible case. True miss means that the fault propagates to the functional outputs and onwards to the system. However, the system has no information that a critical fault has occurred.

Traditionally, in order to evaluate the fault detection quality of the checkers, *fault injection* has been applied. Fault injection refers to injecting faults into a circuit at a certain time step and simulating it with the input stimuli to see whether any functional output of the circuit changes and whether any of the checker output fires. Due to the fact that it is generally impossible to inject and simulate all the faults at each circuit line at each time step, a statistically significant sample of random faults would normally be injected and simulated.

However, in this paper a methodology is proposed which is based on automated extraction of a pseudo-combinational circuit out of the original functional logic by breaking the flipflops and converting them to pseudo primary inputs and pseudo primary outputs. Further, an exhaustive test for the extracted circuit is fed through a filtering tool in order to derive the complete valid set of input stimuli which will serve as the environment for checker evaluation. This means that in this paper full evaluation of the checkers with all the valid stimuli and faults is obtained.

Let D be the number of true detections, X be the number of benign misses and W be the number of true misses over all the injection runs. Then we define the metrics of *Fault Coverage* (*FC*) and *Checkers' Efficiency Index* (*CEI*) as follows.

$$FC = \frac{D + X}{D + X + W} \quad (1)$$

$$CEI = \frac{D}{D + W} \quad (2)$$

Here, FC shows the probability of the checkers behaving correctly over all possible fault cases while CEI shows the probability of checkers ability to detect critical faults. Due to the fact that none of the checkers resulted in false positives, this information is excluded from the metrics.

IV. Checkers Evaluation and Minimization Flow

Fig. 2 presents the evaluation and minimization flow for the checkers. The flow starts with synthesizing the checkers from a set of combinational assertions. Thereafter, a pseudo-combinational circuit will be extracted from the circuit of the design under checking. The pseudo-combinational circuit is derived out of the original circuit by breaking the flipflops and converting them to pseudo primary inputs and pseudo primary outputs. Note, that at this point additional checkers that also describe relations on the pseudo primary inputs/outputs may be added to the checker suite in order to increase the fault coverage.

Subsequently, the checker evaluation environment is created by generating exhaustive test stimuli for the extracted pseudo-combinational circuit. This stimuli are fed through a

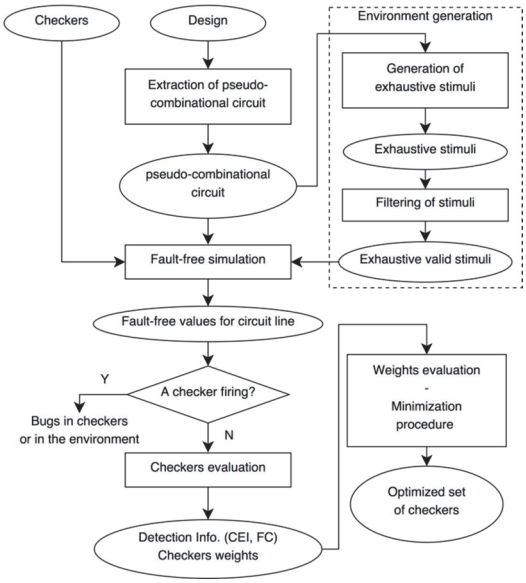


Fig. 2. Checkers' Evaluation and Minimization Flow

filtering tool that selects only the stimuli that correspond to functionally valid inputs of the circuit. As a result, the complete valid set of input stimuli that will serve as the environment for checker evaluation is obtained.

The obtained environment, pseudo-combinational circuit and synthesized checkers are applied to fault free simulation. The simulation calculates fault free values for all the lines within the circuit. Additionally, if any of the checkers fires during fault-free simulation it means a bug in the checker or an incorrect environment. During the case study presented in Section 5 several bugs were detected by this simulation step.

If none of the checkers is firing in the fault-free mode then checker evaluation takes place. The tool injects faults to all the lines within the circuit one-by-one and this step is repeated for each input vector. As a result, the overall fault detection capabilities for the set of checkers, in terms of FC and CEI metrics will be calculated. In addition, each individual checker will be weighted by summing up the total number of true detections by the checker.

Finally, the weighting information will be exploited in minimizing the number of checkers, eventually allowing to outline a trade-off between CEI, or FC, and the area overhead due to the introduction of checker logic.

The framework is developed as an extension of a freeware test system Turbo Tester [10]. The system applies Structurally Synthesized Binary Decision Diagram (SSBDD) models [11] for circuit modeling.

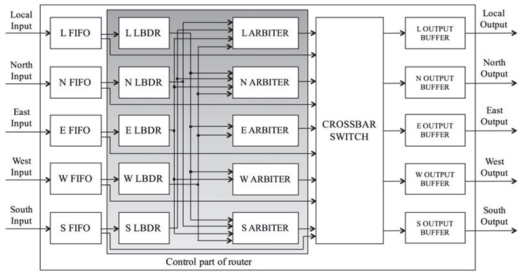


Fig. 3. High-level overview of an NoC router

V. Target Architecture: NoC Router

Fig. 3 demonstrates the high-level overview of a 5-port 2D NoC router that we have chosen as a target architecture for applying the checkers. Mainly, the router consists of a datapath and a control part. The datapath is composed of input buffers (implemented as First-In-First-Out (FIFO)), one for each input port, a crossbar switch and an output buffer for each output port.

The flow of data through the data path is managed and controlled by the control part, which consists of a routing computation unit for each input port and an arbitration unit (arbiter) for each output port, which prioritizes the requests from different input ports to the corresponding output port. The router has 5 input/output ports, four ports connected to four cardinal directions (North – N, East – E, South – S, West – W) and one Local (L) port connected to the local processing element. The NoC router utilizes wormhole switching. Therefore, packets are sent in form of flits, consisting of header flit, body flit(s) and tail flit.

For the routing computation unit of our target architecture, we have opted for Logic-Based Distributed Routing (LBDR) [12], which is considered as a scalable solution compared to routing tables. The mechanism describes the topology and the routing function in form of connectivity and routing bits, therefore the logic can be easily re-configured. Routing decision is distributed and only requires local and destination addresses for forwarding flits.

In this work we focus on a 2D Mesh topology, we consider XY as the routing algorithm, which is a deterministic dimension-ordered algorithm, and we assume that 180 degrees turns are not allowed. This would in turn lead to further simplification of the logic of LBDR. The basic mechanism of the logic is shown in Fig. 4, for instance for the East input port.

For the arbitration unit (arbiter) we have chosen Round-Robin (RR) policy for prioritizing the requests from the routing logic of different input ports. Prioritization is circular, thus ensuring the absence of starvation, and guaranteeing that eventually any input port will get access to the requested output port.

Arbiter grants the access to the requesting input port winning the eventual contention, allowing data to go from the input FIFO to the corresponding output port, through the crossbar switch. The arbitration mechanism is based on an internal Finite State Machine (FSM). In this work one-hot encoding has been considered for the state variable, in order to improve detections of faults in the logic. Moreover, one-hot encoding is extended to grant signals and select lines for the crossbar switch.

The design decision to implement a one-hot encoded arbiter state machine versus a decimal encoded one did increase the area of the arbiter by 27.7%. However, the CEI nearly doubled from 58.55% to 100% and the fault coverage increased from 93.69% to 100%, respectively.

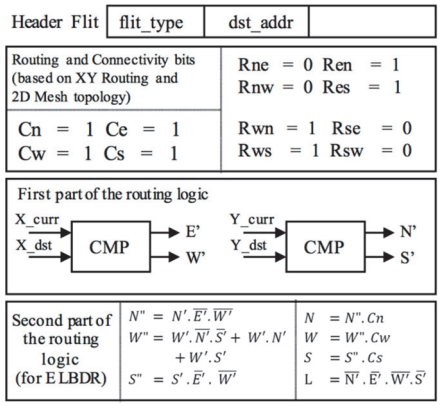


Fig. 4. Logic-based Distributed Routing (LBDR) logic for the East input port

VI. Application of the Framework to the Design

In the control part of the router, we have limited our focus to the case in which the LBDR and arbiter logic have the most number of connected signals, more specifically considering ELBDR and SArbiter. For ELBDR the existing output port signals are N, W, S and L and for SArbiter, request and grant signals exist for N, E, W and L. Such scenario provides the case with the most number of connectivities between LBDR and arbiter logic. The checkers that cover faults for such scenario, are symmetrical to the other cases (different connections between each LBDR logic to arbiter logics).

From the output of the checker evaluation tool it can be observed that the two set of checkers for the ELBDR and the SArbiter are *independent*, i.e. they cover faults for different and separate parts of the circuit, without any overlap. Therefore the fault table will be partitioned into two clusters. First, the ELBDR alone will be considered. Secondly, the circuit under study will be expanded, interconnecting the routing logic with the SArbiter. The second considered scenario is depicted in Fig. 5.

Connectivity and routing bits and also the current address are set to fixed values according to the scenario under consideration: 2D Mesh topology, XY routing algorithm, 180 degrees turns not allowed, focus on router with ID 5 in a 4x4 network. This scenario allows minimizing the number of circuit inputs and previous state input bits that together form the inputs for the pseudo-combinational circuit to be considered in both

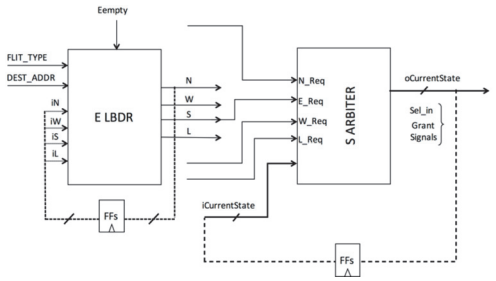


Fig. 5. The pseudo-combinational circuit for the full scenario

experiments. When ELBDR only is considered, the amount of inputs is limited to 11 bits:

- 2 flit identifier bits;
- 4 destination address bits;
- 4 ELBDR previous output values bits;
- 1 empty bit (coming from East input buffer).

With the interconnection to the SArbiter in the second experiment, the number of input bits is increased to 19, introducing:

- 3 SArbiter request signals bits;
- 5 SArbiter previous state bits.

This, in turn, makes the exhaustive approach in checker evaluation fully feasible.

Once the pseudo-combinational circuit to be studied is extracted, a set of checkers can be devised from the functional behaviour of the considered circuit, evaluating the possible implications existing in between input and output signals. It is interesting to underline that *a priori* it may be very difficult to outline the effectiveness of a single checker or the overlap of different checkers in detection.

Together with the considered pseudo-combinational circuit and its set of checkers, a set of input patterns is needed for performing fault simulation. The exhaustive test would require 2^{11}=2,048 and 2^{19}=524,288 input stimuli, respectively for the ELBDR and for the East-South control path experiments. However, in order to minimize the stimuli, and more important, to avoid checkers being evaluated in non-realistic conditions, the exhaustive set of stimuli has to be filtered to contain only the functionally feasible values.

The filtering step is based on the implemented routing algorithm (i.e. allowed destinations from the current router), restrictions in the routing logic (e.g. no 180 degrees turns) and emptiness condition of the input buffer, as well as on invalid conditions for the state of the arbiter logic (i.e. violation of one-hot encoding - only for the second experiment). It is important to stress the fact that none of the checkers is firing in fault free simulation with any of the considered input stimuli, in neither of the scenarios.

TABLE I. PROPOSED CHECKERS FOR ELBDR

Checkers for Routing Logic (LDBR)		
1	Valid LBDR output	If there is a request to the routing logic (the corresponding input buffer is not empty), LBDR has to compute at least one valid output direction (according to XY routing).
2	No LBDR output	If no flit arrives (the corresponding input buffer is empty), all the output port signals of LBDR should remain zero.
3	Single LBDR output	If the corresponding input buffer is not empty (there is a request to LBDR), because of using XY routing, at most only one output port signal of the LBDR logic can become active.
4	Switch LBDR output	If the corresponding input buffer is not empty (there is a request to LBDR) and a non-header flit has arrived, LBDR outputs should remain the same.
5	Local Port output	If the corresponding input buffer is not empty (there is a request to LBDR) and a header flit has arrived, the local output should become active only if the packet has reached its destination.

VII. EXPERIMENTAL RESULTS

Experiments for the checker evaluation and minimization framework were carried out on the scenarios described in previous section, first on the ELBDR circuit only, then on its interconnection with the SArbiter, as displayed in Fig. 5. In both cases an initial set of checkers was devised a priori, together with a filtering scheme to obtain a valid set of input stimuli. Each individual checker was weighted by the tool by summing up the total number of true detections by the checker, and this information was used in a heuristic way to minimize the initial set of checkers, with the final aim of achieving highest possible CEI and FC, and at the same time with the lowest possible area overhead. These quantities were evaluated iterating the fault

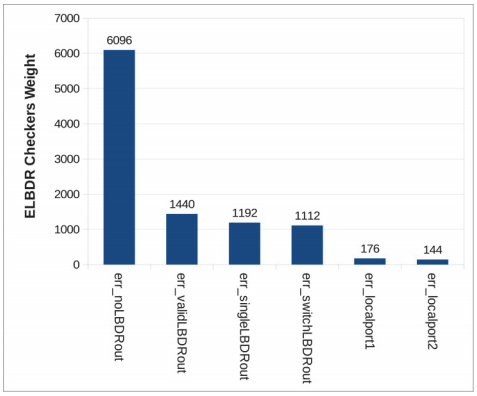


Fig. 6. Weights of checkers proposed for EBLDR

simulation, including at each step the next heaviest checker still not included in the currently considered set of checkers, initialized only with the first heaviest checker.

ELBDR experiment

All the experiments in this paper were carried out on an Asus ux32vd-r4002v computer with a 1.9 GHz Intel Core i7-3517U processor and 10 GB RAM. Table I lists the initial a priori set of checkers for ELBDR, devised from the functionality of the logic. The pseudo-combinational circuit for ELBDR has 11 input bits, as mentioned in the previous section, thus the exhaustive set of stimuli presents 2^{11}=2,048. A filtering scheme based on the following statements was devised:

- if input buffer's empty signal is high, any other input bit is meaningless, and therefore any value is allowed for it;
- if the incoming flit is a header, the destination address has to be valid according to the XY routing and turns restrictions;
- if the incoming flit is a body or tail flit, the previous output values must be valid, they must follow a one-hot fashion, according to XY routing.

This allowed to obtain a valid and complete set of stimuli consisting of 1536 vectors, which forms 75% of the exhaustive set. The run-time for generating the stimuli was 2 seconds.

Fig. 6 displays the weight information output of the tool, on the initial set of checkers for the ELBDR. The checker, *err_noLBDRout* (checker 2 in Table I) is considerably detecting more faults than any other checker. The 5 remaining checkers, in descending order of weights are *err_validLBDRout* (checker 1), *err_singleLBDRout* (checker 3), *err_switchLBDRout* (checker 4), and finally the two *err_localport* checkers (entry 5). The checkers' analysis required 10 ms of run-time from the proposed framework.

Fig. 7 depicts the results obtained with the weight-based greedy heuristic approach applied to the ELBDR and its initial set of checkers, in terms of achieved CEI, FC and area overhead. Considering at first only the heaviest weight, and adding at each step the next heaviest checker still not included in the considered set, all the quantities gradually increase. When the three most significant checkers are used, CEI and FC reach 100%. This result shows that minimization of the set of considered checkers is achieved, with the three heaviest checkers *dominating* the three lightest, i.e. the three considered checkers cover all the faults detected by the other checkers. Reducing the used set of checkers to the three most significant ones allows to limit the area overhead to 78.57% over the ELBDR circuit, far lower than 185.71% imposed by the initial non-minimized set of checkers.

TABLE II. PROPOSED CHECKERS FOR THE ARBITER LOGIC

Checkers for the Arbiter logic		
6	Valid Grant output	If there is a request from LBDR, arbiter has to assert at least one of the grant signals for the corresponding output direction.
7	No Grant output	If there is no request to the arbiter, it should not assert any of the grant signals for any direction.
8	Invalid Grant output	Whenever there is a request to the arbiter, the grant signals should go active corresponding to that specific requested direction and invalid direction should not be chosen.
9	Invalid arbiter output state	Output state variable (oScurrentState – which represents the grant signals) in arbiter's pseudo-combinational circuit can not possess invalid values due to the one-hot coding.
10	Invalid IDLE state for arbiter input state	If the input previous state variable (iScurrentstate) is in IDLE state and there is a request for arbitration from LBDR, oScurrentstate should not remain in IDLE state i.e. a grant signal should be asserted.
11	Priority Grant	In case there is one or multiple request(s) to the arbiter, it should follow the correct prioritization (Local, North, East and then West) according to the input previous state variable (iScurrentstate).

ELBDR + SArbiter combined scenario experiment

ELBDR is connected to SArbiter according to Fig. 5, thus providing the East request signal to the arbitrating logic. Table II lists the a priori initial set of checkers for the arbiter. Multiple individual checkers are grouped to the same table entry according to types. The initial set amounts to 28 checkers.

The exhaustive test for the considered pseudo-combinational circuit would require 2^{19}=524,288 input stimuli. The test stimuli were generated in 270 seconds run time. The considered filtering scheme is an extension of the one used for the ELBDR experiment valid input patterns set, adding the one-hot encoding restraint to the 5 previous state value bits of the arbitrating pseudo-combinational unit. This allowed to shrink the exhaustive set of 2^{19} input stimuli to a valid and complete set consisting of 61,440 input vectors, which is less that 12% of the initial number. This may be considered as a proof of the effectiveness of the one-hot encoding for the arbiter state variable.

First, the evaluation tool was run considering the whole set of checkers for the SArbiter, altogether with the minimized set of 3 checkers for the ELBDR. This analysis required 1 second of run time by the framework. Figure 8 lists the considered 31 checkers, with their corresponding weights in a descending order. Focusing on the arbitrating unit, two checkers look to be far more significant than the others, *Serr_validgrant* (checker 6 in Table II), *Serr_invalidstate* (checker 9), both of them monitoring different aspects of the one-hot encoding condition for the arbiter's state variable.

From the output of the evaluation tool it can be observed that the two set of checkers for the ELBDR and the SArbiter are *independent*, i.e. they cover faults for different and separate parts of the circuit, without any overlap. For this reason the minimized set of ELBDR checkers is used, and the previously introduced weight-based greedy minimization heuristic is applied to the SArbiter checkers set.

Fig. 9 displays the obtained results. As it could have been expected from the weighting information in Fig. 8, the two most significant checkers dominate all the lightest checkers, ensuring

100% CEI and FC. Thus, considering a total of 3 ELBDR and 2 SArbiter checkers, area overhead over the partial control path circuit is limited to 56.82%, while using the whole initial set of 28 checkers for the SArbiter would lead to 170.45% area overhead. It is interesting to observe that the minimized set of 5 checkers corresponds to one third of the whole 31 checkers set area.

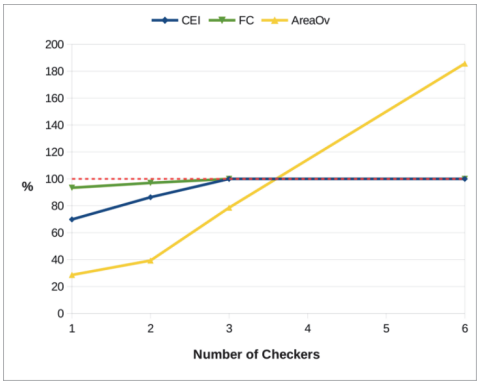


Fig. 7. ELBDR scenario results

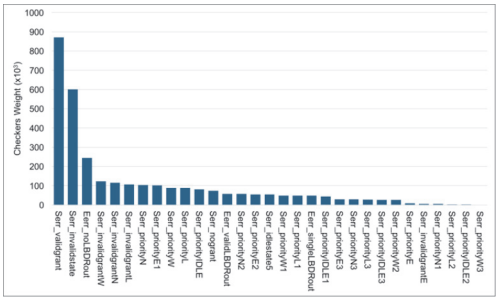


Fig. 8. ELBDR + SArbiter checkers ranked by weights

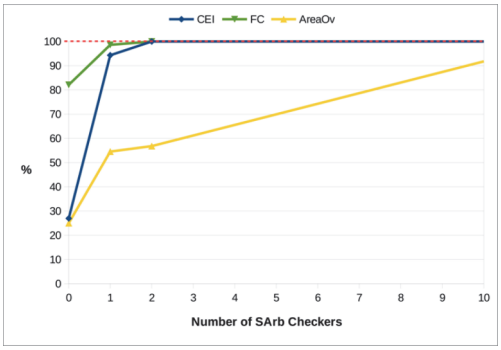


Fig. 9. ELBDR + SArbiter scenario results

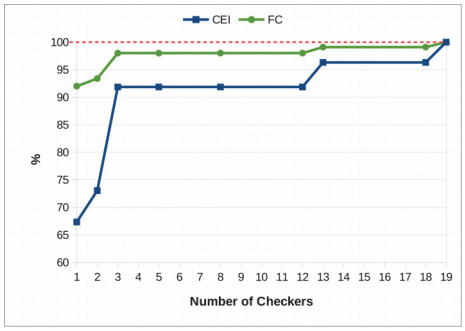


Fig. 10. Results without considering independent clusters

Checker	Weight
Serr_validgrant	871552
Serr_invalidstate	600512
Eerr_noLBDRout	243840
Eerr_validLBDRout	57600
Eerr_singleLBDRout	47680

Fig. 11. Weights for minimized set of checkers

Impact of clustering the faults

Assuming that we had no information of the overlap of faults detected by the checkers for ELBDR and SArbiter, the weight-based greedy heuristic, starting from the heaviest checker *Serr_validgrant*, would add at each step the next heaviest checker still not considered in the current set of checkers, based on the weight information displayed in Fig. 8. Fig. 10 shows the inefficiency of the heuristic approach caused by the lack of the clustering information. The number of steps in the greedy procedure is heavily increased, and only after 19 steps, when the *Eerr_singleLBDRout* checker is considered, the 100% upper bound for CEI and FC is reached.

However, when partitioning of the fault set to clusters is taken into account and minimization is performed on the clusters separately then total of five steps are needed. Fig. 11 illustrates the importance of considering the clustering information. It can be observed that the weights of the ELBDR checkers are far less than those of the SArbiter, but they are still needed to achieve full coverage for the considered design.

VIII. Conclusions

The paper proposes a new tool providing an automated flow for evaluation and minimization of concurrent online checkers, which is formal (able of proving the presence or absence of true misses), yields minimal fault detection latency and enables accurate, fully automated evaluation of the fault detection characteristics of a given set of checkers.

Experiments carried out on the control part (routing and arbitration) of a Network-on-Chip (NoC) router showed on a realistic application the feasibility and efficiency of the framework and the underlying methodology. Experimental results showed that the approach allowed selecting the minimal

set of 5 checkers out of 31 verification assertions with the fault coverage of 100% and area overhead of only 56.82%.

ACKNOWLEDGEMENT

The work has been supported by EU FP7 STREP BASTION, EU's H2020 RIA IMMORTAL, Estonian Science Foundation grant ETF9429, Estonian institutional research grant IUT 19-1, funded by Estonian Ministry of Education and Research, and by EU through the European Structural and Regional Development Funds.

REFERENCES

[1] R. Sedmak and H. Liebergot. Selective triple modular redundancy (STMR) based single-event upset (SEU) tolerant synthesis for FPGAs. *IEEE Transactions on Nuclear Science*, 51:2957-2969, 2005.

[2] J. M. Berger. A note on an error detection code for asymmetric channels. *Information and Control*, 4:68-73, 1961.

[3] D. Das and N. A. Touba. Synthesis of circuits withlow-cost concurrent error detection based on Bose-Lin codes. *In VLSI Test Symposium*, pages 309-315, 1998.

[4] K. Mohanram, E. Sogomonyan, M. Gossel, and N. Touba. *Synthesis of low-cost parity-based partially self-checking circuits*, 2003.

[5] S. Ghosh, N.A. Touba, and S. Basu. Synthesis of low power ced circuits based on parity codes. *In VLSI Test Symposium*, pages 315-320, 1-5 May 2005.

[6] R. Sharma and K.K. Saluja. An implementation and analysis of a concurrent built-in self-test technique. *In Digest of Papers Eighteenth International Symposium on Fault-Tolerant Computing FTCS-18*, pages 164- 169, June 1988.

[7] P. Drineas and Y. Makris. Concurrent fault detection in random combinational logic. *In Proceedings Fourth International Symposium on Quality Electronic Design ISQED*, pages 425-430, March 2003.

[8] Alves, N.; Shi, Y.; Dworak, J.; Bahar, R.I.; Nepal, K. "Enhancing online error detection through area-efficient multi-site implications", *IEEE 29th VLSI Test Symposium (VTS)*, pp. 241 – 246, 2011.

[9] Marc Boule, Jean-Samuel Chenard, and Zeljko Zilic. Assertion checkers in verification, silicon debug and infield diagnosis. *In Proceedings of the ISQED '07*.

[10] M Aarna, E Ivask, A Jutman, E Orasson, J Raik, R Ubar, V Vislogubov, HD Wuttke. Turbo Tester-Diagnostic Package for Research and Training. *The 1st East-West Design and Test Conference*, Alushta, 2003.

[11] Artur Jutman, A Peder, J Raik, M Tombak, R Ubar. Structurally synthesized binary decision diagrams. *6th International Workshop on Boolean Problems.* pp. 271-278, 2004.

[12] J. Flich, J. Duato, Logic-Based Distributed Routing for NoCs, *IEEE Computer Architecture Letters*, Vol. 7, No. 1, January-June 2008.

[13] R. Parikh and V. Bertacco. Formally enhanced runtime verification to ensure NoC functional correctness. *In Proc. of the International Symposium on Microarchitecture (MICRO)*, 2011.

[14] Prodromou, A.; Panteli, A.; Nicopoulos, C.; Sazeides, Y., "NoCAlert: An On-Line and Real-Time Fault Detection Mechanism for Network-on-Chip Architectures," *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 60-71, 1-5 Dec. 2012.

[15] Yu, Qiaoyan; Cano, J.; Flich, J.; Ampadu, P., "Transient and Permanent Error Control for High-End Multiprocessor Systems-on-Chip," *2012 Sixth IEEE/ACM International Symposium on Networks on Chip (NoCS)*, vol., no., pp.169,176, 9-11 May 2012.

[16] Alaghi, A.; Karimi, N.; Sedghi, M.; Navabi, Z., "Online NoC Switch Fault Detection and Diagnosis Using a High Level Fault Model," *22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems*, 2007. DFT '07., vol., no., pp.21,29, 26-28 Sept. 2007.

APPENDIX D

P. Saltarelli, B. Niazmand, J. Raik, R. Hariharan, V. Govind, T. Hollstein and G. Jervan, "A framework for combining concurrent checking and on-line embedded test for low-latency fault detection in NoC routers", 9th International Symposium on Networks-on-Chip (NOCS) 2015, September 28-30, 2015, Vancouver, Canada.

A framework for combining concurrent checking and on-line embedded test for low-latency fault detection in NoC routers

Pietro Saltarelli[1,2], Behrad Niazmand[1], Jaan Raik[1], Ranganathan Hariharan[1], Vineeth Govind[1], Thomas Hollstein[1], Gert Jervan

[1]Tallinn University of Technology, Estonia
[2]Università degli Studi di Ferrara, Italy

***Abstract*— The focus of the paper is detection of faults in NoC routers by combining concurrent checkers with embedded on-line test to enable cost-effective trade-offs between area-overhead and test coverage. First, we propose a framework of tools for formally evaluating the quality of the checkers and for optimizing the overhead area with given fault coverage constraints. The stress is in particular on the minimization of the error detection latency, which is a crucial aspect in order to eliminate (or limit) error propagation. Second, the concurrent checkers will be complemented by embedded on-line test packets which are to be applied as a periodic routine during the idle periods in router operation. The framework together with the corresponding methodology has been successfully applied to a realistic case-study of a fault tolerant NoC router design. The case study shows that combining concurrent routers with embedded test allows reducing the area overhead of the checkers from 31-35% down to 1.5-10% without sacrificing the fault coverage.**

Keywords—network-on-chip, fault tolerant router design, concurrent online checking, embedded test, test packets.

I. Introduction

One of the main challenges related to the design of Network-on-Chip (NoC) routers is the extreme down-scaling of modern technologies that increases the probability of the components to wear-out as well as their vulnerability towards environmental effects. These are phenomena occurring during the life-time of the system and cannot be screened out by manufacturing testing. Thus, cost-efficient mechanisms for detecting faults during system's life-time are needed. These mechanisms should detect errors within routers and enable reconfiguration of the routing network in order to isolate the problem and provide graceful degradation for the system. In this paper, we propose combining concurrent checkers with embedded on-line test packets in order to achieve early and cost-effective detection of faults in NoC routing infrastructure.

Regarding the development of on-line checkers, we introduce a new framework and a methodology with a stress on the level of automation, fault coverage, detection latency and area-efficiency. The methodology consists of four main steps. The first step of the methodology is *formal checker qualification* which includes identification of control-intensive parts of the router architecture, converting them to pseudo-combinational counterparts, preparation of the checkers synthesized from verification assertions and specifying the environment in terms of valid input stimuli for the pseudo-combinational circuit. As a result, the faults detected by each individual checker will be calculated.

Second, the number of checkers within the set will be minimized by applying the *checker optimization* step. As a starting point is the fault detection characteristics for each individual checker as well as their weights in terms of silicon area. Further, a heuristic minimization method is applied resulting in a minimal selection of checkers to achieve a target fault coverage level. The minimization technique is based on a divide-and-conquer approach of partitioning the checkers' fault table into independent clusters. This approach is very effective as the checkers devised for different modules normally do not have overlapping fault sets.

Third, and optional, step of the methodology includes devising additional checkers from temporal assertions for modules that do not achieve 100% fault detection. For these checkers the formal qualification step described above is not possible and traditional *fault injection* experiments are carried out by a sequential fault simulation tool included to the framework.

Finally, the checkers for the control part of the router are to be complemented by embedded on-line test packets which are to be applied as a periodic routine during the idle periods in router operation, e.g. slacks in the task scheduling. The framework together with the corresponding methodology has been successfully applied to a realistic case-study of a fault tolerant NoC router design. The case study shows that combining concurrent checkers with embedded test packets allows reducing the area overhead of the checkers from 31-35% down to 1.5-10%, depending on the router bitwidth, without sacrificing the fault coverage.

The paper is organized as follows. Section 2 provides an overview of related works in concurrent online testing and embedded test for NoC routers. Section 3 explains the concurrent online checking concept. Section 4 discusses application of embedded test packets. In Section 5, the automated framework and the corresponding methodology for checkers' minimization combined with the embedded test are presented. Section 6 discusses application of the framework and the underlying methodology to the NoC router design. Section 7 provides the experiments. Finally, Section 8 concludes the paper.

II. Related Works

Online detection of errors in logic is a thoroughly studied research area. Traditional Triple-Modular Redundancy (TMR) and duplication based approaches are too costly in terms of multiplying the area and correspondingly the power consumption. An alternative to minimize this overhead is the selective TMR that identifies Single Event Upset (SEU) sensitive sub-circuits that are to be protected [1].

In addition, there exists a variety of solutions based on coding techniques such as Berger [2] or Bose-Lin [3] codes. In many works the coding techniques are combined with synthesis [4,5]. The approaches suffer from significant area overhead to the design to be checked.

Concurrent on-line built-in self-test techniques such as Built-In Concurrent Self-Test (BICST) [6] and Reduced Observation Width Replication (ROWR) [7] provide high fault coverage at low area overhead but only consider a limited subset of pre-computed test vectors. Hence these approaches are likely to miss faults occurring in a normal circuit operation.

Several alternatives based on checkers that do not require modification of the circuit under test have been developed. Creating checkers automatically based on logic implications derived from the circuit structure [8] is feasible but suffers from low fault coverage and high area overhead, often exceeding the duplex solutions. On the other hand, deriving checkers from functional assertions, or reusing verification assertions, is similarly known to yield low coverage of structural faults as it is difficult to correlate functional coverage to structural one [9].

Many previous works have focused on addressing faults in the control logic of NoC routers. In [16], Yu et al. have addressed fault tolerance for NoC topologies and proposed an error control method for detecting transient errors in routing logic implemented using Logic-Based Distributed Routing (LBDR) mechanism and its extension for high-radix topologies, LBDRhr. The proposed error control method utilizes the inherent information redundancy (IIR) to reduce the error control overhead. However, the method does not guarantee full fault coverage.

Authors of [17] have presented a method for online error detection and diagnosis of NoC switches. The proposed method deals with routing faults that cause packets to be forwarded to unintended output ports. Regarding modeling routing faults in switches, a high-level fault model has been introduced in this work. The fault coverage is measured only at the functional level and there is no estimates on correlation to gate-level fault coverage.

In order to deliver correctness guarantees for the complete network, Parikh et al. have proposed a network-level detection and recovery solution ForEVeR [14] that monitors the traffic in the NoC and protects it against functional bugs that were not detected during design time. To this end, ForEVeR augments the baseline NoC with a lightweight checker network that alerts destination nodes of incoming packets ahead of time and is used for the recovery process. The approach suffers from extremely high latency. Only 30% of the faults will be detected during the first clock cycle by the approach.

The work in [15] proposes checkers synthesized from a set of 32 verification assertions. The checkers detect most of the injected faults. The faults that are not covered correspond to non-catastrophic failures. The work proposed in [15] is not automated and lacks the completeness and minimization aspects present in the current paper.

In [18] a hybrid method is introduced for synthesis of fault-secure NoC switches utilizing error detecting codes for the data path (data flits) and a concurrent error detection structure for dealing with faults not covered by the flit encoding (using multiple parity trees). However, the work still results in more than 50% area overhead.

The use of embedded test configurations for testing the datapath of NoC routers has been proposed in [19], with design-for-testability structures included in [20] and built-in self-test application in [21]. However, all the mentioned approaches are targeting the global network and not a concrete router. Furthermore, only off-line test scenarios have been considered in [19-21].

This paper exceeds the existing state-of-the-art in fault tolerant router design by proposing:

- a framework for *formal checker qualification*. The underlying approach is complete, i.e. it allows proving the absence or presence of true misses by the checkers. In addition, it provides minimal fault detection latency due to the fact that the circuit is transformed into a pseudo-combinational one and therefore only checkers with a single clock cycle latency are considered.
- *automated minimization of checkers*. The formal qualification of the combinational checkers provides the fault detection capabilities for them. These, along with the checker area requirements are applied in an automated minimization process resulting in a minimal area overhead checker solution under certain fault coverage constraints.
- complementing the resulting checkers withtemporal checkers and *on-line embedded test packets*. This enables combining best of both worlds. In the case of NoC control part, where embedded test packet based approaches have proven inefficient, low area concurrent checkers are applied. On the other hand, in the datapath, the embedded test yields full fault coverage whereas error correcting codes would be expensive.

Experimental results on a realistic NoC router design demonstrate the efficiency of the proposed approach.

III. The Concept of Concurrent Checkers

Fig. 1 presents the role of concurrent on-line checkers in detecting faults within a circuit. In addition to the original circuit (functional logic), a set of checkers (checker logic) will be connected to functional inputs/outputs of the circuit. These checkers are derived based on functional assertions obtained from relationships between variables

corresponding to inputs and outputs of the circuit. The checker logic targets the faults at lines at the inputs of each gate within the functional logic (marked by green circles). The lines at the functional outputs succeeding the checker inputs (marked by a red cross) cannot be detected by the checker. In addition, the checkers are not targeting the faults at functional inputs preceding checker inputs, since the checker may not detect that the input value has been altered by a fault (such functional input lines are also marked by a red cross in Fig. 1). In this paper, we consider the single stuck-at fault model. However, due to the fact that concurrent checkers are implemented and at-speed embedded test packets are applied, the model also covers timing related faults.

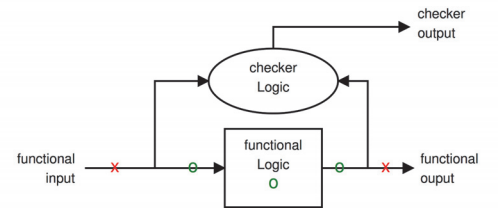


Figure 1. The concept of concurrent checking

Given a fault at a line within the functional logic and a set of input stimuli, four possible scenarios may occur:
Case 1: Fault occurs at an internal line and is visible at functional output(s) and checker logic flags a violation. The term *True Detection* is used to describe this situation, since a critical fault is effectively detected by the checker.

Case 2: Fault occurs at an internal line but is not visible at primary output(s). Checker catches the fault and flags a violation. The term *False Positive* is used to describe this situation. False positive is not harmful because an error is flagged which did not have any effect. However, it has negative impact on design's performance because normally it causes re-execution of the task.

Case 3: Fault occurs at internal line but is not visible at primary output(s) and the checker logic does not detect the violation. The term *Benign Miss* is used to describe this situation. Benign miss shows correct operation by the checker.

Case 4: Fault occurs at internal node and is visible at primary output(s). Checker does not detect violation. The term *True Miss* is used to describe this situation, which is the worst possible case. True miss means that the fault propagates to the functional outputs and onwards to the system. However, the system has no information that a critical fault has occurred.

Traditionally, in order to evaluate the fault detection quality of the checkers, *fault injection* has been applied. Fault injection refers to injecting faults into a circuit at a certain time step and simulating it with the input stimuli to see whether any functional output of the circuit changes and whether any of the checker output fires. Due to the fact that it is generally impossible to inject and simulate all the faults at each circuit line at each time step, a statistically significant sample of random faults would normally be injected and simulated.

However, in this paper a methodology is proposed which is based on automated extraction of a pseudo-combinational circuit out of the original functional logic by breaking the flipflops and converting them to pseudo primary inputs and pseudo primary outputs. Further, an exhaustive test for the extracted circuit is fed through a filtering tool in order to derive the complete valid set of input stimuli which will serve as the environment for checker evaluation. This means that in this paper full formal qualification of the combinational checkers with all possible stimuli and faults can be obtained.

Let D be the number of true detections, X be the number of benign misses, F be the set of false positives and W be the number of true misses over all the injection runs. In order to evaluate the fault detection capabilities of the checkers we define the metrics of *Fault Coverage* (*FC*), *Checkers' Efficiency Index* (*CEI*) and *False Positive Ratio* (*FPR*) as follows.

$$FC = \frac{D + X}{D + X + W} \quad (1)$$

$$CEI = \frac{D}{D + W} \quad (2)$$

$$FPR = \frac{F}{F + X} \quad (3)$$

Here, FC shows the probability of the checkers behaving correctly over all possible fault cases, CEI shows the probability of checkers ability to detect critical faults whereas FPR reports the ratio of false positives over all the cases a fault did not propagate to circuit outputs. The mentioned three metrics are calculated for checkers by the automated checker qualification framework proposed in this paper.

IV. Embedded Online Test Packets

The functional fault model that is applied to cover the stuck-at faults in the datapath of the NoC router is based on the idea proposed for functional testing of mesh-like NoC networks in [19-21]. However, in this paper the fault model is applied to a "localized" approach, where resources (i.e. processing elements) connected to neighbouring routers West (W), East (E), North (N), South (S), and Local (L) are utilized as senders/receivers of test packets to test the central router as the Circuit Under Test (CUT). Figure 2 visualizes the overall setup of the sending/receiving resources and the CUT.

In the proposed setup, whenever there are idle periods or slacks in scheduling with length K for the send/receive resources, K test patterns will be applied from them. This will be done periodically fetching K next tests from the test

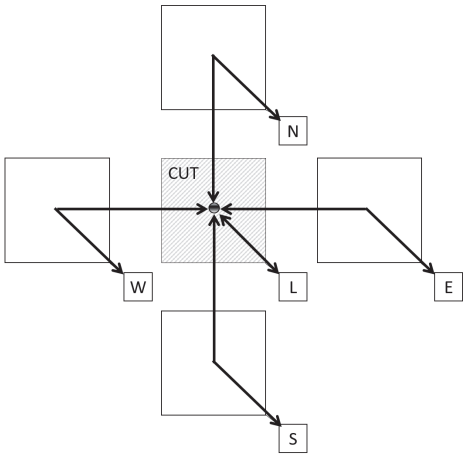


Figure 2. The setup for sending/receiving test packets

set in a circular manner, i.e. if the end of the test is reached then it starts again from the beginning. This scenario provides online test capabilities for regularly checking the health of the datapath of the routing infrastructure.

In the proposed setup, whenever there are idle periods or slacks in scheduling with length K for the send/receive resources, K test patterns will be applied from them. This will be done periodically fetching K next tests from the test set in a circular manner, i.e. if the end of the test is reached then it starts again from the beginning. This scenario provides online test capabilities for regularly checking the health of the datapath of the routing infrastructure.

A fault model proposed in [19-21] is applied, where the value at a selected router input is distinguished from the values at other inputs of the router. In order to fully cover the structural faults in the multiplexers of the crossbar, tests for each address value have to be performed. An additional constraint is that all turns must be covered by the distinguishing tests. In [19] it was shown that by applying them, near 100% fault coverage for the crossbar switch and the I/O buffers comprising the datapath of the NoC router is achieved.

V. Framework and Methodology

This Section presents the framework for fault tolerant NoC router design that has been developed as an extension of the Turbo Tester test framework [10]. The proposed methodology of combining concurrent checkers with embedded online test consists of three main steps:

1. Checkers' qualification and minimization (combinational checkers);
2. Checkers' evaluation by fault injection (temporal checkers);
3. Fault simulation of the embedded online test packets.

In the following, these steps are explained in more detail.

A. Checker Qualification and Minimization

Fig. 3 presents the qualification and minimization flow for the checkers. The flow starts with synthesizing the checkers from a set of combinational assertions. Thereafter, a pseudo-combinational circuit will be extracted from the circuit of the design under checking. The pseudo-combinational circuit is derived out of the original circuit by breaking the flipflops and converting them to pseudo primary inputs and pseudo primary outputs. Note, that at this point additional checkers that also describe relations on the pseudo primary inputs/outputs may be added to the checker suite in order to increase the fault coverage.

Subsequently, the checkers' qualification environment is created by generating exhaustive test stimuli for the extracted pseudo-combinational circuit. This stimuli are fed through a filtering tool that selects only the stimuli that correspond to functionally valid inputs of the circuit. As a result, the complete valid set of input stimuli that will serve as the environment for checkers' qualification is obtained.

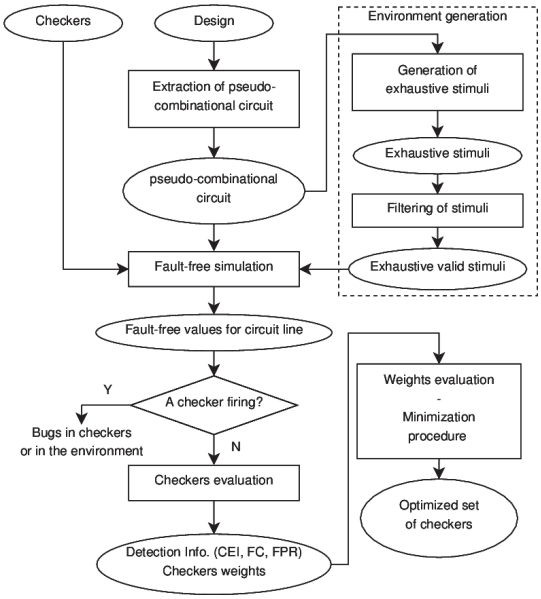


Figure 3. Checkers' qualification and minimization flow

The obtained environment, pseudo-combinational circuit and synthesized checkers are applied to fault free simulation. The simulation calculates fault free values for all the lines within the circuit. Additionally, if any of the checkers fires during fault-free simulation it refers either to a bug in the checker or an incorrect environment.

If none of the checkers is firing in the fault-free mode then checkers' qualification takes place. The tool injects faults to all the lines within the circuit one-by-one and this step is repeated for each input vector. As a result, the overall fault detection capabilities for the set of checkers, in terms of FC, CEI and FPR metrics will be calculated. In

addition, each individual checker will be weighted by summing up the total number of true detections by the checker.

The weighting information will then be exploited in minimizing the number of checkers, eventually allowing to outline a trade-off between the fault coverage, and the area overhead due to the introduction of checker logic.

B. *Checkers' Evaluation by Fault Injection*

There are cases when a module under checking cannot be handled by the combinational checker qualification and minimization approach. For example the module may have a large number of inputs so that the set of generated valid input stimuli would be too large (e.g. datapath modules) and/or the fault coverage reached by the combinational checkers is too low.

In those cases, the checkers are to be evaluated by traditional fault injection. Here a test bench is created for the design and the circuit with the checkers is simulated by a sequential fault simulator with a sufficiently large random sample of faults injected into the circuit. In this paper, all the datapath checkers and the FIFO checkers were evaluated using this approach.

C. *Fault Simulation of the Embedded Online Test*

Finally, the stuck-at fault coverage of the online embedded test packets for the datapath of the NoC router is measured by a fault simulator belonging to the framework. As experimental results show, full fault coverage for the datapath with the test application time of 196 clock cycles is achieved.

VI. Experimental Results

Fig. 4 demonstrates the high-level overview of a 5-port 2D NoC router that we have chosen as a target architecture for applying the checkers. The router consists of a datapath and a control part. The datapath is composed of input buffers (implemented as FIFO), one for each input port, a crossbar switch and an output buffer for each output port. The control part contains routing units, arbiters and FIFO control. For the routing unit of our target architecture, we have opted for Logic-Based Distributed Routing (LBDR)[13], which is considered as a scalable solution compared to routing tables. As an arbiter, round-robbin arbitration was implemented.

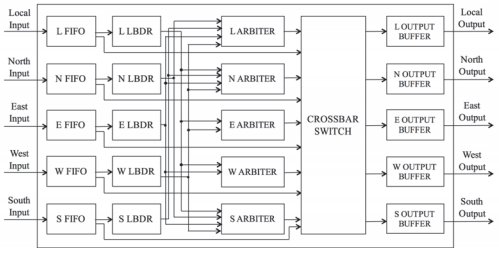


Figure 4. High level architecture of the NoC router

A. *Checker qualification/minimization for LBDR /Arbiter*

The pseudo-combinational circuit for ELBDR has 11 input bits, as mentioned in the previous section, thus the exhaustive set of stimuli presents 2^{11}=2,048. A filtering scheme based on the following statements was devised:

- if input buffer's empty signal is high, any other input bit is meaningless, and therefore any value is allowed for it;
- if the incoming flit is a header, the destination address has to be valid according to the XY routing and turns restrictions;
- if the incoming flit is a body or tail flit, the previous output values must be valid, they must follow a one-hot fashion, according to XY routing.

This allowed to obtain a valid and complete set of stimuli consisting of 1536 vectors, which forms 75% of the exhaustive set. The run-time for generating the stimuli was 2 seconds. (All the experiments in this paper were carried out on an Asus ux32vd-r4002v computer with a 1.9 GHz Intel Core i7-3517U processor and 10 GB RAM.)

Table I lists the obtained minimized set of three checkers for the LBDR. Reducing the set of checkers to the three most significant ones allows to limit the area overhead to 78.57% over the ELBDR circuit, far lower than 185.71% imposed by the initial non-minimized set of checkers, while the CEI and FC remain at 100%.

TABLE I. Minimized list of checkers for LBDR

Checkers for Routing Logic (LDBR)		
1	Valid LBDR output	If there is a request to the routing logic (the corresponding input buffer is not empty), LBDR has to compute at least one valid output direction (according to XY routing).
2	No LBDR output	If no flit arrives (the corresponding input buffer is empty), all the output port signals of LBDR should remain zero.
3	Single LBDR output	If the corresponding input buffer is not empty (there is a request to LBDR), because of using XY routing, at most only one output port signal of the LBDR logic can become active.

Similarly, Table II lists the minimized set of two checkers for the Arbiter that was obtained from an initial set of 28 verification checkers by applying the checker qualification and minimization framework.

TABLE II. Minimized list of checkers for Arbiter

Checkers for Arbiter logic		
4	Valid Grant output	If there is a request from LBDR, arbiter has to assert at least one of the grant signals for the corresponding output direction.
5	Invalid arbiter state	State variable of the arbiter FSM has to respect one-hot encoding.

B. *Fault injection experiments for the FIFO*

Table III lists the set of 8 checkers generated from the verification assertions for the FIFO control part. The checkers were evaluated by the fault injection tool of the framework. A set of input stimuli for the FIFO was devised, aiming to cover all the possible situations for the control logic. The following conditions were considered in the pattern generation procedure:

- reset condition;
- filling the FIFO, followed by reading up to empty condition;
- smooth traffic condition, i.e. concurrent writing and reading operations, avoiding the FIFO to get full;
- idle condition, i.e. write and read enable signals low, during reading and writing operations, in different conditions of fulfillment of the buffer.

100% CEI and FC were achieved on the control part of the fifo, considering the patterns derived from the previously listed conditions, amounting to 134. Run time for the experiment was 0.06 s. No false positives were encountered in this experiment.

TABLE III. CHECKERS FOR THE FIFO CONTROL PART

Checkers for FIFO control part		
6	Reset checker	Whenever reset goes high, at the next clock cycle empty flag should be high (reading and writing pointer are reset to the same value).
7	Flags checkers	Empty and full flags should never be high at the same time. Whenever the defining condition occurs, the corresponding flag should go high at the next clock cycle.
8	One-hot pointers checkers	Reading and writing pointers have to respect one-hot encoding.
9	Registers enable DMR checker	Duplication and comparison for the logic enabling the writing operation in data registers.
10	Reading pointer update checker 1	Whenever read enable is high and the fifo is not empty, at the next clock cycle the reading pointer should be updated.
11	Reading pointer update checker 2	If either read enable is low or the fifo is empty, at the next clock cycle the reading pointer should preserve its value.
12	Writing pointer update checker 1	Whenever write enable is high and the fifo is not full, at the next clock cycle the writing pointer should be updated.
13	Writing pointer update checker 2	If either write enable is low or the fifo is full, at the next clock cycle the writing pointer should preserve its value.

Table IV lists the set of 3 additional checkers which were included in order to achieve the full fault coverage after fault injection experiments for the control part identified uncovered faults in the interconnections of control part modules.

TABLE IV. CONTROL PART INFRASTRUCTURE CHECKERS

Control Part Infrastructure Checkers		
14	FIFOs read enable DMR checker	Logic producing read enable signals for the FIFOs (5 OR gates) is duplicated, then real and duplicated outputs are compared.
15	Output registers enable DMR checker	Logic producing enable signals for the output registers (5 OR gates) is duplicated, then real and duplicated outputs are compared.
16	Flit type LBDR error	Flit type field of a flit has to respect one-hot encoding.

C. *Checkers for the datapath*

In order to fully cover the faults in the NoC datapath two types of concurrent checkers were introduced (listed in Table V). First, for each input port an even parity bit is included, whereas each output port has a checker evaluating the even parity. Second, since fault injection experiments for the whole router identified undetected faults within the crossbar multiplexers, dedicated checkers for the crossbar were devised.

TABLE V. CHECKERS FOR THE NOC DATAPATH

Datapath Checkers		
17	Even parity checker	An even parity bit is computed and added to data entering each input port, which is later evaluated before data leaves the router through any of the output ports.
18	Crossbar checker	Crossbar MUXs are duplicated, then real and duplicated outputs are compared.

D. *Putting it all together*

Fig. 5 reports the area overhead required by the checkers for routers of varying bitwidth (from 32 bits to 256 bits). It can be observed from the Figure that the required area for the control part checkers stays constant while the overhead area of datapath checkers (parity and crossbar) grow proportionally to the router size.

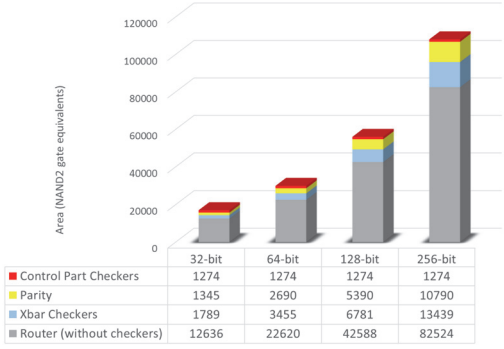


Figure 5. Area consumption for different data-widths

TABLE VI. OVERHEAD AREA FOR DIFFERENT DATAWIDTHS

	32-bit	64-bit	128-bit	256-bit
Router (w/o checkers)	12636	22620	42588	82524
Control part checkers	1274	1274	1274	1274
Xbar Checkers	1789	3455	6781	13439
Parity	1345	2690	5390	10790
Area overhead (contr. p. checkers), %	10.08	5.63	2.99	1.54
Area overhead (all checkers), %	34.88	32.80	31.57	30.90

The same trend is revealed in Table VI. It can be seen that if datapath checkers are included then the required area overhead would be in the range of 31-35%. Whereas, the control part checker circuitry demands significantly less area, especially for larger bitwidths.

However, when combining the control part checkers with embedded online test packets presented in Section 4, full fault coverage for the NoC router can be achieved with a minor area overhead. As it has been shown by experiments in [21] an embedded test of length K=196 clock cycles will achieve FC=100% within the NoC router datapath. Thus, combining the concurrent checkers for the control with embedded test solution for the datapath results in a cost-effective solution for fault tolerant NoC routers.

VII. CONCLUSIONS

The paper proposes a framework for formal qualification of checkers and for minimizing the overhead area with the given fault coverage constraints. The goal is to achieve low-latency, low area overhead checkers for network on chip routers. In addition the paper proposes complementing the concurrent checkers with embedded on-line test packets which are to be applied as a periodic routine during the idle periods in router operation.

The framework together with the corresponding methodology has been successfully applied to a realistic case-study of a fault tolerant NoC router design. The case study shows that combining concurrent routers with embedded test allows reducing the area overhead of the checkers from 31-35% down to 1.5-10% without sacrificing the fault coverage.

REFERENCES

[1] R. Sedmak and H. Liebergot. Selective triple modular redundancy (STMR) based single-event upset (SEU) tolerant synthesis for FPGAs. *IEEE Transactions on Nuclear Science*, 51:2957-2969, 2005.

[2] J. M. Berger. A note on an error detection code for asymmetric channels. *Information and Control*, 4:68-73, 1961.

[3] D. Das and N. A. Touba. Synthesis of circuits withlow-cost concurrent error detection based on Bose-Lin codes. *In VLSI Test Symposium*, pages 309-315, 1998.

[4] K. Mohanram, E. Sogomonyan, M. Gossel, and N. Touba. *Synthesis of low-cost parity-based partially self-checking circuits*, 2003.

[5] S. Ghosh, N.A. Touba, and S. Basu. Synthesis of low power ced circuits based on parity codes. *In VLSI Test Symposium*, pages 315-320, 1-5 May 2005.

[6] R. Sharma and K.K. Saluja. An implementation and analysis of a concurrent built-in self-test technique. *In Digest of Papers Eighteenth International Symposium on Fault-Tolerant Computing FTCS-18*, pages 164- 169, June 1988.

[7] P. Drineas and Y. Makris. Concurrent fault detection in random combinational logic. *In Proceedings Fourth International Symposium on Quality Electronic Design ISQED*, pages 425-430, March 2003.

[8] Alves, N.; Shi, Y.; Dworak, J.; Bahar, R.I.; Nepal, K. "Enhancing online error detection through area-efficient multi-site implications", *IEEE 29th VLSI Test Symposium (VTS)*, 2011.

[9] Marc Boule, Jean-Samuel Chenard, and Zeljko Zilic. Assertion checkers in verification, silicon debug and infield diagnosis. *In Proceedings of the ISQED '07*.

[10] M Aarna, E Ivask, A Jutman, E Orasson, J Raik, R Ubar, V Vislogubov, HD Wuttke. Turbo Tester-Diagnostic Package for Research and Training. *The 1st East-West Design and Test Conference*, Alushta, 2003.

[11] Artur Jutman, A Peder, J Raik, M Tombak, R Ubar. Structurally synthesized binary decision diagrams. *6th International Workshop on Boolean Problems*. pp. 271-278, 2004.

[12] Raimund Ubar, Sergei Devadze, Jaan Raik, Artur Jutman. Parallel X-fault simulation with critical path tracing technique. *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*, pp. 879-884, 2010.

[13] J. Flich, J. Duato, Logic-Based Distributed Routing for NoCs, *IEEE Computer Architecture Letters*, Vol. 7, No. 1, January-June 2008.

[14] R. Parikh and V. Bertacco. Formally enhanced runtime verification to ensure NoC functional correctness. *In Proc. of the International Symposium on Microarchitecture (MICRO)*, 2011.

[15] Prodromou, A.; Panteli, A.; Nicopoulos, C.; Sazeides, Y., "NoCAlert: An On-Line and Real-Time Fault Detection Mechanism for Network-on-Chip Architectures," *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 60-71, 2012.

[16] Yu, Qiaoyan; Cano, J.; Flich, J.; Ampadu, P., "Transient and Permanent Error Control for High-End Multiprocessor Systems-on-Chip," *2012 Sixth IEEE/ACM International Symposium on Networks on Chip (NoCS)*, vol., no., pp.169,176, 9-11 May 2012.

[17] Alaghi, A.; Karimi, N.; Sedghi, M.; Navabi, Z., "Online NoC Switch Fault Detection and Diagnosis Using a High Level Fault Model," *22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems*, vol., no., pp.21,29, 26-28 Sept. 2007.

[18] Dalirsani, A.; Kochte, M.A.; Wunderlich, H.-J., "Area-efficient synthesis of fault-secure NoC switches," *IEEE 20th International On-Line Testing Symposium* (*IOLTS*), pp.13,18, 7-9 July 2014.

[19] J. Raik, V. Govind, R. Ubar. An External Test Approach for Network-on-a-Chip Switches. Proc. of the IEEE Asian Test Symposium, pp. 437-442, Nov. 2006

[20] J. Raik, V. Govind, R. Ubar. Design-for-Testability- Based External Test and Diagnosis of Mesh-like NoCs. *IET Computers and Digital Techniques*, Vol. 3, Issue 5, pp. 476-486, September 2009.

[21] Raik, J.; Govind, V. Low-area boundary BIST architecture for mesh-like network-on-chip, *IEEE 15th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, pp. 95-100, 2012.

APPENDIX E

B. Niazmand, S. P. Azad, J. Flich, J. Raik, G. Jervan and T. Hollstein, "Logic-based implementation of fault-tolerant routing in 3D network-on- chips," 2016 Tenth IEEE/ACM International Symposium on Networks- on-Chip (NOCS), Nara, Japan, 2016, pp. 1-8.

Logic-Based Implementation of Fault-Tolerant Routing in 3D Network-on-Chips

Behrad Niazmand*, Siavoosh Payandeh Azad*, José Flich†, Jaan Raik*, Gert Jervan*, Thomas Hollstein*‡
*Department of Computer Engineering,Tallinn University of Technology, Tallinn, Estonia
†Department of Computer Engineering, Universitat Politècnica de València,Valencia, Spain
‡ Informatik und Ingenieurwissenschaften, Frankfurt University of Applied Sciences, Frankfurt, Germany
{bniazmand, siavoosh, jaan, gert.jervan, thomas}@ati.ttu.ee, jflich@disca.upv.es, hollstein@fb2.fra-uas.de

***Abstract*—The susceptibility of on-chip communication links and on-chip routers to faults has guided the research towards focusing on fault-tolerance aspects of 2D and 3D Network-on-Chips (NoCs). In this paper, we propose Logic-Based Distributed Routing for 3D NoCs (LBDR3D), a scalable, re-configurable and fault-tolerant mechanism, which utilizes only two virtual channels for implementing any deadlock-free turn model routing algorithm in partially vertically connected 3D NoCs. Such networks might emerge either due to the limitation of on-chip area for vertical links or due to occurrence of fault because of wear-out. LBDR3D guarantees live-lock freeness as well as connectivity regardless of the location and number of vertical links as long as faults do not disconnect the network. Our method relies on a limited set of bits which describe the topology and routing algorithm, updated using an offline algorithm. Our Experimental results show the comparison of LBDR3D with three previously proposed fault-tolerant mechanisms, Elevator-First, North-East To Z (NETZ) and East-Then-West (ETW). Compared to Elevator-First, our proposed mechanism is more flexible and in terms of packet latency, it performs better or equal under even extreme fault scenarios for vertical links. Furthermore, as long as the topology is supported by the routing algorithm, LBDR3D can tolerate faults on horizontal links in each layer. In contrast to NETZ and ETW, LBDR3D does not rely on the location of vertical links as long as the network is connected.**

***Keywords*—fault-tolerance; routing algorithm; reconfigurability; reliability; Network-on-Chip.**

I. Introduction

Three-dimensional (3D) integration is one of the solutions that has gained momentum recently in order to alleviate the interconnect wire delay by stacking active silicon layers [1]. On the other hand, Network-on-Chip (NoC) is one of the trends that has been considered in research in order to overcome the communication bottleneck in previous architectures such as shared bus [2] [3], by providing scalability, flexibility, transparency and modularity.

In an on-chip network, processing cores communicate with each other on one layer and they might also need access to their memory blocks at the same time, therefore one approach can be placing the memory blocks on an adjacent layer in a 3D NoC architecture. Different research works have focused on the topic of 3D integration of NoCs by using stacked layers [1].

As the number of vertical links is reduced in a 3D NoC - thus, transforming them into vertically partially connected 3D NoCs [4] [5] - the utilization of the remaining vertical links increases, therefore creating a communication bottleneck. These missing vertical links can be either the result of faults, such as wear-out, or they can be related to saving area due to the on-chip area constraints. Therefore, in order to run an application on such NoCs, a routing mechanism that is both fault-tolerant and adaptive, would help to mitigate the issue by uniformly distributing packets on the communication links and bypassing the faulty links, while being re-configurable at the same time.

A scalable logic-based distributed routing mechanism, named LBDR, supporting turn model adaptive routing algorithms, has been proposed in [6] for 2D NoCs, however, the potential exists for the case of 3D NoCs with partially vertically connected links. In this paper, we extend the logic of LBDR to support 3D NoC topologies, and we name it LBDR3D. The mechanism is augmented with additional configuration bits, i.e. new connectivity bits for supporting 3D topologies and a new set of bits called as vertical bits. The proposed mechanism is scalable in a sense that, unlike some previous works (e.g. [7]), routers do not require to store the location address of nodes with vertical links in each layer. In addition, the approach does not incur additional overhead to packets when steering them across the layers of a 3D NoC.

LBDR3D gives the highest priority to the vertical links if the destination of a packet is not on the same layer as source, thus, steering it to the nearest node with vertical link, if needed. the nearest node is calculated using an offline algorithm (described later in Section III.B) and fed into the logic of the mechanism by setting the so called vertical bits using a re-configuration framework [8]. In case the destination is on the same layer as the source, any deadlock-free adaptive and deterministic turn model routing algorithm (already supported by LBDR [6]) can be implemented using LBDR3D. By using only two Virtual Channels (VCs) and not allowing packets to change their VC, the mechanism separates packets going upwards and downwards (a separate VC is used for each direction), thus guaranteeing deadlock freeness for cross-layer traffic flows. In addition, the mechanism ensures live-lock freeness using signals from the input ports (described in Section III.C), and guarantees connectivity as long as faults do not disconnect the network, therefor,e making it possible to tolerate scenarios with extreme faulty vertical links and faulty horizontal links (as long as LBDR supports the topology in each layer).

The rest of the paper is organized as follows: Section II reviews the previous works in the field of fault-tolerant routing algorithms for 3D NoCs. Section III is dedicated to

the description of the LBDR3D logic, along with the proposed offline algorithm for computing the vertical bits. Section IV is dedicated to proofs of deadlock and live-lock freeness of LBDR3D, along with a proof of providing connectivity for every source-destination pair. Section V is dedicated to the experimental results and comparison with three other fault-tolerant routing mechanisms for 3D NoCs in terms of performance (average latency), reliability, flexibility and area overhead. Finally, section VI concludes the paper.

II. Related Work

There has been a number of works proposing fault-tolerant routing algorithms for Network-on-Chips. In this paper, we have mostly focused on the state-of-the-art regarding fault-tolerant routing algorithms for 3D NoCs, however, our proposed mechanism also works for the 2D domain. Authors of [9] have introduced a fault-tolerant routing scheme in 3D NoCs, named 4NP-First, based on the Negative-First turn model. The drawback of the algorithm is the overhead of the packet replication if the number of faulty links in the network exceeds a threshold. In [10], a low-overhead fault-tolerant deflection routing algorithm is proposed for 3D Mesh NoCs. The limitation of this work is scalability due to using routing tables per layer. Authors of [11] have introduced AFRA, a deadlock-free and deterministic routing algorithm (based on an extension of ZXY routing algorithm) for 3D NoCs. However, the proposed algorithm has limitations regarding the location of faults occurring on vertical links. Ebrahimi et al. have proposed HamFA [12], which takes advantage of Hamiltonian paths in order to tolerate faults in 2D and 3D NoCs. Despite the advantages compared to [9] and [10], HamFA is not able to address faults on vertical links at the end of the Hamiltonian paths and also some of the horizontal links in each layer, as shown in [12].

Jiang et al. have presented an efficient fully adaptive fault-tolerant routing algorithm for 3D NoCs [13]. The algorithm consists of two phases: inter-layer and intra-layer routing. Two assumptions that limit this work are as follows: 1) Processing Elements (PEs) will never get faulty and 2) faults on links are considered as bidirectional. Also, the deadlock recovery mechanism used in this work can impose additional performance overhead. Authors of [14] have proposed a high-performance reliable and deadlock-free routing scheme (HARS), which follows a mid-node searching method in 3D NoCs without requiring any Virtual Channels (VCs). However, reliability results are only provided when up to 10% of the network vertical links are faulty.

In [4] a distributed routing algorithm has been proposed for partially vertically connected 3D NoCs, named Elevator-First. The algorithm can tolerate faults on vertical links, regardless of the location and the number of faulty links. In order to guarantee deadlock freeness, the method depends on using two virtual channels along X and Y dimensions. Despite the advantages, the algorithm relies on an additional overhead in header flits, when steering packets to nodes with vertical links (called as elevator nodes). Also, each router should store the location of at least one up and one down elevator node in its layer for fault-tolerance purposes which can impose additional memory overhead as the network scales up.

The mechanism proposed in this work is based on an extension of Logic-Based Distributed Routing (LBDR) [6] to the 3D domain. LBDR is capable of implementing different deadlock free turn model routing algorithms and depends only on a set of routing and connectivity bits that describe the routing function and the topology, respectively. The mechanism removes the need for routing tables at routers, thus being scalable. LBDR is able to support 2D Mesh and topologies derived from the 2D Mesh (as shown in [6]), however, it lacks the support for 3D NoC topologies. In [15], σLBDR is introduced which is a congestion-aware version of LBDR, able to take routing decisions based on the traffic status of links connecting each router to its neighbors, however, it only supports 2D topologies and support for the 3D is remained as future work. On the other hand, LBDR3D uses the concept of inter-layer and intra-layer routing, depending on the location of packet's destination with respect to the source node. The proposed mechanism relies on 2 Virtual Channels (VCs) per input port at each router. Based on whether the packet should be steered upwards or downwards, a fixed VC is assigned to it at the source node and therefore, up-going and down-going packets are separated until they reach their corresponding destinations, which can guarantee the prevention of deadlock even when different data flows are transferred across different layers of a 3D NoC.

The main contributions of this work are the following: 1) LBDR3D, A logic-based routing mechanism which supports implementing dead-lock free routing algorithms in partially vertically connected 3D Network-on-Chips, 2) an offline algorithm integrated with a framework (OSR-Lite [8]) for calculating and updating the new set of configuration bits (*i.e.* vertical bits) of LBDR3D logic, 3) providing proof of dead-lock freeness, by using only two VCs at each router and utilization of input signals for prevention of live-lock, while at the same time guaranteeing connectivity between all communicating nodes. 4) Tolerating extreme cases of faulty vertical links, as long as faults on vertical links do not disconnected the network, thus not being dependent on the number and location of the faulty vertical links, unlike [5] [16]. As it will be shown in Section 5, LBDR3D is also capable of tolerating faults on horizontal links in each layer, as long as the topology of each layer is supported by the original LBDR mechanism.

III. Description of LBDR3D Mechanism

In this section, first the LBDR3D mechanism and its logic are described. Afterwards, the proposed offline algorithm which is used for calculating the vertical bits, is explained in detail.

A. LBDR3D Mechanism

Our proposed mechanism is based on LBDR, which supports 2D Mesh and some of the topologies derived from 2D Mesh [6]. However, in order to add support for 3D NoCs, the connectivity bits (C_x) of the logic are extended to cover Up and Down directions, in addition to the existing ones for 4 cardinal 2D directions (North, East, West and South), therefore, leading to six connectivity bits per router, as follows:

$$\boldsymbol{C_x : C_n, C_e, C_w, C_s, C_u, C_d}$$

LBDR3D uses the same number of routing bits (R_xy) as LBDR for implementing the routing algorithm in each layer, as follows:

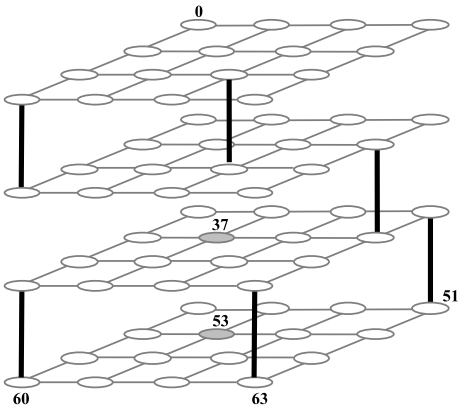


Fig. 1: A $4 \times 4 \times 4$ 3D Mesh with 88% faulty vertical links

$$\boldsymbol{R_{xy} : R_{ne}, R_{nw}, R_{en}, R_{es}, R_{wn}, R_{ws}, R_{se}, R_{sw}}$$

One of the new additions to the mechanism is a new set consisting of 8 bits per router, named as *vertical bits*, based on which the logic can determine whether there is at least one node with up/down vertical link in the corresponding direction or not (4 bits for up and 4 bits for down links). The vertical bits are as follows:

$$\boldsymbol{Nu, Eu, Wu, Su, Nd, Ed, Wd, Sd}$$

The bits ending with u indicate that there is at least one vertical node with up link in the corresponding direction. The same applies to the bits ending with d, but for down links. For instance, if the S_u bit is set, it means that there exists at least one vertical node with up link in the Southward direction in the current layer. In order to cover the situations in which the vertical node is located on a quadrant with respect to the current node, both the corresponding bits are set. For instance, if a router has a node on the North-East quadrant with the up vertical link, both Nu and Eu bits at the current router are set. Such approach for showing the existence of vertical nodes would save area, as there would not be any necessity to store the location (address) of the node(s) with vertical links (unlike the methods used in [4] and [5]).

One important issue is the approach taken to compute the values of the vertical bits, which is addressed in the next subsection via the proposed offline algorithm that calculates the vertical bits at each router at the same time when connectivity and routing bits are initialized.The re-configuration process is performed using the OSR-Lite framework [8] in a transparent way, without imposing significant run-time latency and affecting normal operation of the network.

B. Offline algorithm for calculating vertical bits

In order to select a node (router) as a vertical node in a 3D NoC for steering a packet upwards or downwards (if the layer of the destination node is not the same as the current node), a policy is required for prioritization of the direction to take. As shown in Fig. 1, for example, in a $4 \times 4 \times 4$ 3D Mesh with

Algorithm 1: Offline algorithm for calculating vertical bits at routers in each layer

```
1  forall the nodes router[curr] ∈ NoC do
2      DVL = ∅ // Down Vertical Nodes List
3      UVL = ∅ // Up Vertical Nodes List
4      HTV = 0 // hops to Vertical node
5      if Cu[router[curr]] = 1 then
6          SetVerticalBits("U", router[curr], 0)
7      else
8          forall the router[i] ∈ Layer[curr] do
9              if router[i] ≠ router[curr] then
                   // for each router[i] in current
                      layer except the current node
10                 if Cu [router[i]] = 1 then
                       // router[i] is an up vertical
                          node
11                     HTV = Distance (router[i],router[curr])
12                     append HTV to UVL
13         SetVerticalBits("U",router[curr],FindMin(UVL))
14     if Cd[router[curr]] = 1 then
15         SetVerticalBits("D", router[curr], 0)
16     else
17         forall the router[i] ∈ Layer[curr] do
18             if router[i] ≠ router[curr] then
                   // for each router[i] in current
                      layer except the current node
19                 if Cd [router[i]] = 1 then
                       // router[i] is a down vertical
                          node
20                     HTV = Distance (router[i],router[curr])
21                     append HTV to DVL
22         SetVerticalBits("D",router[curr],FindMin(DVL))
   // Functions Description:
   // 1. Distance (router[i],router[curr]) returns
      |Δx + Δy|= |x_curr − x_router[i]|+ |y_curr − y_router[i]|
   // 2. FindMin(VerticalNodesSet) returns the
      vertical node in set VerticalNodesSet with
      shortest distance to current node,ties are
      broken randomly
   // 3. SetVerticalBits(Dir,router[curr],VNode)
      sets the vertical bits of router[curr] in
      direction Dir("U"=up,"D"=down), to vertical
      node VNode
```

88% faulty vertical links, if node 53 wants to send a packet to node 37, it has 3 choices for choosing a node on the current layer as an up vertical node (nodes 51, 60 and 63). As it can be seen, we can not guarantee that the total path the packet will take to reach its destination will be the minimal path. Therefore, instead of trying to take the minimal possible path from source to destination, the offline algorithm calculates the values of vertical bits at each router based on its Manhattan distance to a vertical node. As it can be seen in Fig. 1, two nodes can be chosen as candidates for up vertical nodes (nodes 51 and 60), values of vertical bits for the up direction for node 53 would be as follows (calculated by the offline algorithm):

$$\boldsymbol{Nu = 0, Eu = 0, Wu = 1, Su = 1}$$

Also, since node 53 is located on the bottom-most layer, all the down vertical bits for this node would be set to zero, as follows:

$$\boldsymbol{Nu = 0, Eu = 0, Wu = 0, Su = 0}$$

The pseudo-code for the algorithm that computes the

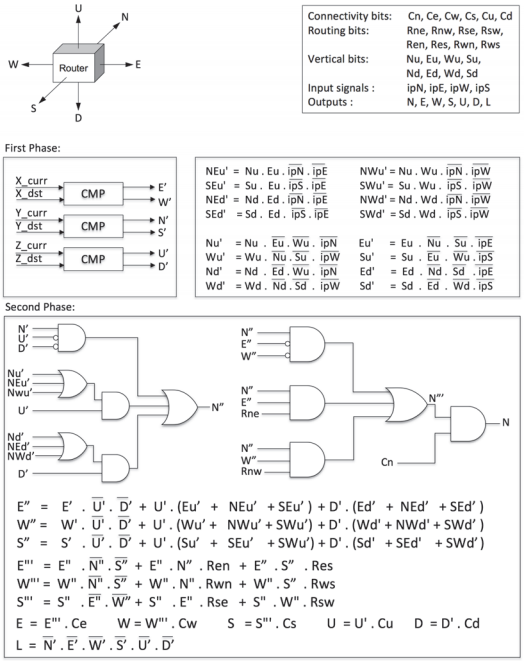


Fig. 2: Proposed logic of LBDR3D mechanism

vertical bits for each router is summarized in Algorithm 1. These calculations are performed once for up and once for down vertical nodes. As mentioned before, the algorithm is executed offline and before the normal operation of the network, and afterwards, the values of vertical bits are re-configured (if necessary) using the OSR-Lite framework [8]. Thereafter, the algorithm will only be executed if a new fault occurs in the network and there is a need for re-configuration of the connectivity, routing and vertical bits. How the faults are detected on the links would be out of the scope of this paper. It is worth noting that if a node is an up (down) vertical node itself, all the up (down) vertical bits for that router are set to zero, as the current node can already be chosen by LBDR3D for steering the packets one layer up (down).

C. Description of LBDR3D Logic

The logic of LBDR3D is proposed based on the principle that packets should be steered towards a node with vertical link, making the packet getting closer to its destination (if possible), but it should not wander between different nodes with vertical links in one layer, since in that case, it can lead to live-lock and affect performance. The complete logic of LBDR3D mechanism is shown in Fig. 2. In the first phase, the direction signals are computed by comparing the current address of the packet (stored in the current router) and the destination address of the packet (extracted from the header flit of the packet), i.e. signals N', E', W', S', U' and D' are computed. Also, in this phase, first the quadrants or directions that the packet cannot go are filtered out temporarily for the packet. In order to prevent a packet from fluctuating between two vertical nodes (which guarantees live-lock freeness), we have utilized four additional signals which are fed from the 2D input ports, as follows:

$$\boldsymbol{ipX : ipN, ipE, ipW, ipS}$$

As an example, if a packet comes from the North input port, ipN signal is set to one. As the packet should not go back to the North direction again, it should not be possible for the packet to be steered towards North (N) direction in search of a vertical link. Next, the directions that the packet may take, are computed, that means the packet is transmitted on the plane using any kind of deadlock free turn model routing algorithm that can already be implemented using LBDR on a 2D NoC. In order to explain the logic of LBDR3D, we focus on one output port, for instance the North (N) output port logic. Similar deductions can be inferred for other output port signals. For the North port to be selected for forwarding the packet, one of the following conditions can hold: (1) The packet's destination is located on the same layer as the current node and it is located towards the North direction ($N'.U'.D'$), or (2) The current node is not a vertical node, but there exists at least one up/down vertical node on the same layer as the current node towards the North direction (i.e. on North direction or on North-East or North-West quadrant):
$U'.(Nu' + NEu' + NWu') + D'.(Nd' + NEd' + NWd')$

The second phase of the logic of LBDR3D is similar to the basic LBDR. For instance, in case of the North output logic, if one of the above-mentioned conditions hold, the North output port can be selected if either (1) the destination is located on the same column as the current node in the North direction or (2) it is located on the North-East (NE) or (3) North-West (NW) quadrant and the turn at the next router along North direction allows the packet to take the North to East ($R_{ne} = 1$) or North to West turn ($R_{nw} = 14$), respectively. Finally, for the North port (N) to be considered as the output port for transmitting the packet, the corresponding connectivity bit of North port (C_n) should also be set. Therefore, in the end, the packet will be forwarded to the North output port (if North is also chosen by the arbitration unit) and it will either reach its final destination (if destination is on the same layer as current node) or it will reach the nearest node with up/down vertical link, depending on whether it has to go upwards or downwards (when destination is not on the same layer as current node).

The only output port signals that have slightly different logics are U (Up) and D (Down) and the L (Local) output port signals. If a packet reaches a vertical node and has to be steered upwards or downwards, only U or D output port can become active, respectively, and other output port signals are automatically set to zero (based on the logic's behavior and because the offline algorithm will calculate all the corresponding vertical bits as zero for a node with vertical link). It should be noted that depending on the topology, and based on the nearest vertical node, the vertical bits for a node might change during the lifetime of the system, if re-configuration would be needed.

Also, regarding the Local output port (L), it is activated only when the packet has reached its destination (all the direction signals N', E', W', S', U' and D' are zero). In such case, the packet is forwarded to the Processing Element (PE)

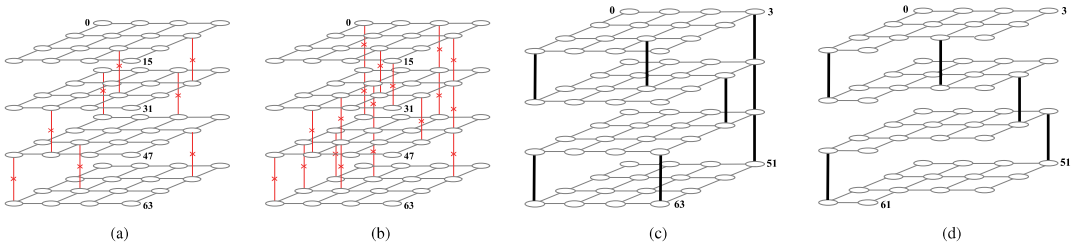


Fig. 3: Different topology scenarios considered for analysis of routing mechanisms: with (a) 20%, (b) 40%, (c) 84% faulty vertical links, and (d) 88% faulty vertical links with some faulty horizontal links for a $4 \times 4 \times 4$ 3D Mesh. It should be noted that in figures (a) and (b), the red vertical links are the faulty and the vertical links that are not shown are the healthy ones. However, in figure (c) and (d) only the healthy vertical links are shown for the sake of figure's simplicity and the faulty ones are not shown.

connected to the router through its Local port.

IV. Deadlock and Live-lock Freeness and Connectivity Guarantee

A. Proof of deadlock freeness

In this section, a formal proof is provided regarding the fact that any deadlock free routing algorithm that is implemented using LBDR3D in each layer of a 3D NoC, would not lead to deadlock when packets are transmitted across the layers. To this end, similar to [4], virtual channels are introduced in this work.

A deadlock situation can occur whenever nodes in the network try to access resources (communication links and/or buffers) in a circular way, which can cause the packets to get locked and not advance anymore. In 3D NoCs, Intra-layer communications might still lead to deadlock situation even if each layer already has a deadlock free routing algorithm. The proof of deadlock freeness for LBDR3D is as follows, using only two Virtual Channels (VCs) per each input port and the concept of Channel Dependency Graph (CDG) [17], [18]. A CDG is a directed graph whose nodes are network channels and edges represent dependencies between these channels. If we assume that a cycle exists in the CDG, the cycle can either (1) include vertical links at least one in each direction (up or down), or (2) not include any vertical links. If (2) is the case, then the cycle cannot be formed since all the packets follow the deadlock-free CDG-acyclic routing algorithm on the plane of the supposed cycle (as mentioned before, the routing algorithm used in each layer is deadlock-free). If (1) is the case, then there is at least one link in the up direction and one in the down direction. These two links have no dependency through the cycle since one is using VC1 (for up direction) and the other is using VC0 (for down direction). In that case, we need to demonstrate that there are no dependencies from the down resource (VC0) to the up resource (VC1). This would mean there is one layer where there is a dependency between link X of VC1 and link Y of VC0. Since we do not allow VC0 to VC1 and VC1 to VC0 transitions, therefore the cycle can not form, thus the proof.

B. Proof of live-lock freeness

A live-lock situation can occur when packets usually take non-minimal paths and never reach their destination. However, as mentioned in Section III, LBDR3D uses a set of signals from input ports when computing the candidate output port(s) for routing packets and it does not allow a packet that has come from a direction going back to the same direction (input signals are denoted as ipN, ipE, ipW and ipS, each corresponding to one of the 4 main cardinal 2D directions). Therefore, the logic automatically guarantees packets would not fluctuate between different nodes with vertical links when they are being routed to another layer in a 3D NoC, thus, it guarantees live-lock freeness.

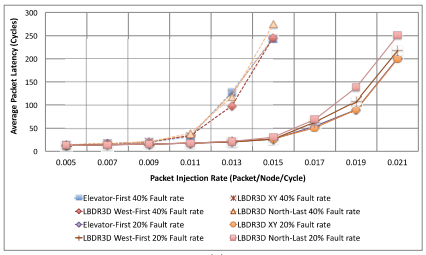
C. Proof of connectivity

Similar to the Original LBDR, the proposed mechanism also guarantees connectivity between every pair of communication nodes as long as faults do not disconnect the network. In other words, as long as for each source, at least one minimal path exists to its destination under the current routing algorithm, connectivity is assured. Our work outperforms [5] and [16] in a sense that there are no limitations on the location of vertical links in each layer for guaranteeing connectivity and there is no need for the existence of a pillar connecting all layers.

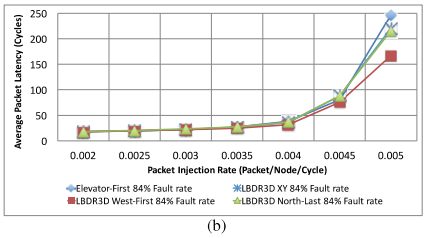
V. Experimental Results

A. Performance analysis

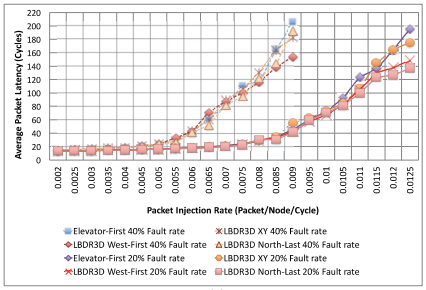
In order to analyze the performance of LBDR3D, we have compared it with Elevator-First [4]. Both approaches are implemented in an extended cycle-accurate NoC simulator supporting 3D NoCs, i.e. Noxim simulator [19]. The extended version of Noxim supporting Virtual Channels (VC) and LBDR3D is maintained as an open-source project on [20]. In order to evaluate the effect of faulty links on the performance of the network using different routing mechanisms, we considered different scenarios for a $4 \times 4 \times 4$ 3D NoC with 20%, 40%, 84% faulty vertical links and 88% faulty vertical links with some faulty horizontal links, as shown in Fig. 3a-d. The location of faulty vertical links are chosen randomly in the scenarios. However, it is noteworthy that LBDR3D is not limited only to these scenarios. The two other fault-tolerant routing mechanisms are North-East to Z (NETZ) [16] and East-Then-West (ETW) [5], [21]. However, since NETZ relies on the existence of a pillar on the North-East column



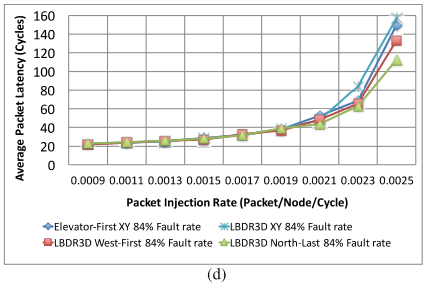
(a)



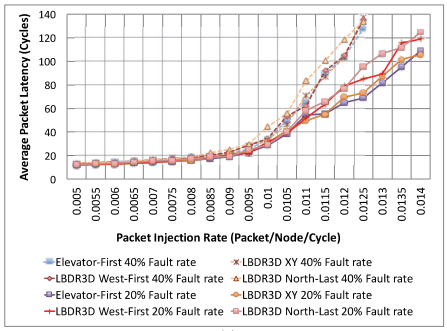
(b)



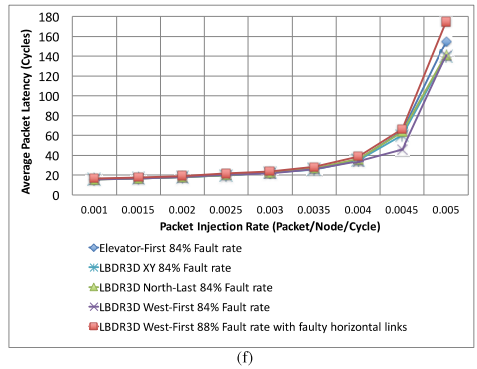
(c)



(d)



(e)



(f)

Fig. 4: Average latency results for different fault scenarios under (a),(b) Random Uniform, (c),(d) Transpose and (e)(f) Bit-Reversal traffic patterns

of the network and also ETW relies on the existence of at least one vertical link on the East side of each layer, they are excluded from the performance analysis results (we have assumed random locations for faulty vertical links).

Fig. 4a-f illustrate the simulation results of average packet latency for the compared algorithms, in case of 20%, 40%, 84% faulty vertical links(the percentages correspond to the fault rates written in the figure). The scenarios represent the ones shown in Fig. 3a-c. In addition, in order to show that LBDR3D supports faults on some horizontal links in addition to vertical ones, simulation results for one scneario is provided in Fig. 4f which corresponds to the topology scenario in Fig. 3d (88% faulty vertical links with some faulty horizontal links). Three different synthetic traffic patterns, i.e. Random Uniform, Bit-Reversal and Transpose are considered in the simulation in order to assess the effect of different fault scenarios on the average network latency and analyzing network's saturation point for latency. The simulation setup parameters (for Noxim simulator) and the considered scenarios for the experiments are summarized in Table I. It should be noted that during the experiments, each simulation is performed 10 times for each packet injection rate (pir) and the average value of the latency is considered.

As it can be seen, in Fig. 4a, 4c, 4e, under 20% and 40% random faulty vertical link configurations (which correspond to the scenarios shown in Fig. 3a and Fig. 3b, respectively), LBDR3D is able to achieve slightly better average latency results compared to Elevator-First, under Random Uniform (Fig. 4a), Transpose (Fig. 4c) and Bit Reversal (Fig. 4e) traffic patterns. Of course, as it can be observed, when the fault rate increases to 40%, the network starts saturating at a lower packet injection rate (pir) compared to the 20% fault rate case. In addition, when LBDR3D is programmed to a partially adaptive routing algorithm such as West-First and North-Last in each layer (as shown in Fig. 4a, 4c and 4e), under all considered traffic scenarios except random uniform, it can achieve better or similar performance (average latency) results due to lowering the chance of congestion in the network, compared to deterministic routing (Elevator-First and LBDR3D-XY). This also conforms to the observation made

TABLE I: Considered Scenarios and simulation parameters

Routing Algorithm	LBDR3D-XY,Elevator-First XY, LBDR3D-West First, LBDR3D-North Last
Network Topology	$4 \times 4 \times 4$ 3D Mesh with 20%,40%, 84% faulty vertical links and 88% faulty vertical links with some faulty horizontal links
Number of VCs	2 (per each router input port)
VC depth	4 flits
Network Frequency	1 Ghz
Simulation Time	10000 cycles (1 cycle = 1ns)
Warm-up time	1000 cycles (1 cycle = 1ns)
Traffic patterns	Random Uniform, Bit-Reversal and Transpose

in [22] which illustrates that deterministic dimension-ordered routing algorithms such as XY achieve better latency results compared to the adaptive routing algorithms as they distribute the traffic in a more uniform manner in the long run.

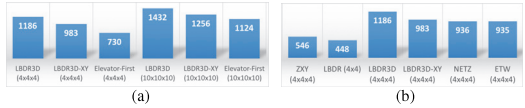
Furthermore, as demonstrated in Fig. 4b, 4d and 4f, as the percentage of faulty vertical links increases to 84%, LBDR3D still achieves comparable average latency results to Elevator-First under the three considered synthetic traffic patterns. In addition, Fig, 4b, 4d and 4f confirm that as more links get faulty, there would be less path diversity for packets being transmitted between layers and therefore the network saturates at a lower packet injection rate. This is also due to higher utilization of vertical links for cross-layer traffic flows.

It is noteworthy that all the scenarios demonstrated in Fig. 3 can be simulated using the configuration files (for Noxim) provided available at [20], [23] for all the considered traffic patterns and the packet injection rates shown in Fig. 4.

B. Flexibility and scalability analysis

In order to evaluate the flexibility and scalability of LBDR3D, it is compared with Elevator-First. As mentioned earlier, both LBDR3D and Elevator-First follow the concept of inter-layer and intra-layer routing when the desintation of a packet is not on the same layer as the current node. However, unlike Elevator-First [4], in our approach, the packets are not augmented with additional intermediate destination address of the elevator nodes. Moreover, we remove the need for storing the location address of elevator node(s) per each router at the current layer, and substitute it by a fixed limited set of vertical bits, thus making LBDR3D scalable (more details explained in Section V-C).

From another point of view, in [4] only the deterministic XY routing is considered and for simplicity, the topology of each layer is assumed to be 2D Mesh. Thus, one might infer that the routing logic of Elevator-First is fixed, therefore, if a fault occurs in the network and the topology changes or the routing algorithm needs to be re-configured, the whole routing logic needs to be re-designed, and only the addresses of the elevator nodes can be updated. However, in case of LBDR3D, if a fault occurs, during the re-configuration phase, not only the vertical bits, but also the connectivity and routing bits can be updated, thus, in addition to add support for the new faulty topology, the routing algorithm can also be updated. As an example, Elevator-First with XY routing would not be able to support the topology shown in Fig. 3d, however LBDR3D when programmed to West-First routing, can still guarantee connectivity among all nodes and tolerate faults on horizontal links to some extent. Also, for such scenario, LBDR3D can obtain acceptable performance (average latency) results compared to the other scenarios, as shown in Fig. 4f (under Bit-Reversal traffic pattern).



(a) (b)

Fig. 5: Area consumption (in μm^2) for different routing mechanisms, (a) for showing scalability of LBDR3D over Elevator-First, (b) for comparison of LBDR3D to ZXY, LBDR, NETZ and ETW.

C. Area consumption analysis

In our experiments, we also made an area consumption comparison, by describing the RTL logic of LBDR3D for a $4 \times 4 \times 4$ and $10 \times 10 \times 10$ 3D Mesh and also the logic of ZXY routing (which is a non-fault-tolerant mechanism), Elevator-First, NETZ and ETW, and synthesized them using Synopsys Design Compiler [24]. The results showed 34.6% increase in area for a $4 \times 4 \times 4$ 3D Mesh and 11.7% increment in area for a $10 \times 10 \times 10$ 3D Mesh, when comparing LBDR3D logic to Elevator-First using XY routing. The decrease in the area overhead can be a proof of the scalability of LBDR3D as it does not store the location of nodes with vertical links in each layer. In addition, when comparing LBDR3D programmed to XY routing (LBDR3D XY) with NETZ and ETW for the case of a $4 \times 4 \times 4$ 3D Mesh network, the area overhead was only around 5.02% and 5.1%, respectively. Another explanation for the area overhead of LBDR3D compared to ZXY, LBDR, Elevator-First, NETZ and ETW would be the additional set of vertical bits and the new logic for supporting 3D NoC topologies, but at the same time it brings the advantage of providing flexibility and not relying on existence of any pillars in the topology.

The area consumption results are illustrated in Fig. 5. For better readability, two groups are considered. In Fig. 5a LBDR3D is compared only to Elevator-First (both the general logic and also when programmed to XY routing). Also, its scalability comparison is performed in Fig. 5a when the size of the network grows. Our experiments showed that after the network size of $25 \times 25 \times 25$, LBDR3D can even achieve better area results, as it does not rely on the location address of elevator nodes. The second group, which is shown in Fig. 5b, includes comparison of the ZXY routing, LBDR3D and LBDR3D XY, NETZ and ETW, all for a $4 \times 4 \times 4$ network. It is noteworthy that the area results are obtained using NanGate Open Cell 45 nm Library [25] and synthesis of the RTL of the designs is performed using Synopsys Design Compiler [24].

D. Reliability analysis

In terms of reliability and tolerance to faults on vertical links, we have compared LBDR3D with two other proposed approaches in [5], [21] and [16], named East-Then-West (ETW) and North-East To Z (NETZ). Both methods rely on the existence of a pillar, *i.e.* NETZ requires the existence of at least one pillar in the North-East column of the network

and ETW also needs at least one vertical link on the East column of the network. However, our proposed mechanism does not impose such limitations. For instance, as mentioned before, LBDR3D supports all the topologies shown in Fig. 1 and Fig. 3a-d. In case of faults on horizontal links, LBDR3D and Elevator-First, both support all the topologies shown in Fig. 1 and Fig. 3a-c. But in the case of Fig. 3d, since the routing algorithm of Elevator-First in each layer is XY, the topology is not supported as some of the minimal paths between the nodes are broken. However, in such topology (Fig. 3d) LBDR3D can be re-programmed, for instance, to the West-First turn model.

VI. Conclusion

In this paper, LBDR3D, a fault-tolerant, deadlock and live-lock free logic-based distributed mechanism for implementing routing algorithms in 3D NoCs was introduced. The mechanism, which is an extension to the baseline LBDR approach, only relies on a fixed set of configuration bits describing the routing algorithm and the topology. In addition, an offline algorithm was utilized in order to calculate the new set of vertical bits for the proposed mechanism, re-configured in transparent way using a previously introduced framework. The proposed mechanism is able to tolerate extreme fault scenarios for vertical links in a 3D NoC as long as the network is not disconnected. Moreover, faults on horizontal links are tolerated, as long as the topology in each layer is already supported by LBDR. Experimental results, compared LBDR3D with three other proposed fault-tolerant routing algorithms for vertically partially connected 3D NoCs. It was shown that it can achieve similar performance results under synthetic traffic patterns compared to the Elevator-First routing algorithm, without storing the location of elevator nodes and not augmenting packets with additional information. In addition, our proposal does not rely on the existence of pillars in the network, unlike NETZ and ETW. Finally, area consumption results showed the scalability of the mechanism compared to Elevator-First and its insignificant overhead compared to NETZ and ETW, which comes at the price of more flexibility.

Acknowledgments

The work has been supported by EU FP7 STREP BASTION, EU's H2020 RIA IMMORTAL, EU's H2020 Twinning TUTORIAL, Estonian Science Foundation grant ETF9429, Estonian institutional research grant IUT 19-1, and funded by Estonian Ministry of Education and Research.

References

[1] J. Flich and D. Bertozzi, Eds., *Designing network on-chip architectures in the nanoscale era*, ser. Chapman & Hall/CRC computational science series. Boca Raton, FL: Chapman and Hall/CRC, 2011. [Online]. Available: http://opac.inria.fr/record=b1132571

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

[3] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," in *Design, Automation and Test in Europe Conference and Exhibition 2000. Proceedings*, 2000, pp. 250–256.

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

[5] R. Salamat, M. Ebrahimi, and N. Bagherzadeh, "An adaptive, low restrictive and fault resilient routing algorithm for 3d network-on-chip," in *Parallel, Distributed and Network-Based Processing (PDP), 2015 23rd Euromicro International Conference on*, March 2015, pp. 392–395.

[6] J. Flich and J. Duato, "Logic-based distributed routing for nocs," *Computer Architecture Letters*, vol. 7, no. 1, pp. 13–16, Jan 2008.

[7] M. Bahmani, "Architectural exploration and performance analysis of Vertically-Partially-Connected Mesh-based 3D-NoC," Theses, Université de Grenoble, Dec. 2013. [Online]. Available: https://tel.archives-ouvertes.fr/tel-01070020

[8] A. Strano, D. Bertozzi, F. Trivino, J. Sanchez, F. Alfaro, and J. Flich, "Osr-lite: Fast and deadlock-free noc reconfiguration framework," in *Embedded Computer Systems (SAMOS), 2012 International Conference on*, July 2012, pp. 86–95.

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

[10] C. Feng, M. Zhang, J. Li, J. Jiang, Z. Lu, and A. Jantsch, "A low-overhead fault-aware deflection routing algorithm for 3d network-on-chip," in *VLSI (ISVLSI), 2011 IEEE Computer Society Annual Symposium on*, July 2011, pp. 19–24.

[11] S. Akbari, A. Shafiee, M. Fathy, and R. Berangi, "Afra: A low cost high performance reliable routing for 3d mesh nocs," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, March 2012, pp. 332–337.

[12] M. Ebrahimi, M. Daneshtalab, and J. Plosila, "Fault-tolerant routing algorithm for 3d noc using hamiltonian path strategy," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, March 2013, pp. 1601–1604.

[13] X. Jiang and T. Watanabe, "A novel fully adaptive fault-tolerant routing algorithm for 3d network-on-chip," in *TENCON 2013 - 2013 IEEE Region 10 Conference (31194)*, Oct 2013, pp. 1–4.

[14] J. Zhou, H. Li, Y. Fang, T. Wang, Y. Cheng, and X. Li, "Hars: A high-performance reliable routing scheme for 3d nocs," in *VLSI (ISVLSI), 2014 IEEE Computer Society Annual Symposium on*, July 2014, pp. 392–397.

[15] N. Gupta, M. Kumar, V. Laxmi, M. Gaur, and M. Zwolinski, "σlbdr: Congestion-aware logic based distributed routing for 2d noc," in *VLSI Design and Test (VDAT), 2015 19th International Symposium on*, June 2015, pp. 1–6.

[16] H. Ying, K. Hofmann, and T. Hollstein, "Dynamic quadrant partitioning adaptive routing algorithm for irregular reduced vertical link density topology 3-dimensional network-on-chips," in *High Performance Computing Simulation (HPCS), 2014 International Conference on*, July 2014, pp. 516–522.

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

[18] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 12, pp. 1320–1331, Dec 1993.

[19] S. M. M. P. Vincenzo Catania, Andrea Mineo and D. Patti. (2015, jun) Noxim - the noc simulator. [Online]. Available: https://github.com/davidepatti/noxim

[20] Noxim with LBDR3D support. [Online]. Available: https://github.com/bniazmand/Noxim-LBDR3D

[21] R. Salamat, M. Khayambashi, M. Ebrahimi, and N. Bagherzadeh, "A resilient routing algorithm with formal reliability analysis for partially connected 3d-nocs," *IEEE Transactions on Computers*, vol. PP, no. 99, pp. 1–1, 2016.

[22] G. Ascia, V. Catania, M. Palesi, and D. Patti, "Neighbors-on-path: A new selection strategy for on-chip networks," in *2006 IEEE/ACM/IFIP Workshop on Embedded Systems for Real Time Multimedia*, Oct 2006, pp. 79–84.

[23] Noxim with Elevator-First support. [Online]. Available: https://github.com/bniazmand/Noxim_Elevator_first

[24] Synopsys design compiler. [Online]. Available: http://www.synopsys.com/

[25] Nangate, inc. nangate 45nm open cell library. [Online]. Available: http://www.nangate.com/?page id=2325, 2008

APPENDIX F

S. P. Azad, B. Niazmand, A. K. Sandhu, J. Raik, G. Jervan and T. Hollstein, "Automated area and coverage optimization of minimal latency checkers," 2017 22nd IEEE European Test Symposium (ETS), Limassol, 2017, pp. 1-2.

Automated Area and Coverage Optimization of Minimal Latency Checkers

Siavoosh Payandeh Azad*, Behrad Niazmand*, Apneet Kaur Sandhu*, Jaan Raik*, Gert Jervan*, Thomas Hollstein*†
Department of Computer Engineering
*Tallinn University of Technology, † Frankfurt University of Applied Sciences
Email: {siavoosh, bniazmand, akaur, jaan.raik, gert.jervan, thomas}@ati.ttu.ee

***Abstract*—With the scaling of silicon technology beyond the sub-micron domain, the probability of the system being exposed to different sources of faults increases. Manifestation of new defects during system's run-time, necessitates the need for a mechanism providing cost-effective online fault detection which performs concurrently with the circuit's normal operation and has low area overhead and high fault coverage. Especially crucial is the fault detection latency, as the system's ability to isolate faults and recover from them is highly dependent on the detection time. This paper proposes two heuristics (branch-and-bound and greedy) for minimization of concurrent online checkers. Both algorithms use the concept of dominant checkers, proposed in this work. The method allows generating minimal area checkers satisfying a target fault coverage with the shortest possible fault detection latency. Experimental results demonstrate the area efficiency of the approach compared to other methods.**

I. Introduction

Miniaturization of nano-electronic technology increases the vulnerability of components towards wear-out and environmental effects. Thus, concurrent online checkers for detecting faults during circuit's life-time are a must in modern reliable systems. In this paper, we propose an approach for evaluation and minimization of concurrent online checkers using two different heuristics: branch-and-bound, and greedy. Greedy heuristic scales well with the growing number of checkers while branch-and-bound provides an exact solution. Both algorithms utilize the concept of *dominant checkers*, proposed in this work, in order to speed up the heuristics. The method allows generating minimal area checkers satisfying a target fault coverage with the shortest possible fault detection latency.

II. Concurrent Online Checkers Concept

In this work, the concept of concurrent online checkers, introduced in [1], has been utilized, and single stuck-at fault model has been used. When applying a set of valid input stimuli to the circuit under check, four different conditions may occur, which are named as True Detection, True Miss, False Positive and Benign Miss. Checkers' Efficient Index (CEI) [2] has been used as a metric for evaluating the checkers for minimization heuristics.

III. Checkers Evaluation and Minimization Flow

The proposed flow for evaluation and minimization of the checkers is demonstrated in Fig. 1. Details of the framework's flowchart has been explained in [2]. An important part of the framework is to minimize the area of the checkers (by trading-off with fault localization capability) while having 100% CEI.

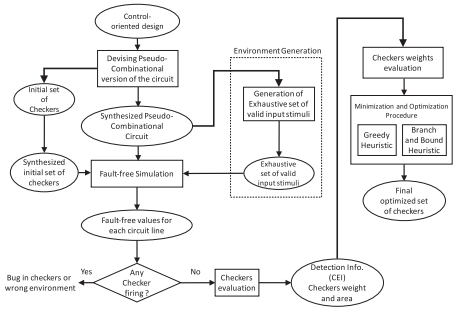


Fig. 1: Checkers Evaluation and Minimization Framework Flowchart

In this work, the minimization part of the flow is equipped with two new heuristics, Branch-and-Bound and Greedy.

It is worth noting that this work focuses on combinational checkers for *control-oriented* circuits. Regarding the data-path, it is assumed to be already protected by an error detection/correction technique. As an example in our experiments, we have used the control part of a NoC router as the circuit under check. However, the idea of checkers is not limited to a specific control-oriented circuit and it can be applied to any arbitrary one.

IV. Checkers' Minimization Heuristics

In this work, two heuristics are used for the minimization and optimization part of the flow: greedy and branch-and-bound. Greedy algorithm uses Checkers' Efficiency Index (CEI) for sorting the checkers and then tries picking them from the top of the list and calculates CEI and checks for the area feasibility of the solution. Also, a depth First Search (DFS) implementation of *Branch-and-Bound* algorithm was implemented for checker minimization. At each step, a checker is chosen (to be taken or being discarded), and the CEI and area of the selected checkers are estimated. Also, optimistic evaluation of sub-tree below the chosen option is estimated (which is the CEI of all the remaining checkers without considering the area constraint). Branch-and-bound algorithm provides exact solution for any set of checkers. In our experiments, two techniques are used to improve the result quality:

TABLE I: The result of checkers optimization without application of area constraints and comparison with DMR and TMR of the control circuit.

Unit Information									
Control Unit	Control unit area		Control unit DMR		Control unit TMR		Complete set of checkers		
	Full unit	pseudo comb.	Area	Overhead (%)	Area	Overhead (%)	Number	Area	Overhead (%)
Routing Logic	77	39	91	67.5%	153	148%	18	123	159%
Arbiter	174	124	209	48.8%	464	195%	56	337	193%
FIFO	129	60	133	56.5%	235	135.6%	13	125	96.8%

Optimization Results									
Control Unit	Opt. checkers without Dominant checkers				Control Unit	Opt. checkers with Dominant checkers			
	Opt. method	Area	Overhead (%)	Search space size		Opt. method	Area	Overhead (%)	Search space size
Routing Logic	B&B	99	128%	262144	Routing Logic	Greedy	123	159%	1
	Greedy	111	144%			B&B			
Arbiter	Greedy	261	150%	7×10^{16}	Arbiter	Greedy	266	152.8%	3.5×10^{13}
FIFO	B&B	120	93%	8192	FIFO	B&B	120	93%	64
	Greedy	122	94.5%			Greedy	122	94.5%	

1) Coverage Density (CD) as sorting factor: In this variation, instead of just using CEI as the selection function of checkers for greedy algorithm, the coverage density (CD) is utilized as the sorting factor (based on the checker's area) (Eq. 1):

$$CD = \frac{CEI_{checker}}{Area_{checker}} \tag{1}$$

2) Dominant checkers' extraction : While evaluating the checkers separately, based on the number of detected and undetected stuck-at-0 and stuck-at-1 faults (which denote the number of True Detections and True Misses), it might be possible to extract checkers which provides the smallest values of True Misses for each line in the circuit, hence improving the CEI. A *dominant checker* for a circuit line is defined as a checker that has a minimum number of True Misses, while having a non-zero value for True Detections for that specific circuit line. If the number of dominant checkers for a circuit line is only one, that checker is called a *single dominant checker*. By selecting *single dominant checkers* in the beginning of the minimization process, the search space size is reduced, leading to speed-up of the optimization algorithm. However, it should be noted that picking such checkers does not necessarily result in a global optimal solution and it might be the case that the combination of two or more checkers results in 100% CEI with lower area. But, starting the optimization by picking the dominant checkers first, adds significant speed-up to the process.

V. Experimental Results

In this section, experimental results of checker minimization for control part of a NoC router are presented, which include the control part of FIFO, the routing logic and the arbitration unit. The synthesis of all the circuitsis performed using Class library by means of Synopsys Design Compiler [3]. For fault simulation, all the experiments are carried out using an extension of an in-house freeware test system Turbo Tester [4]. The experimental results for checker minimization are represented in Table I. DMR and TMR circuits for pseudo-combinational equivalent of each control unit circuitry are designed and synthesized. For routing logic and control part of FIFO, both optimization methods are used, but branch-and-bound algorithm was not applied to arbiter unit due to the huge problem size. The area overhead of the checkers are calculated based on the original control circuit size. The greedy algorithm used in the experiments, uses Coverage Density (CD) as sorting factor for checkers. The experiments show that the proposed method falls between DMR and TMR in terms of area overhead while providing fault localization information. Considering the search space size, using dominant checkers for speeding up the search process depends on the design of the checkers. In case of routing logic, mostly structural checkers are used which are directly extracted from the RTL. Most of the resulting fine tuned checkers will be categorized as dominant checkers which in turn results in massive reduction in search space and leaves no room for optimization (in case of routing logic, reducing the search space size to 1). A more balanced initial set of checkers (for example in case of arbiter unit), will result in a more balanced set of dominant checkers and will provide a reasonable search space size.

TABLE II: Area comparison of router with checkers and with the base-line router

	area (μm^2)	area overhead (%)	critical path delay (ns)	critical path overhead (%)
Baseline router	91163	–	3.38	–
Router with opt. checkers	107237	14.9%	3.43	1.4%

The full router with final set of minimized checkers and the baseline router (without any checkers) are synthesized using AMS 0.18 μm CMOS technology library [5]. The full router with checkers includes the control part, the data-path and the minimized set of checkers obtained from table I. The results of the synthesis and area overhead and critical path delay of the proposed method are presented in Table II. The area overhead compares the router including all checkers for all modules of the control part, with the baseline router without any checkers. The area overhead when considering the minimized set of checkers (while still reaching 100%) is about 15%, which is small compared to the total area of the baseline router. Moreover, with regards to the timing, the additional delay in the critical path of the router is also negligible.

Acknowledgments

The work has been supported by EU's H2020 RIA IMMORTAL, EU's Twinning Action TUTORIAL, Estonian Science Foundation grant ETF9429, Estonian institutional research grant IUT 19-1, Estonian IT Academy programme and funded by Excellence in IT in Estonia (EXCITE) project.

References

[1] P. Saltarelli, B. Niazmand, J. Raik, V. Govind, T. Hollstein, G. Jervan, and R. Hariharan, "A framework for combining concurrent checking and on-line embedded test for low-latency fault detection in noc routers," in *Proceedings of the 9th International Symposium on Networks-on-Chip*, ser. NOCS '15. New York, NY, USA: ACM, 2015, pp. 6:1–6:8.

[2] P. Saltarelli, B. Niazmand, R. Hariharan, J. Raik, G. Jervan, and T. Hollstein, "Automated minimization of concurrent online checkers for network-on-chips," in *Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2015 10th International Symposium on*, June 2015, pp. 1–8.

[3] (1994) Synopsys design compiler. http://www.synopsys.com/.

[4] M. Aarna, E. Ivask, A. Jutman, E. Orasson, J. Raik, R. Ubar, V. Vislogubov, and H. d. Wuttke, "Turbo tester - diagnostic package for research and training," *Tallinn University of Technology, Department of Computer Engineering Raja 15, Tallinn, Estonia Turbo Tester*, vol. 3, pp. 69–73.

[5] (2016) AMS 0.18um CMOS process. http://ams.com/eng/Products/Full-Service-Foundry/Process-Technology/CMOS/0.18-m-CMOS-process/.

APPENDIX G

S. P. Azad, B. Niazmand, K. Janson, N. George, A. S. Oyeniran, T. Putkaradze, A. Kaur, J. Raik, G. Jervan, R. Ubar, T. Hollstein, "From Online Fault Detection to Fault Management in Network-on- Chips: A Ground-Up Approach", 2017 The IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS), Apr 19-21 2017, Dresden, Germany.

From Online Fault Detection to Fault Management in Network-on-Chips: A Ground-Up Approach

Siavoosh Payandeh Azad*, Behrad Niazmand*, Karl Janson*, Nevin George*, Adeboye Stephen Oyeniran*, Tsotne Putkaradze*, Apneet Kaur*, Jaan Raik*, Gert Jervan*, Raimund Ubar*, Thomas Hollstein*†
Department of Computer Systems
*Tallinn University of Technology, †Frankfurt University of Applied Sciences
email: {siavoosh, bniazmand, karl.janson, nevin, steph, tsotne, akaur, jaan.raik, gert.jervan, raiub, thomas}@ati.ttu.ee

***Abstract*—Due to the ongoing miniaturization of silicon technology beyond the sub-micron domain and the trend of integrating ever more components on a single chip, the Network-on-Chip (NoC) paradigm has emerged to address the scalability and performance shortcomings of bus-based interconnects. As the feature size shrinks, the system gets much more susceptible to faults caused by wear-out and environmental effects. Thus, in order to increase the reliability, creates the need for having mechanisms embedded into such a system that could detect and manage the faults in run-time.**

In this paper, a ground-up approach from fault detection to fault management for such a NoC-based system on chip is proposed that utilizes both local fault management for fast reaction to faults and a global fault management mechanisms for triggering a large-scale reconfiguration of the NoC. Also, detailed description of strategies for fault detection, localization, classification and propagation to a global fault management unit are provided and methods for local fault management are elaborated.

***Keywords*—Fault Detection, Checkers, Fault Classification, Fault Localization, Fault management, Reconfiguration, Network-on-Chip.**

I. Introduction

Network-on-Chip (NoC) has emerged as a paradigm to address the scalability and performance shortcomings of traditional bus-based architectures [1], [2]. The trend of nano-scale electronics shrinking in size, makes them more susceptible to wear-out and environmental effects. This necessitates the detection and management of faults occurring at the run-time of the system, in order to provide higher reliability.

This work addresses a reliable NoC framework, which is maintained as an open-source project named Bonfire [3]. It provides support for fault detection and localization, local fault management, local fault classification, and fault information propagation to a global system health monitoring unit. In a NoC-based System-on-Chip, routers are responsible for transmitting data between the Processing Elements (PEs).

In Network-on-Chip, a router is composed of a data-path and a control part. The packets are transmitted via the data-path, while the control part directs the flow of data and the path the data should take when being transmitted between routers. Thus, both for the data-path and the control part, fault tolerance is of utmost importance for a reliable communication.

For the data-path, error detection and/or error correction techniques (such as single parity and Hamming encoding [4]) can be used. However, due to the area overhead of error correction techniques such as Hamming, the focus of this work is on single bit parity for the detection of faults in the data-path (inter-router links and data-path components of the routers).

On the other hand, faults in the control part of NoC routers should be handled. One way is to detect them via concurrent online checkers (for instance via the approaches proposed in [5], [6]) due to their low fault detection latency. There are also other methods such as Built-In-Self-Test (BIST) [7]. However, they interrupt the normal operation of the system for testing upon a fault occurrence. Thus, in the scope of this work, we focus on concurrent online detection of faults for the control part of routers. It is important to note that the checker outputs also facilitate fault localization [8], pinpointing the defective part in the circuit. Additionally, higher abstract deductions can be made based on them, such as existence of defect in turns in a router (a path from an input port to an output port). Such information can be used for reconfiguration of the routing algorithm or re-mapping of the tasks by units in charge of application mapping and scheduling. Works such as [9] have addressed multi-layer fault diagnosis and combining checkers at different levels of abstraction, however, they impose high latency. Furthermore, they have not addressed any mechanism for classification of faults and fault management, which are considered in our work.

In this work a ground-up approach from fault detection to management for NoC-based System-on-chips is proposed. Strategies for fault detection, localization, classification and propagation to a global fault management unit are described. Furthermore, in order to improve the reaction time to faults, methods for local fault management are elaborated.

The rest of this paper is organized as follows: in section II the basics of the Bonfire framework, including the NoC and router architecture are discussed. Section III describes the fault model used in this work and the method of fault injection on the links. In section IV different fault detection mechanisms for control and data path of the routers are discussed. Section V describes methods of fault localization. Sections VI describe the the process of fault classification and Section VII provides methods of handling faulty packets at the router level. Section VIII details fault information propagation to system health monitoring unit and finally, section X concludes the paper.

II. Bonfire framework

A. Bonfire NoC Architecture

The aim of the Bonfire project is to create a fault-tolerant framework for testing dependability mechanisms in a NoC-based System-on-Chip(SoC). The targeted NoC is using a 2D mesh topology where each tile of the network consists of a wormhole switching router equipped with fault tolerance

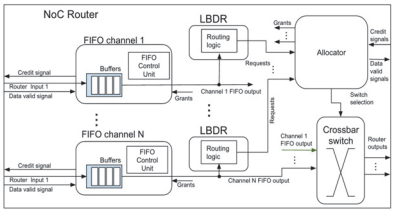


Fig. 1. Overview of the architecture of baseline credit-based flow control NoC router used in Bonfire network

mechanisms and a Processing Element (PE) connected to it via a Network Interface (NI). Each PE comprises a Plasma core [10], which is a 32-bit MIPS-I based open-source processor with three pipeline stages, along with 8 KB of RAM (as local memory). Details of the components of the framework are described in the following subsections. Bonfire is maintained as an open-source project, available at [3] .

B. Bonfire Router

The Bonfire network described in this paper utilizes 32 bit credit-based wormhole switching in the routers. Fig. 1 shows an overview of the baseline router used in the Bonfire network, without any fault-tolerance mechanism. The router comprises of an input buffer (implemented as First-In-First-Out (FIFO)), routing computation unit (implemented using Logic-Based Distributed Routing (LBDR) mechanism [11]), switch allocator (prioritizing multiple requests to the same output port based on Round-Robin policy) and crossbar switch.

We have opted for LBDR [11], since it is scalable compared to table-based routing in NoCs. Furthermore, LBDR describes the topology and the routing algorithm in a 2D NoC in terms of a fixed number of configuration bits, *i.e.* connectivity and routing bits. This makes it possible to use the connectivity bits for the indication of links in the 4 main directions as healthy or faulty, by setting the corresponding connectivity bit to zero (faulty) or one (healthy). Routing algorithm re-configuration (if necessary) can be done by changing the routing bits.

C. System Health Monitoring Unit (SHMU)

The Bonfire project targets a holistic system health monitoring and management solution. To implement this, a dedicated unit, called System Health Monitoring Unit (SHMU) [12], [13], is proposed which handles fault information collection and system-scale fault management and reconfiguration.

In Bonfire project SHMU runs as software on one of the Processing Elements (PEs) in the network. And if the processor fails, the SHMU tasks can be mapped on another node. Details about functionality and implementation of SHMU is beyond the scope of this paper.

III. Fault Model

In this work, we focus on single stuck-at fault model [14], which means in each router module only one fault can occur at a time. For data-path related modules, including the links, only one bit can get faulty at a time on the specific link. The same applies to the control part related modules. Thus, separate control part modules and data links from different ports can get faulty at the same time, but only one fault in each of them at a time. Transient faults are modeled as single stuck-at faults which last one clock cycle. Intermittent faults are modeled as bursts of transient faults in short periods. Permanent faults are modeled as a moving from transient fault to intermittent state and then finally with a permanent stuck-at fault.

In this work, fault injection is done using force command of ModelSim from Mentor Graphics [15]. The injection points are links between routers and also internal signals of the individual modules inside the router.

IV. Fault Detection

The Bonfire framework uses different methods for detection of faults in data-path and in the control part of the network.

A. Data-path Fault Detection

Since this work focuses on a single stuck-at-fault model, a simple parity checker module is used to cover all single-bit faults on the input ports of the router. Upon receiving a faulty flit, the router starts a fault classification process and also manages the fault locally in order to prevent network congestion (for more information, please refer to section VII).

B. Control Part Fault Detection

Concurrent Online checkers are utilized to detect faults in the control part of the NoC routers. A *checker* is a concurrent online fault detection module [5], [6]. It detects faults occurring at inputs and outputs of fan-out free regions [16] of the circuit with low latency. Since checkers provide fault information required for fault localization, this method is preferable to Double or Triple Modular Redundancy (DMR and TMR) schemes. The use of concurrent checkers for online fault detection in control part of NoC routers are described in more detail in [5], [6], [17], [18]. It is worth noting that the complete set of checkers for the control part of Bonfire NoC are available at [3], which covers the control part of FIFO, routing logic (LBDR) and allocator unit (allocator) shown in Fig. 1.

V. Fault Localization

As the number of checkers can grow very large (in the order of hundreds per router), it is not feasible to send the fault detection information from all these checkers to SHMU. Also, in case of a NoC router, for example, flipping of a bit in a register in one of the router's internal modules will not provide valuable information to the SHMU in the application layer. However, if the outputs of the checkers connected to this module are combined, it is possible to translate the output of the checkers into more meaningful abstracted information.

By combining the checkers for the control part of the router, it is possible to report faults at a more abstract level. For instance, in [8], a fault localization method is introduced which groups sets of checkers, making an assertion vector, facilitating finding fault location at different granularity levels in the control part of a NoC router. This can also be used when signaling higher levels in the architecture, such as the application level about the occurrence of faults.

Works such as [19], model faults in the control part as a complete node failure. In [20], illegal turns in the routers are detected, however, each router depends on the information

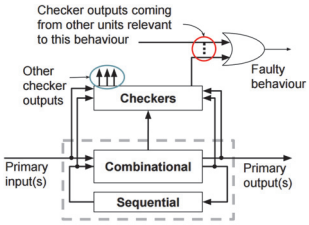


Fig. 2. General structure of the fault detection, grouping and classification mechanisms

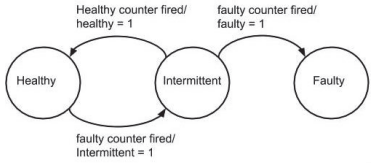


Fig. 3. Finite State Machine (FSM) for the fault classifier unit

from its neighbor routers for online fault detection. On the other hand, in our work, we combine checker outputs (as shown in Fig. 2) for the control part of a router. Further, this can be translated into detection of a *turn fault*. Unlike [20], we use the checker outputs in the current router to model turn faults, and there is no need for collecting information from neighbor routers. A turn fault is defined as a fault occurring in one of the components on the path from an input port to an output port of the router (*e.g.* a West to North turn fault or a straight path). This information can be passed to SHMU to the application layer. Later, if required, the SHMU can initiate re-configuration of the routing algorithm or re-mapping of the tasks on the nodes based on the fault information received from the lower (hardware) level.

VI. Fault Classification

With the growing number of transient faults, it would be impractical to send a separate notification to SHMU for each occurring fault. Not only would this impose additional latency by sending a notification from hardware to application layer, but it will also incur a significant power overhead.

To overcome this problem, faults are classified locally in the routers as *permanent*, *intermittent* or *transient*. The classified fault information is transmitted to SHMU if the fault is classified as intermittent or permanent. In [21], [22], an online fault classification mechanism is introduced as part of a cross-layer fault management framework, however, no details regarding the implementation of the fault classifier is provided. Whereas, in our work, a fault classification method based on [23], [24] is implemented; where a set of counters are used to count the healthy and faulty packets going through a network link. Each of the counters are compared with a threshold value. When a counter reaches its threshold, a signal is issued which is used by a control Finite State Machine (FSM) in charge of health making decision. Fig. 3 illustrates the FSM Diagram of the classifier unit. Every time the faulty packet counter

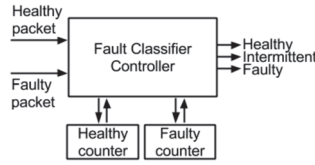


Fig. 4. Fault classifier block diagram

reaches its threshold, the FSM moves one step closer to the Faulty state. Every time the a counter reaches its threshold, both counters would be reset. It is noteworthy to mention that there could be different variations of state diagram models implemented for classification. The current state diagram as described in the Fig. 3 implements a scheme where there is no recycling of once faulty links. In contrast to [23], [24], since no error correction method has been used in this method, only two four-bit counters are utilized (see Fig. 4).

VII. Local Fault Management

Once a fault has been detected in the system, if it is classified as intermittent or permanent, the SHMU is notified. After obtaining the fault information, processing and making a decision, the SHMU can issue a command regarding that particular fault. But, during this time the effect of the fault has already propagated to other parts of the system and containment of the effects would be difficult if not impossible. So even though SHMU is responsible for fault management in the system, it can only manage the faults (in global scale) and some more detailed, distributed, mechanisms are needed for management of faults locally. This problem can be solved by implementing local fault management at each router. In this section two solutions for local management of the faults are provided.

A. Packet Dropping Mechanism

One of the important cases to be addressed is appearance of faulty flit at the input port of a router, where the following situations might happen:

- **Fault in the flit type:** in this case, it is usually not possible to identify the flit type and it (and also subsequent flits belonging to the same packet) cannot be routed. If this is not taken care of, eventually the input buffer (FIFO) of the router will get full, which can, in turn, leads to network congestion.
- **Fault in the payload data:** this type of fault does not have any effect on the network performance. However, since the packet data is corrupt, the fault will manifest itself in the application layer.
- **Fault in the destination address field:** the routing module might not be able to route the packet or the packet gets forwarded to a wrong destination. This might also result in network congestion if it is not properly taken care of.

One of the approaches to bypass the problem of having faulty flits is to use error-correction techniques, such as Hamming coding (single bit error correction, double bit error detection) for all flits. By comparing the overhead of these

TABLE I
AREA AND FLIT SIZE OVERHEAD COMPARISON OF SINGLE BIT PARITY AND HAMMING DECODER

Unit name	Unit area (μm^2)	Overhead in flit size (bits)
32-bit single bit parity checker	663	1
32-bit Hamming decoder	7050	8

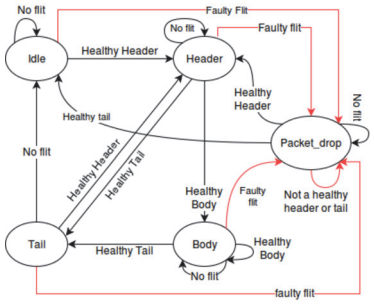


Fig. 5. Finite State Machine (FSM) diagram for the packet dropping mechanism

methods (shown in Table I) using AMS 0.18 μm CMOS technology library [25], it becomes clear that those methods impose additional area overhead to the correction circuit and also increase the flit (due to the additional bits needed for error correction). This, will affect the size of other network parameters, such as the physical link width which, in turn, also increases the size of the input buffer (FIFO) and crossbar switch.

In order to handle the above-mentioned situations, a *packet dropping* mechanism is incorporated in Bonfire framework. However, while using wormhole switching, in some cases, dropping the packet is not possible, for instance, when packet's header flit has already left the router. In such case, it is possible to cut the packet from the current position and attach a fake tail to it and forward it, while dropping the rest of the packet. This will not affect the network's operation. The results of such measures would manifest themselves in the application layer by comparing the packet length with the information in the header flit or as corrupt data. In our router architecture, the packet dropping mechanism is handled by a Finite State Machine (FSM), as is shown in Fig. 5. Additionally, the packet dropping mechanism has to generate fake credits for the upstream router in order not to interrupt the flow of the traffic.

In the Bonfire router, the packet dropping mechanism is improved even further by adding the flit saving functionality – a capability to detect position of the error in flits. In case the error is in the payload part, the packet will still be transmitted to its destination, thus making the application layer to handle the data errors. This will avoid re-transmissions in non-critical applications (for example many multimedia applications).

B. Routing Management

Once a link is classified as faulty, the router automatically sends this information to upstream router to update its LBDR connectivity bits (these bits can be over-written by SHMU later). If the change in connectivity bits happens when a packet is being processed, it might result in the packet being divided or mis-routed. In order to avoid this problem, the new connectivity bits should be stored in a register and routing module should wait until a new header flit arrives. The same approach is applied to routing bits of LBDR reconfiguration. The reconfiguration command is issued by the processing element at each node (once the reconfiguration message is issued by SHMU).

Another important point is to take care of faults occurring in the FIFOs which will be propagated to LBDR modules. In this case, to prevent congestion and network failure, the routing module (LBDR) should manipulate the FIFO modules in order to drop the packet. This is performed with a secondary and much simpler packet dropping mechanism, which generates fake grant signal to the FIFO when a faulty header is detected using a simple parity unit. Since LBDR is only sensitive to the header flit when making routing decisions, there is no need for support for cutting the packet and attaching fake tail. It is assumed that the dropped packets would be handled at the application layer.

VIII. FAULT INFORMATION PROPAGATION

The process of information transmission to the SHMU is also crucial. This can be done either (1) via reusing the existing network, or (2) by using an additional infrastructure working in parallel with the main NoC. There have been many proposals for fault information propagation to a global fault management unit. Some of the proposals, such as iJTAG [21], [22], [26], use scan chains. However, using an infrastructure like iJTAG requires single (or very limited) number of access points which limits the mapping possibilities for the SHMU on nodes since SHMU must have direct access to iJTAG access point. In addition, in approaches such as [21], [22], [26], the Instrument Manager (IM), which works as the iJTAG network controller and knows the structure of the instrumentation network, can become a single point of failure.

Some of the previous works in the literature have taken advantage of dual NoC architectures, such as [27]–[34]. In [34], in addition to the main network, a checker network is used (which is assumed to be 100% reliable) in order to deliver data to its destination in case of a fault occurrence in the main network. In [33], in parallel to the main NoC (which is used for transmitting the data), an additional control network is considered which is used for sending reconfiguration data for updating the connectivity and routing bits of LBDR in the network routers. The control network is used to inform a global manager node regarding faults occurring in different nodes. Despite the advantages these works might bring, they all incur additional area and power overhead. Moreover, if the area of the augmented circuitry for transmitting the fault information is not negligible, it can increase the probability of faults occurring in the additional network itself as well.

In this work, the classified fault information is propagated to SHMU via the NoC itself. The information would be bypassed to the Network Interface (NI). The NI will check the address of the SHMU and will pass the info to the node if SHMU is mapped on the same node (self diagnostic) or will generate

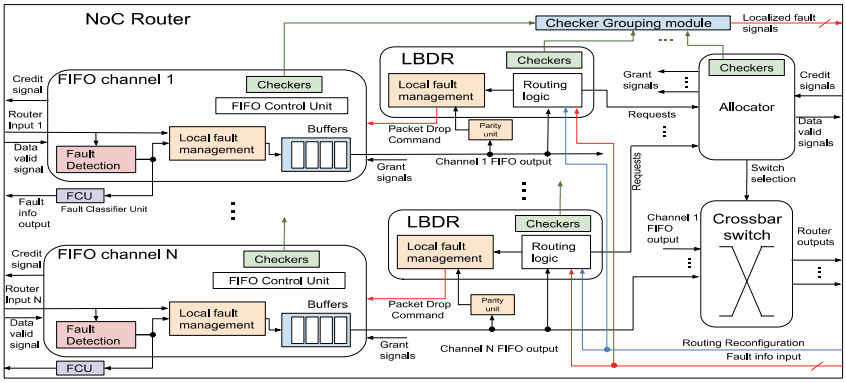


Fig. 6. Overview of the architecture of baseline credit-based equipped with fault tolerance mechanisms

TABLE II
AREA AND AVERAGE PACKET AND FLIT DROP FOR DIFFERENT PACKET DROPPING MECHANISMS.

Unit name	Unit area (μm^2)	Area overhead%	Average packet drop	Average flit drop
Original FIFO	14357	–	–	–
FIFO with packet dropping	16045	11.7%	≃1%	3.3%
FIFO with flit saving	16042	11.7%	≃1%	1.14%

TABLE III
AREA OVERHEAD RESULTS OF SELF-UPDATING LBDR OVER BASELINE LBDR

Unit name	Unit area (μm^2)	Increase in LBDR size	Increase in baseline router size
LBDR	1744	–	–
Self updating LBDR	2940	68.5%	5.9%

and send packets through the network to SHMU as soon as it finds idle time.

As mentioned in the previous section, after the local classification of the faults the information is sent to SHMU, which updates the system health map and can also trigger global re-configuration of the system in order to compensate for the faults. The reconfiguration packets will be sent to each node from SHMU and the node will send the reconfiguration information through the NI to the router. However, if the main NoC is used for transmission of the fault information and reconfiguration packets, under the running routing algorithm, the faults that should be reported, might also themselves prevent the messages to be correctly transmitted to the SHMU.

IX. RESULTS

Table II shows the area overhead of solutions for FIFO described in section VII (obtained using AMS 0.18 μm CMOS technology library [25] and synthesized via Synopsys Design Compiler [35]) along with the average flit and packet dropping ratio with random single stuck-at-fault injection on the network links with average rate of 5×10^6 faults per second. As it can be seen, when comparing the packet dropping approach to flit saving, the average full packet drop rate is not changing. This is due to the faults occurring in the header flit. However, the amount of flit drops is reduced by half, since the flits with the faulty payload will not be dropped in case of flit saving.

Table III shows the area overhead of the self updating LBDR unit. Both the area overhead of the self updating LBDR over the original LBDR (around 68%) and also its area overhead with respect to the baseline router without any fault tolerant mechanism (around 6%) are assessed.

By putting together all the mechanisms described in previous sections (fault detection, localization, classification and local management as shown in Fig. 6), the router grows 60.7% in area which is still lower than duplicate/triplicate-based methods such as DMR and TMR, while it also provides fault localization, management and system reconfiguration support at the same time. Moreover, the instantaneous detection of faults in the control part via the concurrent online checkers and combining them facilitates inferring more abstract and high level fault information (such as turn faults). Two main reasons for using such abstraction of the information are: (1) there is no advantage in transmitting very detailed fault information to the SHMU, since in order to make high-level decisions, SHMU has to abstract the information into turn faults. (2) Additionally, it reduces the amount of information to be transferred to SHMU through the NoC, thus, reducing the network latency and power consumption.

X. CONCLUSION

In this paper, a ground-up approach from fault detection to fault management for a Network-on-Chip based system is proposed. Concurrent online checkers are utilized to detect the faults in the control path and single parity check is used for the data-path. Fault localization and abstraction (into turn faults) are achieved by grouping information gathered from the control part checkers. Moreover, methods of local fault management at the hardware level using different packet dropping mechanisms are introduced and compared. To reduce the overhead of fault information propagation to application layer and its additional processing load, local fault classification mechanism is implemented which generates minimal, classified fault information for propagation.

Additionally, the necessity of having a relocatable System Health Monitoring Unit (SHMU) at the software layer is elaborated. SHMU utilizes the NoC itself for transmitting fault information after classification, thus avoiding a dual-NoC architecture and results in lower area overhead. The experimental results show the overall cost of applying such mechanisms, having 60.7% area overhead, which still makes it a better option than DMR/TMR-based approaches.

ACKNOWLEDGMENTS

The work has been supported by EU's FP7 STREP BASTION, H2020 RIA IMMORTAL, H2020 Twinning TUTORIAL, Estonian institutional research grant IUT 19-1, by the Estonian Center of Excellence in IT EXCITE funded by the European Regional Development Fund, and supported by Estonian IT Academy programme.

REFERENCES

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

[2] L. Benini and G. D. Micheli, "Networks on chips: a new soc paradigm," *Computer*, vol. 35, no. 1, pp. 70–78, Jan 2002.

[3] "Project bonfire network-on-chip," https://github.com/Project-Bonfire, 2015.

[4] S. Ghosh, N. A. Touba, and S. Basu, "Synthesis of low power ced circuits based on parity codes," in *23rd IEEE VLSI Test Symposium (VTS'05)*, May 2005, pp. 315–320.

[5] A. Prodromou, A. Panteli, C. Nicopoulos, and Y. Sazeides, "NoCAlert: An On-Line and Real-Time Fault Detection Mechanism for Network-on-Chip Architectures," in *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-45. IEEE Computer Society, 2012, pp. 60–71.

[6] P. Saltarelli, B. Niazmand, J. Raik, V. Govind, T. Hollstein, G. Jervan, and R. Hariharan, "A framework for combining concurrent checking and on-line embedded test for low-latency fault detection in noc routers," in *Proceedings of the 9th International Symposium on Networks-on-Chip*, ser. NOCS '15. New York, NY, USA: ACM, 2015, pp. 6:1–6:8.

[7] R. Sharma and K. K. Saluja, "An implementation and analysis of a concurrent built-in self-test technique," in *Fault-Tolerant Computing, 1988. FTCS-18, Digest of Papers., Eighteenth International Symposium on*, June 1988, pp. 164–169.

[8] K. Chrysanthou, P. Englezakis, A. Prodromou, A. Panteli, C. Nicopoulos, Y. Sazeides, and G. Dimitrakopoulos, "An online and real-time fault detection and localization mechanism for network-on-chip architectures," *ACM Trans. Archit. Code Optim.*, vol. 13, no. 2, pp. 22:1–22:26, Jun. 2016.

[9] G. Schley, A. Dalirsani, M. Eggenberger, N. Hatami, H. J. Wunderlich, and M. Radetzki, "Multi-layer diagnosis for fault-tolerant networks-on-chip," *IEEE Transactions on Computers*, vol. PP, no. 99, pp. 1–1, 2016.

[10] Plasma CPU. http://plasmacpu.no-ip.org.

[11] J. Flich and J. Duato, "Logic-based distributed routing for nocs," *IEEE Computer Architecture Letters*, vol. 7, no. 1, pp. 13–16, Jan 2008.

[12] S. P. Azad, B. Niazmand, P. Ellervee, J. Raik, G. Jervan, and T. Hollstein, "Socdep2: A framework for dependable task deployment on many-core systems under mixed-criticality constraints," in *2016 11th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, June 2016, pp. 1–6.

[13] S. P. Azad, B. Niazmand, J. Raik, G. Jervan, and T. Hollstein, "Holistic approach for fault-tolerant network-on-chip based many-core systems," *CoRR*, vol. abs/1601.07089, 2016. [Online]. Available: http://arxiv.org/abs/1601.07089

[14] A. Dalirsani, "Self-diagnosis in network-on-chips," PhD Thesis, Institut fr Technische Informatik der Universitt Stuttgart, July 2015.

[15] "ModelSim, Mentor Graphics," https://www.mentor.com/products/fv/modelsim/, 2017.

[16] R. Ubar, J. Raik, and H. Vierhaus, *Design and Test Technology for Dependable Systems-on-chip*, ser. Premier reference source. Information Science Reference, 2010. [Online]. Available: https://books.google.ee/books?id=_1zzPfTZND8C

[17] P. Saltarelli, B. Niazmand, R. Hariharan, J. Raik, G. Jervan, and T. Hollstein, "Automated minimization of concurrent online checkers for network-on-chips," in *Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2015 10th International Symposium on*, June 2015, pp. 1–8.

[18] P. Saltarelli, B. Niazmand, J. Raik, R. Hariharan, G. Jervan, and T. Hollstein, "A framework for comprehensive automated evaluation of concurrent online checkers," in *Digital System Design (DSD), 2015 Euromicro Conference on*, Aug 2015, pp. 288–292.

[19] Y. Jojima and M. Fukushi, "A fault-tolerant routing method for 2d-mesh network-on-chips based on components of a router," in *2016 IEEE 5th Global Conference on Consumer Electronics*, Oct 2016, pp. 1–2.

[20] L. Huang, X. Zhang, M. Ebrahimi, and G. Li, "Tolerating transient illegal turn faults in nocs," *Microprocessors and Microsystems*, vol. 43, pp. 104 – 115, 2016, many-Core System-on-Chip Architectures and Applications (PDP 15). [Online]. Available: //www.sciencedirect.com/science/article/pii/S0141933116000284

[21] K. Shibin, S. Devadze, and A. Jutman, "On-line fault classification and handling in ieee1687 based fault management system for complex socs," in *2016 17th Latin-American Test Symposium (LATS)*, April 2016, pp. 69–74.

[22] A. Jutman, K. Shibin, and S. Devadze, "Reliable health monitoring and fault management infrastructure based on embedded instrumentation and ieee 1687," in *2016 IEEE AUTOTESTCON*, Sept 2016, pp. 1–10.

[23] J. Silveira, M. Bodin, J. M. Ferreira, A. C. Pinheiro, T. Webber, and C. Marcon, "A fault prediction module for a fault tolerant noc operation," in *Sixteenth International Symposium on Quality Electronic Design*, March 2015, pp. 284–288.

[24] J. Silveira, C. Marcon, P. Cortez, G. Barroso, J. a. M. Ferreira, and R. Mota, "Scenario preprocessing approach for the reconfiguration of fault-tolerant noc-based mpsocs," *Microprocess. Microsyst.*, vol. 40, no. C, pp. 137–153, Feb. 2016. [Online]. Available: http://dx.doi.org/10.1016/j.micpro.2015.08.005

[25] (2016) AMS 0.18um CMOS process. http://ams.com/eng/Products/Full-Service-Foundry/Process-Technology/CMOS/0.18-m-CMOS-process/.

[26] "Ieee approved draft standard for access and control of instrumentation embedded within a semiconductor device," *IEEE P1687/D1.71, March 2014*, pp. 1–347, Nov 2014.

[27] A. K. Abousamra, R. G. Melhem, and A. K. Jones, "Deja vu switching for multiplane nocs," in *Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on*, May 2012, pp. 11–18.

[28] R. Das, S. Narayanasamy, S. K. Satpathy, and R. G. Dreslinski, "Catnap: Energy proportional multiple network-on-chip," in *Proceedings of the 40th Annual International Symposium on Computer Architecture*, ser. ISCA '13. New York, NY, USA: ACM, 2013, pp. 320–331. [Online]. Available: http://doi.acm.org/10.1145/2485922.2485950

[29] A. Flores, J. L. Aragon, and M. E. Acacio, "Heterogeneous interconnects for energy-efficient message management in cmps," *IEEE Transactions on Computers*, vol. 59, no. 1, pp. 16–28, Jan 2010.

[30] S. Volos, C. Seiculescu, B. Grot, N. K. Pour, B. Falsafi, and G. D. Micheli, "Ccnoc: Specializing on-chip interconnects for energy efficiency in cache-coherent servers," in *Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on*, May 2012, pp. 67–74.

[31] N. E. Jerger, A. Kannan, Z. Li, and G. H. Loh, "Noc architectures for silicon interposer systems: Why pay for more wires when you can get them (from your interposer) for free?" in *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, Dec 2014, pp. 458–470.

[32] J. Balfour and W. J. Dally, "Design tradeoffs for tiled cmp on-chip networks," in *Proceedings of the 20th Annual International Conference on Supercomputing*, ser. ICS '06. New York, NY, USA: ACM, 2006, pp. 187–198. [Online]. Available: http://doi.acm.org/10.1145/1183401.1183430

[33] A. Strano, D. Bertozzi, F. Trivino, J. Sanchez, F. Alfaro, and J. Flich, "Osr-lite: Fast and deadlock-free noc reconfiguration framework," in *Embedded Computer Systems (SAMOS), 2012 International Conference on*, July 2012, pp. 86–95.

[34] R. Parikh and V. Bertacco, "Formally enhanced runtime verification to ensure noc functional correctness," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-44. ACM, 2011, pp. 410–419.

[35] (1994) Synopsys design compiler. http://www.synopsys.com.

Curriculum vitae

Personal data

Name: Behrad Niazmand
Date of birth: 03/12/1986
Place of birth: Tehran
Citizenship: Iran

Contact data

E-mail: bniazmand@ati.ttu.ee , bniazmand@gmail.com

Education

2014– 2018 Tallinn University of Technology—PhD
2010– 2012 MSC, Science and Research Branch, Azad University
2005– 2009 BSC, South-Tehran Branch, Azad University
2001– 2005 Roshd High school and Pre-University

Language competence

Persian Fluent (Mother Tongue)
English Fluent (B2)
Estonian Basic (A1)
German Basic (A2)

Professional employment

2014– Present Early Stage Researcher, Tallinn University of Technology

Elulookirjeldus

Isikuandmed

Nimi: Behrad Niazmand
Sünniaeg: 03.12.1986
Sünnikoht: Teheran
Kodakondsus: Iraan

Kontaktandmed

E-post: bniazmand@ati.ttu.ee, bniazmand@gmail.com

Hariduskäik

2014–2018 Tallinna Tehnikaülikool – doktorikraad
2010–2012 Azadi ülikool – tehnikateaduse magister
2005–2009 Azadi ülikool – tehnikateaduse bakalaureus
2001–2005 Roshdi keskkool ja kõrgkool – keskharidus

Keelteoskus

Pärsia keel – kõrgtase (emakeel)
Inglise keel – kõrgtase (B2)
Eesti keel – algtase (A1)
Saksa keel – algtase (A2)

Teenistuskäik

2014 – praegune nooremteadur, arvutisüsteemide instituut, Tallinna Tehnikaülikool