TALLINN UNIVERSITY OF TECHNOLOGY
DOCTORAL THESIS
24/2018

Cross-Layer Dependability Management in Network on Chip based System on Chip

SIAVOOSH PAYANDEH AZAD

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 10.05.2018.

Supervisors: Prof. Thomas Hollstein, PhD,
Prof. Gert Jervan , PhD,
Prof. Jaan Raik , PhD,
Department of Computer Systems
Tallinn University of Technology, Tallinn, Estonia

Opponents: Dr. Leandro Soares Indrusiak
Computer Science department
University of York, United Kingdom

Prof. Dr.-Ing. Alberto Garcia-Ortiz
Institute of Electrodynamics and Microelectronics
University of Bremen, Bremen, Germany

Defense of the thesis: June 13th, 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 submitted for doctoral or equivalent academic degree.

ISSN 2585-6898 (publication)
ISBN 978-9949-83-259-0 (publication)
ISSN 2585-6901 (PDF)
ISBN 978-9949-83-260-6 (PDF)

TALLINNA TEHNIKAÜLIKOOL
DOKTORITÖÖ
24/2018

Kiipvõrkudel põhinevate süsteemide kihtideülene usaldatavuse haldus

SIAVOOSH PAYANDEH AZAD

TTÜ KIRJASTUS

To Marieh ...

TABLE OF CONTENTS

LIST OF PUBLICATIONS

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

A T. Hollstein, S. P. Azad, T. Kogge, H. Ying and K. Hofmann, "NoCDepend: A Flexible and Scalable Dependability Technique for 3D Networks-on-Chip," 2015 IEEE 18th International Symposium on Design and Diagnostics of Electronic Circuits & Systems, Belgrade, 2015, pp. 75-78.

B 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.

C 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 IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), Dresden, 2017, pp. 48-53.

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, Cyprus, 2017, pp. 1-2.

E S. P. Azad, B. Niazmand, K. Janson, T. Kogge, J. Raik, G. Jervan, T. Hollstein, "Comprehensive performance and robustness analysis of 2D turn models for network-on-chips," 2017 IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, MD, 2017, pp. 1-4.

F 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, Spain, 2017, pp. 1-8.

OTHER RELATED PUBLICATIONS

G 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.

H 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.

I S. P. Azad, B. Niazmand, J. Raik, G. Jervan, T. Hollstein: ”Holistic Approach for Fault-Tolerant Network-on-Chip based Many-Core Systems”,2nd International Workshop on Dynamic Resource Allocation and Management in Embedded, High Performance and Cloud Computing DREAMCloud 2016 (arXiv:cs/1601.04675), DREAMCloud/2016/05

J A. S. Oyeniran, R. Ubar, S. P. Azad and J. Raik, ”High-level test generation for processing elements in many-core systems,” 2017 12th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), Madrid, Spain, 2017, pp. 1-8.

OTHER PUBLICATIONS

K S. P. Azad, N. Farahini and A. Hemani, ”Customization methodology of a Coarse Grained Reconfigurable architecture,” 2014 NORCHIP, Tampere, 2014, pp. 1-4.

L S. P. Azad, H. Kinks, M. A. Tajammul and P. Ellervee, ”An ad-hoc implementation of a remote laboratory,” 2015 IEEE International Conference on Microelectronics Systems Education (MSE), Pittsburgh, PA, 2015, pp. 48-51.

M M. A. Tajammul, S. P. Azad and P. Ellervee, ”Digital system modeling and synthesis as an introduction to Computer Systems Engineering,” 2015 IEEE International Conference on Microelectronics Systems Education (MSE), Pittsburgh, PA, 2015, pp. 52-55.

N U. Reinsalu, S. P. Azad, M. Leier, K. Tammemae and T. Hollstein, ”Practicing start-up culture in teaching embedded systems,” 2016 11th European Workshop on Microelectronics Education (EWME), Southampton, 2016, pp. 1-6.

O S. K. Dwivedi, S. P. Azad, P. Ellervee and R. Dash, ”Hardware implementation of face recognition using low precision representation,” 2016 15th Biennial Baltic Electronics Conference (BEC), Tallinn, 2016, pp. 63-66.

AWAITING PUBLICATION

P Serhiy Avramenko, Siavoosh Payandeh Azad, Stefano Esposito, Behrad Niazmand, Massimo Violante, Jaan Raik, Maksim 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).

Q Siavoosh Payandeh Azad, Adeboye Stephen Oyeniran, Raimund Ubar, ”Replication-Based Deterministic Testing of 2-Dimensional Arrays with Highly Interrelated Cells”, 2018 IEEE 21st International Symposium on Design and Diagnostics of Electronic Circuits (DDECS).

R Behrad Niazmand, Siavoosh Payandeh Azad, Tara Ghasempouri, Jaan Raik, Gert Jervan, ”Checker Extraction Methodology for Control-Oriented Circuits”, 2018, 2nd International Test Conference in Asia (ITC-Asia).

S Tara Ghasempouri, Siavoosh Payandeh Azad, Behrad Niazmand, Jaan Raik, ”An automatic approach to evaluate assertions’ quality based on data-mining metrics”, 2018, 2nd International Test Conference in Asia (ITC-Asia).

T Adeboye Stephen Oyeniran, Siavoosh Payandeh Azad, Raimund Ubar, ”Combined Pseudo-Exhaustive and Deterministic Testing of Array Multipliers”, International Conference on Automation, Quality and Testing, Robotics (AQTR).

U Adeboye Stephen Oyeniran, Siavoosh Payandeh Azad, Raimund Ubar, ”Parallel Pseudo-Exhaustive Testing of Array Multipliers with Data-Controlled Segmentation”, 2018 IEEE International Symposium on Circuits & Systems (ISCAS).

AUTHOR'S CONTRIBUTIONS TO THE PUBLICATIONS

Contribution to the papers in this thesis are:

A The author optimized the proposed fault information propagation algorithm and developed a tool for calculating the off-line algorithm and later-on extended xHiNoC framework to evaluate the proposed mechanism.

B The author proposed and developed a framework for dependable task deployment on NoC-based SoCs under mixed criticality constraints. Also the effectiveness of the proposed framework was evaluated by the author.

C The author proposed a local fault management mechanism for NoC routers based on packet-dropping, along with a proposal for fault information abstraction, classification and transfer to the system-level fault manager.

D The author proposed and developed an automated minimization framework for concurrent on-line checkers. Also the concept of dominance in such checkers and a new metrics for evaluating their effectiveness (in contrast to the general fault coverage of such circuits) were proposed by the author.

E The author proposed a method for enumerating and analysis of all 2D turn-model-based deadlock-free routing algorithms. The author proposed a new metric for evaluating the degree of adaptivity of non-minimal path routing algorithms, Later on, evaluating the fault tolerance of a selected set of 2D deadlock free routing algorithms under different amounts of link failures.

F The author proposed the architectures investigated in the work. Also the author performed the reliability analysis for the proposed architectures in the paper.

AUTHOR'S CONTRIBUTIONS TO OTHER RELATED PUBLICATIONS

G The author extended the previously developed tool to use NoCDepend method for network partitioning.

H The author proposed optimization in the LBDR3D circuitry to improve the area overhead of the method alongside collaborating on designing the experiments.

I The author proposed an architecture for efficient prediction and storage of mapping of applications on NoC-based SoCs and evaluated the performance gain for such approach.

J The author developed an implementation of the BIST method in hardware and evaluated the efficiency of the such approach and compared to other existing methods.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Professor Thomas Hollstein for his support throughout my PhD studies. I would also like to express my deep gratitude to my co-supervisors Professor Gert Jervan and Professor Jaan Raik. For the numerous meetings, discussions, comments and their open-door approach to supervision. Without your guidance none of this would be possible.

Special thanks to the Head of Department of Computer Systems, Margus Kruus for supporting me throughout my studies.

I would like to thank my colleagues and friends: Muhammad Adeel Tajammul, Behrad Niazmand, Karl Janson, Hannes Kinks, Tara Ghasempouri, Adeboye Stephen Oyeniran, Apneet Kaur, Nevin George, Priit Ruberg and Mairo Leier.

Furthermore, I like to acknowledge the organizations that have supported my PhD studies: Tallinn University of Technology, Estonian IT Academy program, EU's H2020 RIA IMMORTAL, the Estonian Center of Excellence in IT (EXCITE) and EU's Twinning Action TUTORIAL project.

Finally, I would like to thank my family, my parents, my brother and especially my beloved Mari!

Siavoosh Payandeh Azad
Tallinn, 2018

ABBREVIATIONS

AG	Architecture Graph
AHB	Advanced High-performance Bus
AMBA	Advanced Micro-controller Bus Architecture
AXI	Advanced eXtensible Interface
BIST	Built In Self Test
CEI	Coverage Efficiency Index
CB	Credit-Based
CD	Coverage Density
CTG	Clustered Task Graph
DMR	Double Modular Redundancy
ECC	error-correcting code
FIFO	First-In-First-Out
FSM	Finite State Machine
GA	Genetic Algorithm
GUI	Graphical User Interface
HS	Hand-Shaking
ILS	Iterative Local Search
LBDR	Logic Based Distributed Routing
LS	Local Search
MCT	Minimum Completion Time
MET	Minimum Execution Time
MPM	Most Probable Mappings
MPFS	Most Probable Faults Set
MSU	Mapper/Scheduler Unit
NoC	Network on Chip
OPS	Original Problem Size
RG	Routing Graph
RPS	Reduced Problem Size
SA	Simulated Annealing

SHM	System Health Map
SHMU	System Health Monitoring Unit
SoC	System on Chip
TG	Task Graph
TM	Turn Model
TMR	Triple Modular Redundancy

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

With the constant shrinking of the technology node feature size, it is possible to integrate more components in the system. This in-turn brings up the problem of communication between the components, turning the communication medium into the system's bottleneck. Different approaches have been introduced for solving this problem, such as modern bus systems. Long wires in the chips with growing size and limited number of simultaneous accesses to the buses are driving them unusable. Networks-on-Chip (NoCs) [6] have emerged as a scalable solution for solving the communication bottleneck. This is achieved by reducing the wire-length between the resources and providing higher flexibility in communication. NoCs paradigm replaces the direct connections with a network of routers where the data is transferred in form of packets. Using this approach, the network provides simultaneous access to the communication medium for every connected resource. Furthermore, the network topologies can be tailored to fit the application requirements. The provided flexibility however is gained by sacrificing latency and throughput due to data packetization overhead and number of clock cycles spent in each router.

This shrinking technology feature size results in an increase in systems susceptibility to faults and wear-out leading to a shorter lifetime of system. The shorter component lifetime along with growing number of faults during its lifetime, requires new dependability techniques to prolong the systems lifetime with a degraded performance. The NoC-based SoCs suffer even more from the above problems since even Single Event Upsets (SEU) in the network components might result in a network-wide congestion which in turn can result in a system failure. The above mentioned characteristics of failures in the current technology nodes along with the behavior of the NoCs specifically, motivates research towards more dependable NoC-based systems.

Many works have targeted fault tolerance mechanisms for different parts of the NoC based SoCs. However, the main problem is that each of these works solves a single issue regarding dependability in the network with acceptable overhead (which makes each approach scalable by itself e.g. [7, 8]); However, combining these mechanisms to cover all of the issues, will result in such an overhead that will render the final fault tolerance mechanism not scalable. To

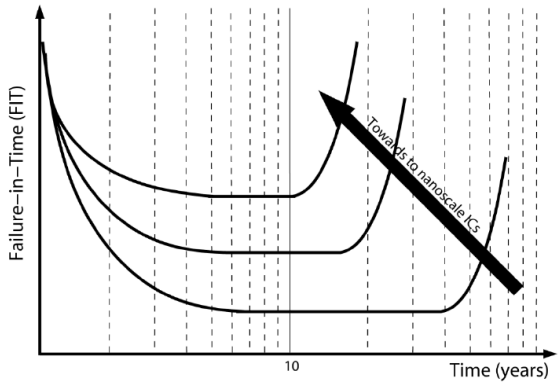


Figure 1 Progression of bathtub curve with shrinking transistor feature size[1]

this end, this thesis provides a scalable cross layer dependability solution for NoC based SoCs.

In the literature, the following three attributes are considered for dependable systems: Reliability (the probability of a failure free operation of the system in time interval $[0,t]$), Availability(average time that the system is operational during the time interval $[0,t]$), and Safety (similar to reliability but while considering failures that can lead to safety hazards) [9]. This work only focuses on improving the first two components by providing software and hardware based solutions.

Motivation

The shrinking feature size, makes system components more susceptible to failures [10]. The system failure rate during its lifetime is transitionally visualized by the famous bath-tub curve [9]; The horizontal axis describes the time while the vertical axis represents the number of failed devices in time. Where three components namely infant mortality, random constant failures and system wear out produce a bath-tub looking curve over the lifetime of the system. However, the bath-tub curve changes for different technology nodes. Fig.1 shows the bath-tub curve progress over time. Shrinking technology feature size have resulted in systems susceptibility to new fault types (such as NBTI [11] resulting in components wear-out) alongside with the increased effects of the known fault sources ($e.g.$ increase in possibility of multiple components –gates, memory cells etc.– being affected by a particle hit). As its shown in the figure, with each technology node, the constant failure rate grows and shifts the curve upwards. At the same time higher infant morality and wear-out rates shrink the size of the tub.

In the meantime, the limitations of traditional and modern bus based (e.g. AMBA's AHB and AXI bus etc.) communication mediums. However, such approaches impose limitations once the number of the connected components grow (16 bus masters in case of AMBA bus[1]). Also these approaches impose a communication bottleneck on parallel computing in multi/many-core systems. Network based communication schemes provide a scalable solution for such problems, rendering Networks-on-Chip (NoCs) a widely accepted paradigm by the industry [12]. In traditional bus based systems, once the access to the bus is acquired, the data can be sent in burst. In contrast, Network based communication requires one or more control flits for navigating the packets in the network along with some wrapping around the payload data (for forming the flits) which reduces the data-throughput significantly. Also factors such as latency of packets in the network (due to packets being forwarded by each router in the network), considerably limits the communication. Networks-on-Chip should be used with great care in Systems-on-Chip(SoCs); Problems such as deadlock (where packets wait on one another in a circular path for release of resources), or live-lock (where a packet moves in circles in the network and never reaches the destination) can easily paralyze the network. Network congestion is another limiting factor of NoCs, if the traffic in the network is not properly distributed. Its important to note that NoCs impose considerable area overhead to the chips. Hence more faults can occur in the communication medium; up to the point that such faults can not be ignored any further.

The shorter lifetime of the components in NoC based systems on a chip (SoCs) –including but not only limited to network components– highlights the need for fault tolerant solutions. The resulting solution should prolong systems availability by providing a graceful performance degradation. To this end, using classical test methods would not be sufficient. and new fault detection, localization and management techniques are required to handle the new problems. Also system availability [9] which is a vitally important factor, requires runtime fault-tolerance mechanisms and methods for shortening system downtime due to reconfiguration.

Considering the pressing issues which lead to shorter life in the components, it is crucial to provide dependable solutions which take into account upcoming problems in all levels of abstraction. More specifically, the following challenges still exist in the field of NoC based SoCs:

- Large area overhead of control-part fault detection and classification mechanisms: Fault detection mechanisms for control-path of the units which provide localization impose a large area overhead in the NoC routers. Depending on the considered fault model in the NoC router, the number of fault signals coming from the detector circuits might be rather

[1]please refer to https://www.arm.com/products/system-ip/amba-specifications

large. The existing hardware-based fault classification methods are not scalable for applying on all important fault signals in a NoC router.

- Network-wide congestion due to faults in the routers' data-path: Since routers make routing decision based on the routing information encoded in the packet's header, If a data-path of the router (including input and output buffers, crossbar switch, physical link between adjacent routers) becomes faulty, It might cause either packet mis-route or in the worst case scenario might lead to routing logic becoming incapable of routing the packet. For example it can happen in cases that the routing algorithm doesn't support the turns necessary for required routing. Another case would be the corruption of flit types which in turn might lead to congestion in the router. Congestion in one router can easily propagate to the rest of the network and cause a network-wide congestion.

- Lack of low-latency reachability guarantee for packets in the network under arbitrary fault configuration: There exist many works in the literature targeting fault avoidance in on-chip networks. However, all these approaches suffer either from high latency due to path-finding methods or from incapability of handling concave fault configurations.

- Reliability degradation of the routers due to fault tolerance mechanisms' hardware overhead: Considering growth of number of faults in relation to growing the hardware size, and the large area overhead of the fault tolerant mechanisms, it is not possible to safely assume that the fault tolerant system is fault free. Since a healthy component with a faulty fault-tolerance (such as detection, classification and handling) mechanism is indistinguishable from a faulty component with healthy fault-tolerance mechanisms, the system's reliability will suffer regardless of the fault's location. An evaluation of the impacts of such methods on reliability of routers is needed.

- Lack of a simulation environment for evaluating fault tolerance mechanisms: In recent years many simulation environments have been proposed for NoC based systems. However, a simulator environment which considers the effects of the faults in the system from application layer and models fault tolerance systems is missing in the literature.

The above mentioned challenges motivates further research on cross-layer fault management in NoC based SoCs. The next section lays the details of the tasks at hand.

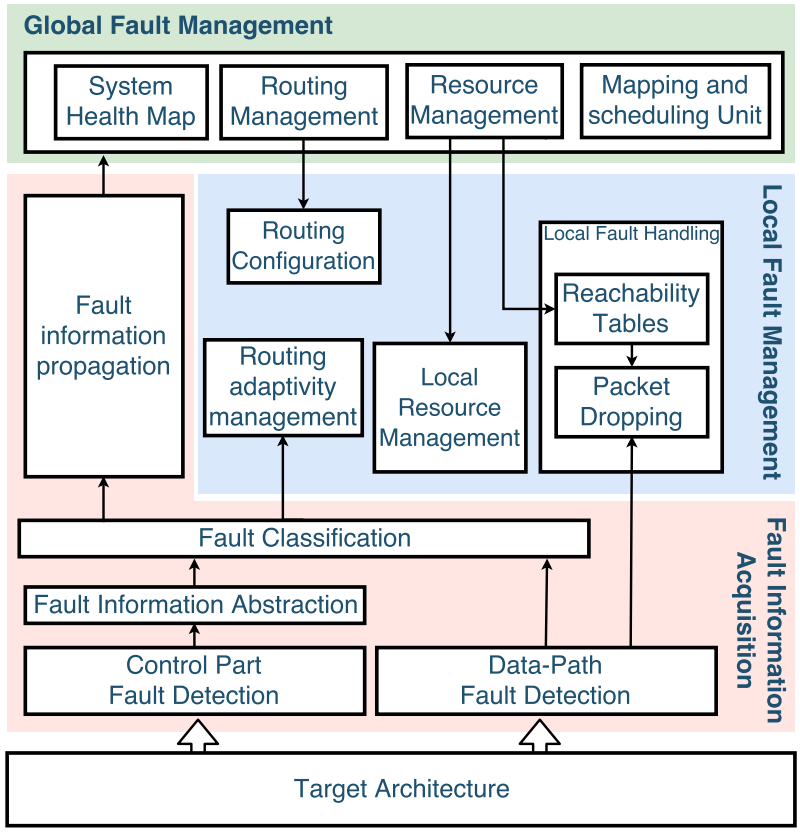


Figure 2 Components of a possible cross-layer fault management system for a NoC based SoC

Problem formulation

Providing support for graceful degradation to the system requires system which have global view of the system in order to enable system-wide decisions. However, latency of the global mechanisms' response is not acceptable. This motivates integration of local mechanisms that are close to hardware which provide fast response. Such mechanisms should be dependent on the global system manager as little as possible.

Considering the above mentioned characteristic of global and local dependability mechanisms, it becomes obvious that a cross-layer dependability system is required. Such a cross-layer approach may consist of three main layers: Fault information acquisition, Local and Global fault management. Fault information acquisition can be performed by collection, abstraction, classification and propagation of fault information obtained from data-path and control-part fault detection mechanisms. The fault information produced in this

layer should be consumed by other layers. In contrast to the fault information acquisition, layer which can be an independent layer, the local and global fault management are closely intertwined; Most of the mechanisms that act locally rely on computation and configuration from a global manager. Figure 2 depicts components of a possible cross-layer fault management mechanism.

Global fault management layer should maintain a system-wide health map by using obtained fault information. Based on this health map, the global fault manager can perform system-wide resource management and application deployment(via mapping and scheduling unit). The mechanisms that rely on system manager for reconfiguration (such as routing reconfiguration, local resource management and reachability management) but act directly at the hardware level, can be excluded from global mechanisms, and may be classified as hybrid-local fault management.

The local and hybrid fault management layer, can utilize the processed fault information from the fault information acquisition layer along with commands from the global fault management unit to perform local fault handling, routing and resource management and providing reachability guarantees. Parts of the local fault management may be controlled/reconfigured globally but act locally (Hybrid-Local mechanisms). A simple example of such systems can be the routing units which provides a level of fault management by utilizing the routing adaptivity to tolerate faulty links in the network, while the routing algorithm may be reconfigured by the global fault manager. However, since in this scenario the global fault manager doesn't make the decisions to avoid a particular faulty situation, this part would be still considered as local fault management.

The main goal of this work is to address problems mentioned in the previous section by introducing:

- Practical hardware solutions for fault-information processing.
- Scalable Local fault management solutions for NoC routers
- Global fault management through dependable application deployment and hardware reconfiguration.

More concretely, considering the challenges discussed in the previous section under cross-layer fault management approach described above, the following problems will be tackled in this work:

- Providing a low latency fault detection and classification mechanism for control-part faults with acceptable area overhead
- Providing lightweight local fault handling mechanism at the router level

- Proposing a scalable mechanism for providing reachability guarantee for packets in the network under arbitrary faulty link configuration and any turn-model-based routing algorithm
- Proposing reliability evaluation framework for NoC routers that considers the effects of fault tolerance mechanisms on reliability
- Devising a simulation environment for evaluating the effects of faults at system level and the efficiency of fault tolerance mechanisms.

The next section describes the contributions of this thesis regarding the above mentioned tasks in more details.

Contributions

This work focuses on a cross-layer approach using the above mentioned structure. In order to clarify the global and local terms, in this work, a mechanism is considered local if it is distributed over the system, close to hardware and does not require global information for each decision. The mechanisms that require global reconfiguration but fall under the above mentioned criteria are not considered global but are labeled as Hybrid-Local mechanisms (since their local nature is much more dominant).

The main contributions of the scientific work presented in this thesis are:

- The fault detection mechanisms for control part of the circuit is considered in this work (concurrent online checkers). This provides detailed fault localization information but has a large hardware overhead. In this work a minimization framework for such checkers is proposed. A new metric, namely coverage density has been introduced for selecting the checkers during the optimization process. Also new concept of dominant checkers have been introduced for speeding up the process by reducing the search space of the problem. This contribution led to **publication D** [13].
- The large number of fault diagnosis signals in the router (specially coming from the concurrent online checkers) requires more lightweight hardware-based classification methods. This thesis extends the existing works for hardware-based online fault classification to make them more practical for abstracted checker information as well as the links. This contribution led to **publication C** [14].
- The current metric available in the literature for comparing adaptivity of different turn model based routing algorithms only considers minimal path routing in the networks. This work extends the degree of adaptiveness used in minimal path routing to cover non-minimal path routing as well.

This metric provides the possibility of further classification of turn model behavior under different link-failure scenarios. This contribution led to **publication E** [15].

- In the current state of the art, several 2D uniform turn-model based routings are identified. However, an enumeration and evaluation of all possible possible cases was missing. This work provides an enumeration of all deadlock free, 2D turn-model based routing algorithms which provide full connectivity in 2D mesh networks. Later, these turn models are compared in terms of their robustness to all configurations of link failure in the network. This contribution led to **publication E** [15].

- Another pressing issue in the field of NoC-based SoC is partial or full-scale network failure due to faults in the data-path of routers. If a router fails to route or miss-route a packet or parts of a packet due to faults, it is possible that all the communication in the upstream routers to become stalled. This work provides a local fault handling mechanism to protect the network from such congestion scenarios. This contribution was reported in **publication C** [14].

- It is possible that a routing algorithm provides a viable path from a source to destination but the local routing units might fail to find the path due to their lack of global information. This will lead to congestion in the network due to packets being routed in undesired (but absolutely legal) directions. This work proposes a novel method to guarantee that the injected packets in the network would reach their destination in faulty networks. This contribution led to **publication A** [16].

- It is crucial to evaluate the effects of faults on NoC-based SoC systems at the application layer along with effectiveness of global fault management mechanisms to handle such situations. This work provides a new simulator for NoC based SoCs for modeling global fault management under mixed-criticality constraints. This contribution led to **publication B** [17].

- Since router architectures are very regular, it is possible to reuse functioning resources that have been cut-off due to their neighboring units. Such a task requires a local resource management along with evaluation of the effects of the additional supporting infrastructure. This work proposes three architectures for resource reuse in the NoC routers along with an evaluation framework to compare their effect on reliability of the router. This contribution led to **publication F** [18]

Thesis organization

This thesis contains 4 chapters. Chapter 2 provides details of fault detection, abstraction, classification and propagation mechanisms. Chapter 3 delves into problem of local and hybrid fault management and chapter 4 discusses system level fault management schemes. Finally chapter 4.6 concludes this work and proposes some ideas for future work.

1. Network-on-Chip Architecture

1.1. Introduction to NoCs

This thesis focuses on cross layer fault tolerance in Network-on-Chip (NoC) based Systems-on-Chip SoCs. This chapter describes the basics of concept of NoCs and introduces the Bonfire router which is an in-house NoC router. The decision to develop an in-house NoC router was twofold: 1) to provide very simple hardware infrastructure in order to reduce unnecessary complexity and 2) to design a small baseline NoC router in order to magnify the effect of area overhead of the proposed approaches. The rest of this section describes the preliminaries of NoCs.

In recent decades NoCs have emerged as a promising communication paradigm for replacing traditional and modern buses. NoCs provide a flexible solution that reduce the wire-length in the chips by using a network of routers in the chip instead of long bus wires.

The NoC topologies can be either *direct* or *indirect*. In a direct topology, each tile consists of a router/switch and a resource. However, in an indirect network, each network tile consists of router/switch, but may or may not include a resource [19]. Fig. 1.1 depicts an example of a direct network while Fig. 1.2 provides an example of indirect networks.

This work focuses on *direct* networks. More specifically, in this work, each tile in a NoC architecture consists of a *router*, a *network interface* and a *resource*. The *router* receives the information in form of packets from the network interface or neighboring routers and passes the information around to adjacent routers or the network Interface. The *network interface* acts as an adapter that connects the tile's router to the tile's resource. Usually the network interface is in charge of making packets out of the information received from the local resource and also to de-construct the packets into raw data to be consumed by the local resource. The *resource* in the network tile can be itself a small Bus-based SoC, a single processor, a shared memory etc.

The Information is transfered in the network in form of messages which consist of one or several packets. Each packet consists of several flits. As a common practice, the following types of the flits are used: Header which contains the routing information, Body flit which contains the payload and the

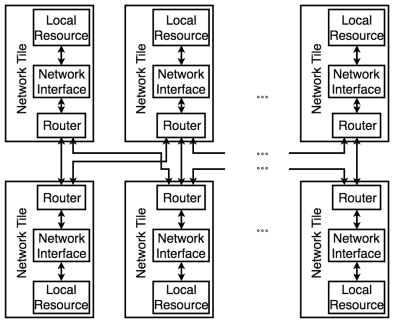


Figure 1.1 An example of a direct network

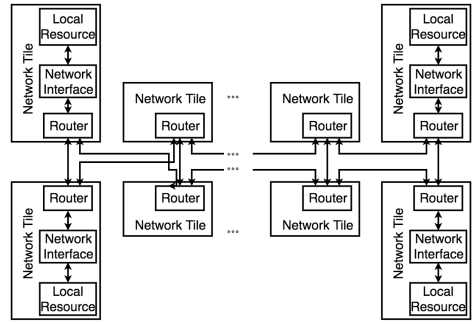


Figure 1.2 An example of an indirect network

Tail flit that ends the packet. A packet can only have one header and one tail flit but can contain several body flits. Each flit in turn consists of one or several phits. Where a phit is the amount of data that can be transfered over the physical links in one clock cycle[19].

The NoC routers can either be bufferless or have input and (or) output buffers. The bufferless routers are smaller (in terms of area) but in case of network congestion, the packets would be either miss-routed or dropped. This in turn imposes a considerable latency in the communication. On the other hand, buffered routers suffer from large area.

Allocation of the network's resources to packets is managed by the network's flow-control mechanism. There exist many approaches for buffer management, some of the most common methods are: store-and-forward, virtual cut-through and wormhole switching [19]. In store-and-forward, the buffers of the router are as big as the largest packet size allowed by the system. A packet is forwarded to the adjacent router or the local resource once the entire packet is stored in the router's FIFO. This approach requires huge buffer sizes while the network has a high latency factor, since a packet's flits need to wait in router's buffer. To improve the latency of store-and-forward, virtual cut-through method was incorporated. In this method, packets can leave the router as soon as they reach the router, but the router inputs buffer are as large as the largest packet supported by the system. While virtual cut-through improves the latency of store-and-forward, it will result in a inefficient buffer management; Usually most of the routers will stay unused during the operation. To improve this, wormhole switching is used. Wormhole switching allows the departure of the flits as soon as they arrive to the router but also has smaller buffers. In case of congestion in the network, the flit's of the same packet will be stored in the routers buffers along the packet's path.

In order to manage the allocation of network's physical links to the packets, a communication protocol is required between the adjacent routers. There exist

many protocols such as acknowledge based, on-off and credit based schemes (for more information please refer to [19]).

Another important factor in NoCs is the routing management. Routing of packets can be either done using *table-based routing*, *source routing* or *algorithmic routing*. In *table-based routing*, each router has a routing table which contains the routing information for each source-destination. This approach is not scalable since the size of the table will grow with network dimension's power of two. *Source routing* approach includes the packet's path (in form of directions that a packet needs to take in each step) in the packet. Each router will remove the part of the processed direction from the packet and forward the packet further. This approach adds considerable overhead to the packet's size. *Algorithmic routing* implements some routing function in hardware that can calculate the direction of packet based on the source and destination information in the packet[19].

While routing a packet in the network, its possible to end up in a *deadlock* or *live-lock* situation. A *live-lock* is the situation that a packet is moving around the network but never reaches its destination. A *deadlock* situation describes the case that several packets form a circular dependency where all are serially waiting for the packet ahead to release a network resource. Both of these cases should be handled properly by the routing algorithm. Specifically, deadlock is handled either by ensuring that it can never happen (deadlock avoidance) or by resolving a deadlock that has already occurred (for more information see [19]).

The rest of this chapter will focus on the Bonfire framework, a NoC based SoC which is used in rest of this thesis.

1.2. Bonfire Framework

The work in this thesis uses the Bonfire framework as a hardware demonstrator. Bonfire [20] is an open-source framework for testing dependability mechanisms in a NoC-based System-on-Chip (SoC). The target NoC architecture is a 2D mesh topology. Each network tile consists of a wormhole switching router equipped with fault tolerance mechanisms and a Processing Element (PE) connected to it via a Network Interface (NI). Each PE comprises a Plasma core [21]; a 32-bit MIPS-I based open-source pipelined processor. Two different versions of Bonfire router have been used in this work; A simple architecture using handshaking flow-control which only supports deterministic turn-model [22] based routing. And a more sophisticated router with credit-based flow-control which supports any turn-model-based adaptive routing algorithm [22]. More details of these architectures are provided bellow:

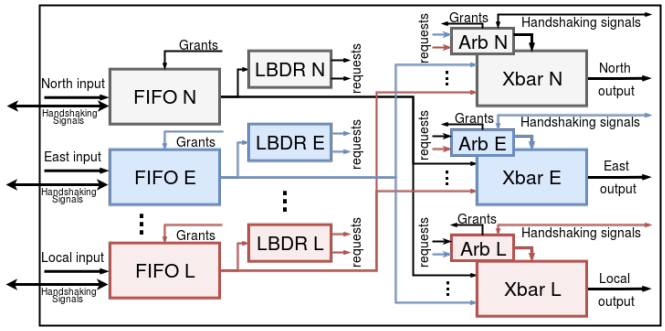


Figure 1.3 Block diagram of the hand-shaking router

3 bits	12 bits	4 bits	4 bits	8 bits	1 bit
Header Flit Type	Packet length	Source Address	Destination Address	Packet ID	parity bit
Body Flit Type	Payload				parity bit
Tail Flit Type	Payload				parity bit

Figure 1.4 Packet format in hand-shaking router.

1.2.1. Router Architecture with Handshaking Flow-Control

The Bonfire router with handshaking flow-control is a simple wormhole NoC router, which is not designed for high performance communication. Fig. 1.3 depicts the block diagram of hand-shaking router. It consists of Input buffer unit, Routing logic , Arbiters and Crossbar Switches (marked as $FIFO\_x$, $LBDR\_x$, Arb_x and $Xbar\_x$ in Fig. 1.3 respectively). The input buffer Unit consists of a 4-flit deep circular buffer FIFO and performs handshaking with the upstream router. The Arbiters provide support for deterministic routing and perform handshaking with the downstream router. Two neighboring routers have to complete a three stage handshaking for each flit transmission. The handshaking process will force a upper-limit for the network throughput (one third of its maximal performance). However, this router was used as a simple design example. This architecture will be referred to as HS router hereafter. The routing logic is implemented using LBDR [23] which is a lightweight distributed routing mechanism that supports any turn-model based routing algorithm. This router is used explicitly in the work presented in Section 2.2.1.

The packet format for this architecture consists of 3 types of flits; Header flit, Body flit and tail flit. Header flit consists of source and destination address (4 bit each), packet identification number and packet length counter. Body and tail flits only contain payload data. Each flit has a three bit one-hot encoded flit-type tag. The packet format used in this router is depicted in Fig. 1.4.

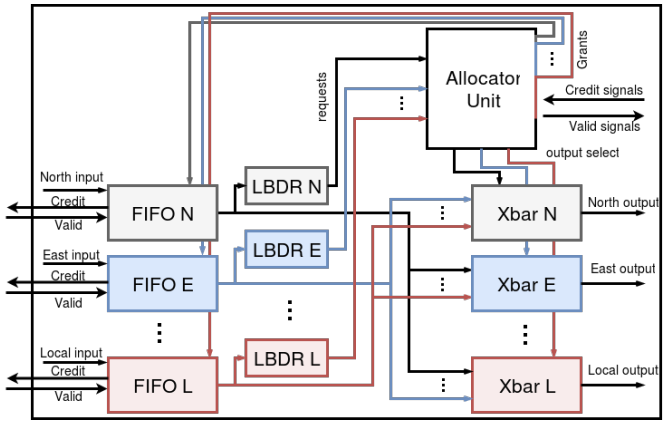


Figure 1.5 Block diagram of the credit based router

3 bits	14 bits	14 bits	1 bit
Header Flit Type	Source Address	Destination Address	parity bit
Body 1 Flit Type	Packet length	Packet ID	parity bit
Body Flit Type	Payload		parity bit
Tail Flit Type	Payload		parity bit

Figure 1.6 Packet format in credit based router.

1.2.2. Router Architecture with Credit Based Flow-Control

The Bonfire router with credit-based flow-control is a more sophisticated NoC router with wormhole switching. Fig. 1.5 shows block diagram of the credit based router used in this work. This router contains input buffers and routing units for each input port, one switch allocator that provides support for any turn-model based adaptive, minimal-path routing algorithm and a crossbar switch for each output port (marked as $FIFO\_x$, $LBDR\_x$, Allocator Unit and $Xbar\_x$ in Fig. 1.5 respectively). Design of the allocator unit is based on [19]. As for the flow-control, each upstream router, keeps track of empty buffers in the downstream router and sends the flits with the assumption that the downstream router can receive them. The FIFO unit utilizes a 4-flit deep circular buffer and is in charge of issuing credit signal to the upstream router. The routing logic is using LBDR [23] which is a lightweight distributed routing mechanism that supports any turn-model based routing algorithm. This architecture will be referred to as CB router hereafter.

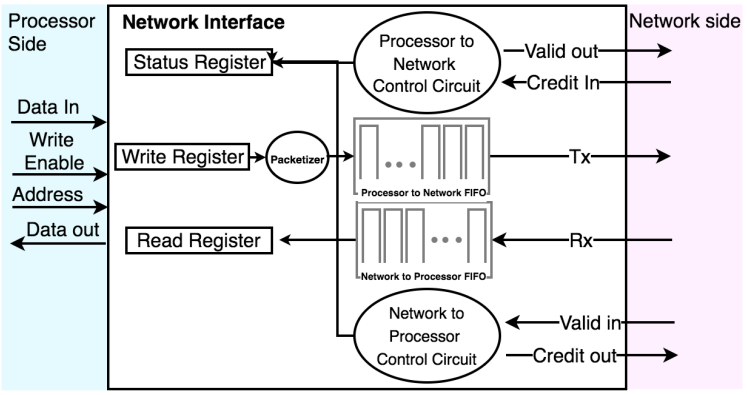


Figure 1.7 Block diagram of the Credit-based Network Interface

Packet format is slightly modified compared to the handshaking router. The header flit contains only source and destination address-each being 14 bits long enabling addressing up to 128×128 networks-. First body flit, contains the packet identification number and packet length. The rest of body flits and the tail are only carrying payload data similar to HS router. Each flit has a three bit, one-hot encoded flit-type tag. Fig. 1.6 depicts the credit-based router packet format.

The Network Interface (NI) designed for this router is a memory-mapped unit connected to the Plasma processor. The network interface includes a packetizer and two FIFO buffers (one from the processor towards the network -PE2NoC- and one in the opposite direction-NoC2PE-). The depth of the PE2NoC is set to 32 flits however for calculating the latency and throughput of network, this number is increased up to 32768 flits (just for simulation purposes). The NI does not provide a depacketization service and it is assumed that this task will be handled in software. Fig. 1.7 illustrates the block diagram of the Network Interface and Processor and Network side signals.

1.3. Evaluation of base-line Bonfire routers

To evaluate the characteristics of Bonfire routers, this section provides data regarding the area of each router. All routers are synthesized by AMS $0.18\mu m$ library.

Table 1.1 Area break-down results of Bonfire Routers

Router name	Combinational Area (μm)	Sequential Area (μm)	Total Area (μm)
Hand-Shaking Router	42801	48361	91163
Credit-based Router	47969	52059	100028

Table 1.2 Handshaking router units area breakdown

Unit name	Combinational Area (μm)	Sequential Area (μm)	Total Area (μm)
FIFO	5105	8893	13999
LBDR	211	324	536
Arbiter	601	259	860
Xbar	2173	0	2173

Table 1.1 shows the combinational and sequential area of handshaking and credit-based routers. As expected, more than half of the area of the router is consumed by the sequential elements, specially the FIFO unit's buffer. Tables 1.2 and 1.3 show the unit's area break down for handshaking and credit-based routers respectively. The main difference between the two is the presence of an allocator unit in credit-based router where in the handshaking router only five arbiters were used before. However, the biggest share of the router area is consumed by the FIFO units in both designs. The area increase in the FIFO unit in credit-based router is due to additional credit counters. Also the increase in LBDR unit of the credit based router is due to additional hardware support for enabling routing reconfiguration.

Table 1.3 Credit-based router units area breakdown

Unit name	Combinational Area (μm)	Sequential Area (μm)	Total Area (μm)
FIFO	5464	8893	14357
LBDR	745	999	1744
Allocator Unit	6677	2596	9274
Xbar	2173	0	2173

1.4. Chapter Summary

In this chapter some basics of the Network-on-Chip paradigm was discussed. Also, this chapter provided details of Bonfire, the in-house developed framework, which is used throughout this thesis. Bonfire architecture being a simple and lightweight router, enables research on NoC dependability by: 1) removing the unnecessary complexities and 2) magnifying the area overhead of the fault tolerance mechanisms. Finally this chapter concluded by area evaluation of Bonfire framework.

2. Fault Information acquisition

One of the most critical parts in a cross-layer dependability scheme, is acquisition of useful fault information in each abstraction layer. Many approaches exist in the literature which consider fault detection in software-layer etc.. However, application-level fault detection usually is possible once a failure has happened. And fixing it requires a roll-back to a correct checkpoint. In contrast, acquiring fault information in the hardware would provide the system with possibility of faster reaction –with isolation and/or correction being totally transparent from the user. Its important to note that once the detection of the faults is moved to hardware, it is possible that the fault information produced is too detailed which would require an abstraction stage before being used. Another advantage of hardware-level fault information acquisition is the possibility of processing this information on site which would be many times faster than doing it at higher abstraction layers. To acquire such information in the hardware level, four essential tasks should be performed: detection of the faults, abstraction of fault information, classification of the faults and finally propagation of such information to the consumer. Consumer of such information being either a system level resource manager or a local fault management unit or a hybrid of these approaches.

This thesis focuses on the faults in the routers in a NoC based SoC. The proposed approach divides the NoC router into data-path and control-part and uses different existing fault detection mechanisms to address the faults in these sub-circuits. To detect the faults in the control-path of the router, the concept of checkers has been used. However, while generating such checkers, the area constraints of the final product is not considered (the focus is more on the fault coverage). Also, during this process, it is very likely that multiple checkers are generated which provide overlapping fault coverage. This problem requires a fully automated minimization framework for the area of such checkers (while maximizing the coverage).

Once the fault information is gathered and abstracted before being used by the higher abstraction layers ($e.g.$ a software-based global system health manager), It should be classified into transient, intermittent and permanent. Moving the classification task to hardware provides the system with the following advantages: (1) the classified fault information is already available in

the hardware layer and can be used by local fault mitigation mechanisms, (2) reduces the software load for such tasks. The idle software time can be utilized for other dependability mechanisms such as dependable application deployment etc. The current state-of-the-art solutions for online fault classification are expensive and not scalable for many fault signals provided by fault abstraction mechanisms. A light-weight fault classification mechanisms is required that enables massive deployment in the routers.

This chapter provides the following contributions:

- A framework for minimization of concurrent online checkers
- Extension of the current online fault classification methods

This chapter is based on the following publications:

- ”From online fault detection to fault management in Network-on-Chips: A ground-up approach,” S. P. Azad, B. Niazmand, K. Janson, N. George, A.S.Oyeniran, T. Putkaradze, A. Kaur, J. Raik, G. Jervan, R. Ubar, T. Hollstein, DDECS 2017
- Automated area and coverage optimization of minimal latency checkers, S. P. Azad, B. Niazmand, A. K. Sandhu, J. Raik, G. Jervan and T. Hollstein, ETS 2017

2.1. Literature Review

There has been considerable research on NoC fault diagnosis. While works like [24, 25, 26] only target faults on the links, Other approaches target a broader space and cover different parts of the router [27, 28] or network [29]. Another point to distinguish these approaches is the timing of the process, hardware BIST approaches such as [29, 30, 27, 26] require a full or partial shutdown of the system while methods like [31, 32, 33, 34, 35, 36, 37, 38, 39] provide on-line fault diagnosis. Its also important to note software based BIST methods such as [25] which use test packets to test the links. The BIST methods have proven to be very efficient and compact however, since they do not provide on-line fault detection, the fault can not be handled locally and there is always the chance of fault information propagation to the rest of the system. On the other hand, the area and power overhead imposed by the on-line fault detector mechanisms is considerable and there is much room for improvement.

In cases that the fault information are not consumed locally, the granularity of the information propagated to the consumer should be adjusted accordingly. For example, the system level manager does not require information about a failing logic gate in a part of the system to make a decision. In fact, if

the information is too detailed, it makes the task for the software level more difficult since it has to make the abstraction first. Also its important to consider the amount of power consumed when transferring all the information to the system manager. To this end, it is crucial to combine the fault information locally and propagate the absolute minimum required.

The importance of fault classification can not be understated in these approaches. BIST approaches can in-fact provide fault classification [40] by repeating the test. However, using BIST-like approaches require the system to be switched fully or partially from operation mode to test mode which will lead to considerable loss of performance. Another approach would be to transfer all the fault information to the software layer [41, 42] and leave the classification task to those units. There are two main problems with this approach; The latency and the communication overhead. Its much faster to perform the task on-line and locally and send the classified fault information to the system manager. Methods like [43] and [44] have made realistic attempts to provide on-line hardware support for fault classification. However, [43] suffers from a short screening window for the faults, and [44] incurs large area overhead. Both of these methods cover the faults on the links and there is no pragmatic solution available which provides classification for all the faults in the NoC router.

There are different approaches for propagating the fault information to a central unit in the system; scan based approaches such as [41, 42, 45], secondary network systems [46] or through the NoC itself [47, 48]. Each of the above mentioned approaches has its own bottlenecks; Scan based approaches have very limited number of access points which limits the mapping application choices for deploying central manager tasks. Secondary network solutions provide a fixed amount of overhead to the network but provide a scalable and very flexible solution. However, considering the size of the secondary network (which is not negligible) considerable amount of faults might happen inside this network, which makes it a reliability issue. Lastly using the NoC itself for sending diagnosis packets has a very low hardware overhead. However, this method suffers from the fact that due to previous network faults, diagnosis packets might not reach the central manager unit.

2.2. Fault Detection

Fault detection comes as the first step of producing fault information. However, this work addresses fault detection on datapath and control path of the system differently.

In this work the data path faults are detected using single parity checker. The parity checker has been chosen due to its low area overhead and its power to detect all odd number of faults. The entire router data-path is divided into

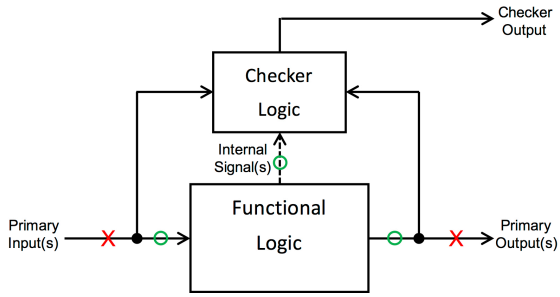


Figure 2.1 concept of a checker circuit

two sections, the path from the upstream router to the input of the FIFO and from the output of FIFO to the output port. These points are critical locations since the data in the input of the FIFO is used for generating the credits to the upstream router and the data on the output of the FIFO is used by the routing unit (in this case LBDR unit) for routing decisions. A simple parity checker module is used to cover detection of all single-event upsets on the inputs and outputs of the FIFO units. The single parity is utilized due to its simplicity and low area overhead compared to other approaches such as hamming code [14]. However, based on the system requirements, a stronger error detection code can be used (for example, [49] has evaluated different ECC techniques for hardening FIFO in a NoC router).

In the following subsection the fundamentals of control path fault detection is laid out.

2.2.1. Control part fault detection

The work introduced in this section builds on the concept of concurrent online checkers introduced in [2, 3, 4]. The concept of checkers are used since they provide 100% coverage of the Stuck-at-Faults in the control part of the router concurrently which can be used to enable online fault handling in the router.

Fig 2.1 describes the placement of concurrent online checkers; A checker is a combinatorial circuit that reads the inputs and outputs and some internal signals of combinational part of a unit and checks the validity of the internal and output signals based on the received input. Checker circuit can detect all the faults happening on the locations marked in Fig. 2.1 with green circles. However, the faults on the locations marked with red crosses can not be detected by the checkers. Since either both the checker and the circuit are receiving faulty input or the fault is not propagated to the checker logic and directly passed to the output.

Table 2.1 classifies the faults in the unit under checking, into four classes based on two factors: i) whether the fault propagates to an output of the

Table 2.1 Different conditions of fault propagation and detection by checkers and their notations

Fault propagated to outputs	Fault detected by the checkers	Condition Name	Notation
✓	✓	True Detection	D
✓	✗	True Miss	W
✗	✓	False Positive	F
✗	✗	Benign Miss	X

circuit and ii) whether the checker can detect the fault. Using the notations in Table 2.1, the Coverage Efficiency Index (CEI) of a checker is calculated as:

$$CEI = \frac{D}{D+W} \tag{2.1}$$

Equation 2.1 only includes the cases that the fault has propagated to the output of the circuit and either captured by checkers (True Detection-D) or not captured (True Miss-W), hence Benign Misses (X) are excluded. Since none of the checkers resulted in false positives firing, False Positives cases (F) are excluded from the metric. The checker generation methodology is out of scope of this thesis, however, the full list of checkers for Handshaking-Router used in this work is provided in Table 2.2.

Checker minimization framework

While checkers provide concurrent online fault detection for the units combinatorial part, they tend to add considerable area overhead to the circuit. To this end, the framework introduced in [2, 3, 4] has been extended to provide different minimization schemes for area of the checkers (see Fig. 2.2).

The framework starts with the *"control part of circuit"*. The next step is *"devising pseudo-combinational version of the circuit"*. During this step the sequential elements would be removed and the inputs and outputs of the sequential components would be added to the circuits interface as pseudo-inputs and pseudo-outputs. Later the *"initial set of checkers"* would be devised for the pseudo-combinatorial part of the circuit. Next, are synthesizing the circuit along with *"checkers synthesis"*. This step will provide *"synthesized pseudo-combinational version of the circuit"* and *synthesized initial set of checkers*. In parallel, the framework includes the *"generation of exhaustive set of valid input stimuli"* for testing the behavior of the checkers (this steps will be called *"environment generation"*). The next step of the process is *"fault-free simulation"* of the checkers. In this step the synthesized checkers and the synthesized combinatorial part of the circuit will be simulated (without presence of any faults) with exhaustive input patterns. This simulation step is performed for evaluating fault-free firing of checkers. In the next step the

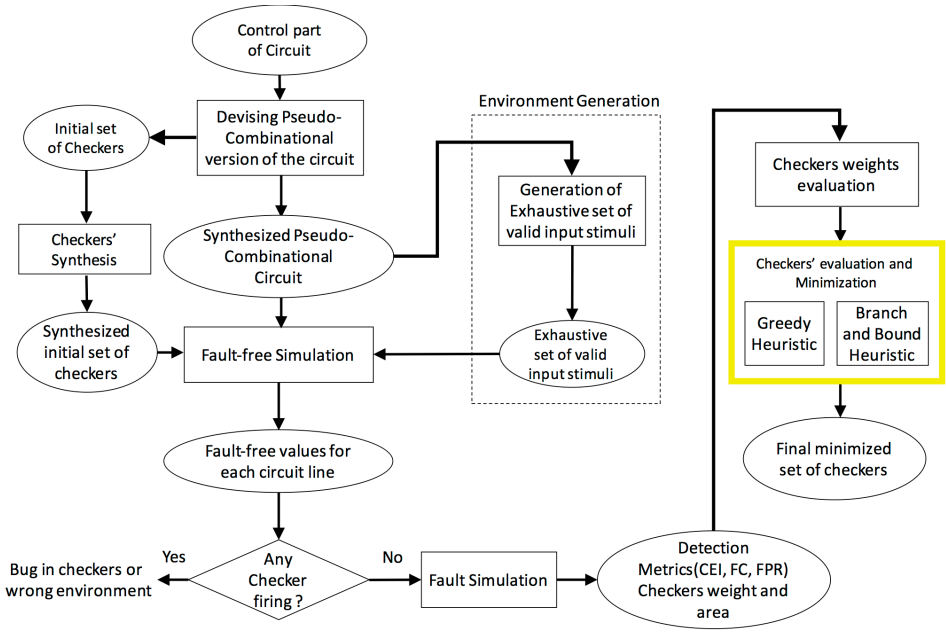


Figure 2.2 Extended checker framework introduced in [2, 3, 4]

"fault-free values for each circuit line" is extracted. Later in the process the user will check if any fault free firing has happened (labeled as *"Any Checkers firing?"* in the Fig. 2.2). Firing of a checker in this part means that either the checker circuit or the input stimuli is incorrect. Once all the fault-free firing cases are removed, the checkers would be evaluated to extract their effectiveness using CEI metric along with their Fault Coverage (FC), False Positive Ratio (FPR), Checker's Weight and Area (for details regarding these metrics please refer to [2, 3, 4]). These values (except area which is calculated during synthesis process) are extracted through *"fault simulation"* using [50]. In the next step, *"checkers' weights evaluation"*, the checkers weights would be compared and initial analysis would be performed by the user. Later the checkers would be fed to the *"checker's evaluation and minimization"* framework introduced in this work for optimization (using *"greedy heuristic"* and *"branch and bound heuristic"*). The *"final minimized set of checkers"* with minimum area overhead and maximum detection would be extracted. This work focuses on the checkers minimization framework (highlighted in Fig. 2.2)

The proposed framework starts from initial set of checkers provided in table 2.2 for bonfire's HS router. The numbers in the first column are the numbers assigned to each checker (these numbers would later be referred to in the end of this section) and the description on the right column, introduces the functionality of each checker. It is important to note that this information is

Table 2.2 Description of the initial set of checkers for each module for Bonfire router with Handshaking Flow-Control

Checker number	Checkers for Routing Logic (LBDR)
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).
Checkers for Arbitration Logic (Arbiter)	
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.
Checkers for FIFO control part logic	
1, 2	Depending on the value of the write enable signal, the write pointer of FIFO's control part must update accordingly (one-hot).
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).
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.
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.

provided for reference and the topic of devising checkers is beyond the scope of this work (for more information please refer to [2, 3, 4]). Two optimization algorithms (greedy and branch-and-bound) are proposed for minimizing the area of the checkers.

Greedy heuristic

A greedy heuristic has been proposed for finding good quality solutions (not

Algorithm 1: Greedy algorithm pseudo code

```
initialization;    // evaluates individual checkers and stores them in a list with
   their CEI
sorted_checkers = sort_checker_based_on_CEI()
best_CEI = 0
current_list = []
if check_feasibility(current_list) then
    for checker in sorted_checkers do
        if item not in current_list then
            temp_list = current_list + [item]
            if check_feasibility(temp_list) then
                CEI = calculate_CEI(temp_list)
                if CEI > best_CEI then
                    current_list.append(item)
                    best_CEI = CEI
                end
                if CEI == 100% then
                    break
                end
            end
        end
    end
end
return current_list
```

necessarily global optimal) in few number of steps (In this case the number of steps are equal to number of checkers). Algorithm 1 describes the process of greedy algorithm for minimizing the checkers. This algorithm uses Checkers' Efficiency Index (CEI) for sorting the checkers and then tries picking them from the top of the list. $check\_feasibility$ function will check if the a set of checkers violate area constraints set by user. the chosen checker would be evaluated along with best found checker set ($current\_list$) and if the CEI of the new set is providing any improvement to the best found checker set, The checker would be added to the list. Once the CEI value reaches 100%, the process terminates.

The main problem with the greedy approach is the fact that usually checkers with higher CEI tend to have higher area as well. As an example consider three checkers C_1, C_2 and C_3 with the areas A_1, A_2 and A_3. These checkers would have the coverage values of CEI_1, CEI_2 and CEI_3 respectively. However, It is possible that $A_1 > A_2 + A_3$ but at the same time $CEI_1 <= CEI_{\{2,3\}}$, where $CEI_{\{2,3\}}$ describes the overall coverage of the set $\{C_2, C_3\}$. This can happen due to the overlapping nature of the checkers. This problem will lead to the algorithm to pick larger checkers in the beginning and which will lead to non

optimal solutions. To solve this issue, The following approach is introduced in this work:

Coverage density as a sorting metric:

After investigating the checkers coverage and area, one of the key findings was that the larger checkers tend to cover more faults, In this sense, if the optimization starts with checkers with largest CEI value, the area overhead grows dramatically in the beginning. To remove the advantage of the bigger checkers, a new metric namely Coverage Density (CD) is introduced as the sorting factor:

$$CD = \frac{CEI_{checker}}{Area_{checker}} \tag{2.2}$$

CD metric shows the fault coverage capability of the each checker in one unit of its area. Using CD as the sorting factor, The algorithm starts with including checkers with the most valuable area units. Using CD as sorting factor, the greedy algorithm provides higher quality solutions under more strict area constraints. Here, the routing logic of the router can be used as an illustrative example. In case of applying area constraint of $77\mu m$ for the checkers (which is equal to the size of the original unit, forcing the checker unit to apply maximum 100% area overhead). Table 2.3 describes the results for different sorting methods:

Table 2.3 Evaluation of greedy algorithm's results for LBDR module's checkers for different initial checkers' sorting method

Control Unit	Area Constraint (# NAND)	sorting method	Selected Checkers	coverage	Area (# NAND)	Overhead (%)
LBDR	77	CEI	1, 2, 3, 4, 7, 8, 10, 11, 12, 16	87.9%	77	100%
		CD	1, 2, 3, 4, 5, 7, 8, 10, 11, 12	89.2%	76	98.7%

Table 2.3 describes the results in terms of selected checkers set and the resulting coverage of the set for different optimization sorting methods (i.e. CEI and CD). Also the area results and area overhead of each of the sets are reported. In this case, the greedy search using Coverage Density (CD) results in much better coverage results with lower area overhead. However, the experiments show that in cases that have stricter or much looser area constraints, both metrics act similarly.

In the end, greedy algorithms provide local optima for the problem and can not guarantee the quality of the solution. To provide such global solutions in the search space, a better optimization heuristic is needed. The next section covers the implementation and evaluation of Branch and Bound algorithm for this problem.

Branch and Bound algorithm

Greedy algorithms are fast and provide a solution with few number of steps.

	Node Numbers																																								
Checker's numbers	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	120	120	264	276	ND	64	ND	ND	138	138	ND	ND	ND	ND	ND	0	ND	ND	0	ND	0	ND	0	ND	ND	0	0	ND	ND	0	0	ND	ND	0	0	0	ND	ND	ND	ND	ND
2	ND	ND	ND	ND	0	ND	ND	ND	ND	ND	ND	ND	ND	160	88	ND	32	32	ND	88	ND	176	ND	32	32	ND	ND	32	32	ND	ND	32	32	ND	ND	ND	32	32	ND	ND	ND
3	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	0	ND	60	ND	ND	ND	ND	60	ND	120	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND
4	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	80	68	ND	16	16	ND	68	ND	136	ND	16	16	ND	ND	16	16	ND	ND	16	16	ND	ND	ND	16	16	ND	ND	ND
5	ND	ND	ND	ND	ND	ND	ND	ND	ND	0	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND
6	ND	ND	ND	144	ND	ND	ND	ND	0	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND
7	ND	ND	0	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND
8	ND	ND	ND	ND	0	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND
9	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND
10	0	0	ND	ND	0	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND
11	ND	ND	ND	0	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND
12	ND	ND	ND	ND	0	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND
13	ND	282	138	ND	282	282	426	ND	60	ND	ND	138	138	ND	ND	ND	ND	18	ND	18	0	0	0	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND
14	ND	ND	282	282	ND	65	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND
15	138	ND	282	ND	282	426	ND	60	ND	ND	ND	ND	ND	ND	ND	ND	18	ND	18	0	0	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND
16	ND	ND	ND	ND	240	ND	32	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	0	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND
17	ND	ND	ND	ND	360	ND	48	ND	ND	ND	ND	ND	ND	ND	ND	ND	0	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND
18	132	ND	276	276	276	276	420	ND	56	ND	ND	ND	ND	ND	ND	ND	ND	12	ND	12	0	0	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND

Figure 2.3 Example of number of True Misses for stuck-at-0 for LBDR circuit nodes.

	Node Numbers																																								
Checker's numbers	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	120	120	264	276	64	ND	64	64	138	138	ND	ND	ND	0	0	0	ND	32	0	0	0	0	0	ND	32	216	0	ND	32	108	0	ND	32	0	0	108	ND	32	ND	ND	ND
2	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	160	ND	ND	ND	ND	ND	ND	160	ND	212	ND	160	ND	106	ND	160	ND	ND	ND	106	160	ND	ND	ND	ND
3	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	0	0	ND	ND	ND	ND	ND	12	ND	ND	ND	ND	ND	ND	8	156	ND	ND	14	78	ND	ND	15	ND	ND	78	ND	15	ND	ND	ND
4	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	80	ND	ND	ND	ND	ND	ND	80	ND	172	ND	80	ND	86	ND	80	ND	ND	ND	86	80	ND	ND	ND	ND
5	ND	ND	ND	144	ND	ND	ND	ND	0	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND
6	ND	ND	ND	ND	ND	ND	ND	ND	ND	0	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND
7	ND	ND	ND	ND	0	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND
8	ND	ND	0	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND
9	0	0	ND	ND	0	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND
10	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND
11	ND	ND	ND	ND	0	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND
12	ND	ND	ND	0	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND
13	ND	276	138	138	ND	276	138	60	ND	60	60	138	ND	ND	ND	ND	49	18	18	18	0	0	108	ND	32	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND
14	ND	ND	ND	ND	65	ND	65	65	ND	ND	ND	ND	ND	49	18	ND	18	0	0	108	ND	32	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND
15	138	138	ND	ND	276	138	60	ND	60	60	ND	ND	ND	ND	ND	49	18	18	18	108	0	ND	32	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND
16	ND	ND	ND	ND	240	32	ND	32	32	ND	ND	ND	ND	ND	56	ND	ND	ND	0	0	ND	32	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND
17	ND	ND	ND	ND	120	48	ND	48	48	ND	ND	ND	ND	ND	52	0	0	ND	32	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND
18	132	132	ND	276	276	264	132	56	ND	56	56	ND	ND	ND	ND	ND	50	12	12	12	216	0	ND	32	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND	ND

Figure 2.4 Example of Nnumber of True Misses for stuck-at-1 for NLBDR circuit nodes.

However, as mentioned in the previous section, greedy algorithms do not provide any guarantee on the quality of solution. Branch and Bound algorithm provides the global optimal solution but in the worse case scenario needs larger number of steps. If there are N checkers, the number of steps to be checked in the worst case scenario is $2^{N+1} - 2$ which is the case of traversing the full tree. This trade-off between quality and speed will determine which method to be used.

Depth First Search (DFS) implementation of *Branch and Bound* algorithm was used 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 performed (which is the CEI of all the remaining checkers without considering the area constraint). In case the optimistic evaluation of the current step is smaller than the best solution found so far (i.e. the solution is *bounded*), the search would be terminated in that *branch*. The process will return the set of checkers with the least area consumption and largest CEI. As already mentioned, this method can provide the global optima in the search space, however,in the circuits with large number of checkers this approach becomes in-efficient.

Dominant checkers:
while evaluating the checkers, two main characteristics for each checker are the *number of True Detections* (for stuck-at-0 and stuck-at-1) and *number of True Misses* (for stuck-at-0 and stuck-at-1) for each circuit line. if all the checkers are evaluated separately, four tables can be generated; two tables for True Misses and two for True Detections. Fig. 2.3 and 2.4 shows examples for true misses tables for stuck-at-0 and stuck-at-1 for the routing logic (LBDR) respectively. The columns show the node numbers in the synthesized circuit and the rows show the checker id numbers. the numbers in the table show the number of true misses of each checker for each circuit node. the cells marked with green shows the cases where there exists a single dominant checker and cells marked with red are dominant but are not unique checkers. Nodes that have zero true detection are marked with ND (not detected) If these tables are processed individually 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 is possible to extract checkers which provides the smallest values of True Misses for each line in the circuit, hence improving the CEI.

Definition 1. *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 other 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.

Optimization results evaluation:
This section provides the experimental results for the optimization algorithms on the HS router and compares the effects of using dominant checkers and different optimization algorithms.

Table 2.4 shows the area of the full units, pseudo combinatorial circuit, area and hardware overhead of DMR and TMR for control units along with the number, area and hardware overhead of the full set of the checkers. The overheads are calculated against the full unit size. From the results, it is clear that the original checker mostly consume larger area comparable to TMR (except in the case of FIFO).

Table 2.5 shows the optimization results without the use of dominant checkers along with the search space size. As it is shown in the table, the search space size can grow dramatically even for small circuits such as arbiter.

Table 2.4 Area comparison of complete concurrent online checkers set with original, DMR and TMR versions of units' control path

Unit Information									
Control Unit	Control unit area ($\#NAND$)		Control unit DMR		Control unit TMR		Complete set of checkers		
	Full unit	pseudo comb.	Area ($\#NAND$)	Overhead (%)	Area ($\#NAND$)	Overhead (%)	Number	Area ($\#NAND$)	Overhead (%)
LBDR	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%

Table 2.5 The result of checkers optimization without application of area constraints without use of dominant checkers.

Optimization without Dominant checkers					
Control Unit	Optimized set of checkers				
	Opt. method	Selected Checkers	Area ($\#NAND$)	Overhead (%)	Search space size
LBDR	B&B	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18	99	128%	262144
	Greedy	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 16, 17, 18	111	144%	
Arbiter	Greedy	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 21, 22, 26, 31, 32, 36, 37, 42, 45, 46, 52, 53, 54, 55	261	150%	7×10^{16}
FIFO	B&B	1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 13	120	93%	8192
	Greedy	1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13	122	94.5%	

Table 2.6 Full list and area, overhead and coverage results of dominant checkers for each module

Dominant checkers for each unit				
Unit	dominant Checkers	Area ($\#NAND$)	Overhead	Coverage
LBDR	1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18	121	157.1%	89.6%
Arbiter	1, 2, 7, 8, 9, 13, 14, 17, 31, 45, 46	200	114.9%	89.8%
FIFO	1, 2, 3, 5, 7, 8, 13	105	81.3%	92.5%

The results reported in Table 2.5 for greedy algorithm are obtained using CEI as sorting factor. As its clear, for units with large number of checkers (such as Arbiter), search space for Branch-and-Bound algorithm grows significantly and becomes in-efficient. However, it provides global optima for smaller problems (such as LBDR and FIFO) and is preferable.

Full list of dominant checkers for each unit and the characteristic of the dominant checker set are presented in Table 2.6. Complete dominant checkers set already covers at least 89% of all modules faults.

Finally the optimization results using the dominant checkers in the starting point along with the search space size are reported in Table 2.7. By comparing the search space size of the algorithms using *Dominant Checkers* with normal case, the search space becomes much smaller and the provided solution is still acceptable. The experimental results show that using the proposed framework, it is possible to keep the area overhead between DMR and TMR of the control logic and still provide 100% coverage. This result is important due to the fact

Table 2.7 The result of checkers optimization without application of area constraints using dominant checkers.

Optimization with Dominant checkers					
Control Unit	Optimized set of checkers				
	Opt. method	Selected Checkers	Area ($\#NAND$)	Overhead (%)	Search space size
LBDR	B&B	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18	123	159%	1
	Greedy				
Arbiter	Greedy	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 21, 22, 26, 31, 32, 36, 37, 42, 45, 46, 52, 53, 54, 55	266	152.8%	3.5×10^{13}
FIFO	B&B	1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 13	120	93%	64
	Greedy	1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13	122	94.5%	

that the checkers can provide localization information which DMR and TMR can not provide.

2.3. Fault information abstraction

Despite the importance of providing localization information to higher layers, too detailed fault information can not be handled efficiently by a system wide manager. To this end the minimized checkers outputs can be grouped to generate abstract information such as Unit-Faults and Turn-Faults in the router.

Definition 2. *A turn is defined as a path from one input port of the router to an output port of the same router.*

Using turn-fault abstraction, it is possible to reduce the number of fault locations from hundreds of circuit lines to twenty turns including eight 90 paths (for example from East to North) and twelve straight paths (for example from South to North). To generate such abstract information, signals from the different unit's checkers in the router are combined (OR-ed together). This abstracted information can be used by system manager which keeps abstract model of the communication architecture in order to perform system-wide reconfiguration. Later on, the classification of the faults would be performed on the abstracted checker information. The fault information abstraction is beyond the scope of this thesis (For more information, please refer to [14]).

2.4. Fault Classification

Classifying the faults in a circuit involves monitoring the frequency of the faults in a circuit. There should also be two thresholds for fault occurrence which is used for discriminating the units into different classes (Healthy, Intermittent and Faulty). This can not be done merely with counting faults in a time window. An example can be used for illustrating this point; Assume two units A and B. Fault detectors of unit A fault fire once every millisecond and fault detectors of unit

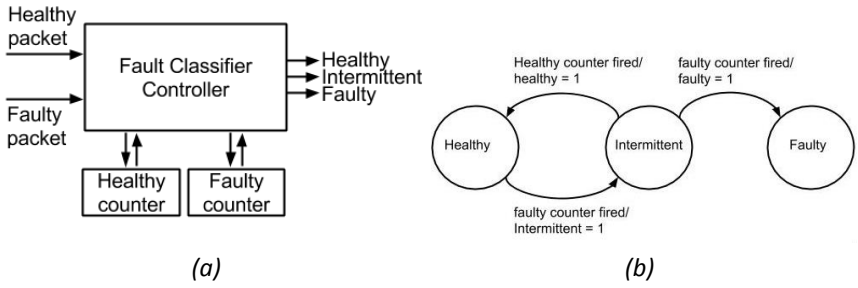


Figure 2.5 Proposed Classifier unit's a)Block Diagram and b)FSM Diagram

B fire three times every millisecond. If the thresholds frequency of the faults is two and three faults per millisecond, using the method described above, unit B would be classified as intermittent and unit A would be classified as healthy. However, different units in the circuit are used with different frequency. Lets assume in the example above, that unit A is used (or enabled) once every millisecond. This would in turn mean that unit A is not only not healthy, but should be classified as permanently faulty. This problem can be seen on physical links on the network. It is important to note that while the data on the link is not valid, the presence of the fault is not affecting the system and can be ignored. The only corner case is the fault affecting control signals between the routers, however, this topic is out of scope of this thesis.

In this work, we target a simple on-line, hardware fault classifier unit based on method introduced in [44]. The method introduced in [44] is deployed in each input port of the router and provides 3 states for the link (healthy, intermittent and faulty). It utilizes hamming code with one bit error correction and two bit error detection. The classifier unit monitors a window of n packets and counts the number of healthy and faulty packets by means of three counters (one for healthy packets and two for two different fault outputs of hamming decoder). Each of the counters has its own threshold value. Once any of these counter reaches this its threshold, all the counters reset and the state of the link is decided by a controller.

Fault classifier in [44] imposes large area overhead to the router and is only covering link faults. This work proposes a light weight version of this classifier that can be deployed on all input links and 20 turn faults [14]. The proposed classifier achieves the above mentioned advantages by: 1) replaced hamming encoding with a single parity (for the links) and 2) shared the counter for intermittent and permanent faults to reduce the area overhead (see Fig. 2.5 a.). The resulting classifier was deployed on all the input links and with small modification to the abstracted turn fault signals of the checkers. Since the checkers do not produce a signal while the circuit is not-faulty, the healthy signal

counter was replaced with a simple counter producing a constant window. However, this will not produce the same problem as described in the beginning of the section, since checkers only check combinatorial circuit and such circuit should provide valid output all the time. Another specific improvement in this design is that the FSM controller is designed in such a way that can move the sate from intermittent to healthy which becomes handy in cases that a series of transient faults are detected as intermittent fault (see 2.5 b.). Table 2.8 shows the area overhead of each classifier unit in the router. The small area overhead of such classifiers enables the designer to deploy them on critical signals reduce the latency of classification (in comparison to software-based solutions).

Table 2.8 Classifier unit overhead reports

Unit name	Unit area (μm^2)	Area overhead to baseline router
Classifier unit	1086	1%

Once a link is classified as permanently damaged, the upstream router would be notified. The LBDR unit will update its connectivity bits which reduces the adaptivity. All this process would be performed locally. In case of other units, the classified fault information would be transmitted to the global System Health Manager and further decisions would be made at software level.

2.5. Chapter summary

An essential part of any dependable system is a mechanism for fault information acquisition. This information would be used by fault mitigation mechanisms in different abstraction layers. Such solutions have four essential parts: fault detection, abstraction, classification and propagation. Software-based fault detection solutions can detect the faults once a fault in the hardware has turned into a failure detectable by the software. In contrast, hardware based low-latency fault information acquisition detects such faults in single clock cycle and enables the system to react and contain the fault propagation. This thesis focuses on the faults in the communication medium in a NoC based SoC. The proposed approach divides the NoC routers into data-path and control path and uses different existing fault detection mechanisms to address the faults in these sub-circuits. To detect the faults in the control-path of the router, the concept of checkers have been used. However, while generating such checkers, the area constraints of the final product is not considered (the focus is more on the fault coverage). To this end, this chapter introduced a minimization framework for reducing the area of the final set of checkers.

The proposed framework utilizes Greedy Heuristics and Depth-First Branch-and-Bound algorithm for reducing fault detection circuitry's overhead to the system. The concept of Dominant checkers was introduced in order to reduce

the search space of the minimization algorithms. Also a new metric for sorting the checkers, namely coverage density, was introduced which improves the results by Greedy Heuristics. The proposed method was able to reduce the area overhead of the checkers from 150%-200% for Arbiter and LBDR unit down to range of 120%-150%. Similarly for FIFO unit, the area overhead of the checkers was reduced reduced from 96.8% down to 93%. In all cases the minimization framework guarantees 100% fault coverage for all the designs. followed by discussion about fault information abstraction.

Once the fault information is gathered and abstracted before being used by the higher abstraction layers ($e.g.$ a software-based global system health manager), It should be classified into transient, intermittent and permanent. Moving the classification task to hardware provides the system with the following advantages: (1) the classified fault information is already available in the hardware layer and can be used by local fault mitigation mechanisms, (2) reduces the software load for such tasks. The free software time can be utilized for other dependability mechanisms such as dependable application deployment etc.. This work proposed an improvement to the existing online hardware fault classification. Each classifier unit adds only 1% overhead to the router which would enable deploying such units for combined fault signals such as turn-faults.

3. Local and Hybrid Fault Management

Fault management in large systems can be done centrally (globally), distributed or a hybrid of these methods. Central fault management has an advantage of having a general view of the system and can make more optimal decisions (for example a system wide reconfiguration while considering system wide metrics for performance, energy etc.) while handling faults. However, this approach usually suffers from higher response latency. This high reaction latency will lead to propagation of the fault to other parts of the system and might result in a system failure. In contrast to global fault management schemes, local fault management solutions , due to their distributed nature, are more scalable. Also their physical proximity to the hardware, enables them to react much faster to the faults, and prevent system failures. This work divides the local fault management mechanisms into two, local mechanisms and hybrid mechanisms. The local mechanisms are oblivious to the state of the system and are only aware of the situations in their close vicinity. In contrast, hybrid mechanisms are reconfigured by the global system manager but they act locally with low latency. The local mechanisms are prone to making decisions based on local information which might not result in an optimal solution for the entire system. This work uses a local and hybrid fault management schemes for local issues (such as handling faulty flits in the router) while using the central manager for system-wide reconfigurations (application mapping or hardware reconfiguration decisions).

A crucial component in this hybrid approach is a mechanism for propagation of fault information to the central manager. In this work its assumed that the propagating the fault information from the router to system manager are handled through two channels: 1) by using IJTAG scan-chain [45] and 2) through the actual NOC itself. As mentioned previously, the scan-chains would limit the mapping possibilities of the system manager on the nodes by requiring a dedicated access point. On the other hand, adding a small dedicated circuitry in the Network Interface for each node, provides the possibility of injection of diagnosis packets to be sent through the network to any designated node which is handling system manager tasks. In this case, the routers would pass the classified fault information to the NI. This classified fault information would be either: 1) directly passed to the connected PE in case the system

manager is mapped on this node, or 2) will be used to compose a diagnosis packet which will be injected back into the network towards the node with the system manager mapped on it. In order to accomplish this task the following functionalities are added to the Network Interface:

- Extension of the packetizer to enable generation of diagnosis packets based on received information from the node's router
- A memory mapped register (accessible by the nodes PE) to keep address of the system manager
- A read-only memory mapped register for passing the diagnosis information to the node's PE
- An extension to NI's flag register. This would enable the PE to check for incoming diagnosis information

The problem with sending diagnosis packets through the NoC itself would be the loss of connectivity to system manager, due to the faults in the network links and resources. However, the proposed approach would open up the possibility of starting the system with more possibilities for mapping system manager tasks. Once the NoC starts degrading and the connectivity to system manager through the network is lost, It is possible to move the system manager tasks to nodes with access points to the IJTAG network and continue the tasks of system manager.

This chapter focuses on local fault management mechanisms in NoC based SoCs. The problem of local fault management can be broken into the following sub-problems:

- Providing routing possibility once the system is experiencing permanent link failures
- Handling transient faults on the data path of the router
- Providing guarantee of reachability for injected packets
- Providing infrastructure for graceful degradation in the NoC routers

This chapter addresses each of these problems separately. This chapter is based on the following publications:

- "NoCDepend: A Flexible and Scalable Dependability Technique for 3D Networks-on-Chip," T. Hollstein, S. P. Azad, T. Kogge, H. Ying and K. Hofmann, DDECS 2015.

- ”Comprehensive performance and robustness analysis of 2D turn models for network-on-chips,” S. P. Azad, B. Niazmand, K. Janson, T. Kogge, J. Raik, G. Jervan, T. Hollstein, ISCAS 2017.

- ”From online fault detection to fault management in Network-on-Chips: A ground-up approach,” S. P. Azad, B. Niazmand, K. Janson, N. George, A.S.Oyeniran, T. Putkaradze, A. Kaur, J. Raik, G. Jervan, R. Ubar, T. Hollstein, DDECS 2017.

- ”Fault-resilient NoC router with transparent resource allocation,” T. Putkaradze, S. P. Azad, B. Niazmand, J. Raik and G. Jervan, ReCoSoC 2017.

3.1. Literature Review

One of the challenges in fault tolerant NoC design is to find an efficient fault tolerant routing algorithm. Many fault tolerant routing algorithms have been proposed in the literature with different characteristics [51, 52, 53, 54, 55, 56, 57, 58]. These approaches can be divided into the following groups:

- Inherently fault tolerant routing algorithms: Works such as [51] and [53, 54] for 2D and 3D Mesh based NoCs respectively. For instance, in [51] Bishnoi et al. have proposed a scalable mechanism for implementing fault-tolerant routing algorithms for topologies derived from the 2D Mesh due to one or two faulty link scenarios faults in the network. [53] defines circular routing paths to avoid horizontal and vertical faulty links. HARS [54], proposed a deadlock-free by using a mid-node searching method in 3D NoCs which only considers the faulty vertical links.

- Architecture-level support for fault tolerance: On the other hand, some approaches such as [59, 52, 56, 55], rely on using Virtual Channels (VCs) in order to avoid the formation of deadlock. Alternatively, some works avoid using VCs but restrict the number of tolerable faults (e.g. [57]) or assume the existence of vertical pillars (a working column in either corners of the chip) for 3D NoCs.

- System reconfigurability: Works such as [58] tackle this problem from network reconfiguration side.

However, a missing research point is to provide a general method for evaluating fault tolerance of routing algorithms which enables a fair comparison between the routing algorithms. Its equally important to keep the overhead of such approaches to the system minimal (which means avoiding the use of virtual channels). To this end, its necessary to go back to basics of the routing algorithms and provide a thorough investigation of 1) all simple routing

algorithms and 2) analysis of their behavior under different fault configuration (see [15]). A set of routing algorithms can be selected that outperform others in terms of robustness. This small set of routing algorithms can be used by the system manager in order to identify the optimal routing algorithm under any fault configuration in the network.

In order to avoid fault propagation in the system, it is crucial to either correct the faults using error-correction code or alternatively remove the faulty packets in the network. Adding error correction code for each flit in the packet will increase the data-path width and has a huge impact on the router's area (for example in case of hamming code, an extra 8 bits is required for 32 bit data). To avoid the mentioned area overhead, this work opts for the second solution and introduces a simple packet dropping mechanism for router's data-path protection (See [14]).

Another important point in local fault management is to avoid using faulty links in the network. Four main approaches to this problem are Local circumvention [60, 61, 62, 63, 64, 65, 66, 67, 68, 69], Global path-based methods [70, 71, 72], Intermediate node solutions [73, 74]and Data-splitting [75]. Also hybrids of these methods can be found in the literature. Local circumvention methods suffer from in-ability to see the reachability of the injected packets to the destination and other methods suffer from huge overheads in terms of time and hardware to the system. To this end, a new method was developed [16] which keeps a limited number of registers for each router output port which keep the information about unreachable areas of the network from that port. The proposed method can be combined with any turn-model based routing algorithm. Using the reachability information, it is possible to provide guarantee of delivery for each injected packet under arbitrary number of faulty links in the network. Also, since possibility of delivery of the packet is checked at every output port that the packet goes through, the proposed method would prevent routing the packets into paths that would not reach the destination.

As a final note, it is crucial to keep in mind that adding the fault tolerance mechanisms will in turn increase the router area. Larger chip area and increased number of components will result in more fault occurrences and will have its toll on the system's reliability. This means that fixing one problem might lead to larger issues once all the impacts of the method are considered. To this end, this work proposes a framework for evaluating the network's reliability [18].

3.2. Fault Tolerance Evaluation of Turn-Model based Routing Algorithms

Routing algorithms are used as an essential tool to avoid faulty links in the network. Using the routing adaptivity, it is possible to route the packets around the faulty links. The decision about routing algorithm is made globally by a global system manager, however, the act of routing is performed locally using limited local information hence the routing is considered a local fault tolerance mechanism in this work.

The focus of this work is on the Turn-Model based routing algorithms [22]. A turn is the change of direction in a packet's path. There are two mainstream naming conventions; Inter and intra router. Intra-router naming that uses the port names of entrance and exit of a single packet inside a router (e.g. a north to west turn describes a packet entering the router from north port and leaving from west port). Inter-router naming considers the turn from previous direction of the the next direction of the packet (e.g. a north to west turn describes a packet moving towards north-entering from south port of the router- and then turning to west-leaving the west port. Using intra-router naming this turn would be called a south to west). In this work we use the intra-router naming convention due to its simplicity. In a 2D Mesh network, directions are named based on the cardinal directions: North (N), East (E), West (W) and South (S). Since the U-turns (a turn starting from one direction and ending in the same direction) are illegal, maximum of 8 *turns* can be defined: N2E, N2W, E2N, E2S, W2N, W2S, S2E and S2W. For instance N2W indicates a turn that enables a packet coming from the North input port of the router be forwarded to the West output port. At the moment, only few of all the existing 2D turn models are reported in the literature. Table 3.1 describes these turn models with their assigned name in the literature.

Table 3.1 List of previously named turn models

#	Allowed turns	Conventional Name
0	E2N, E2S, W2N, W2S	XY [76]
13	S2W, S2E, N2W, N2E	YX [77]
33	E2S, S2W, S2E, N2W, N2E	Restricted North First [78]
39	E2N, E2S, W2N, W2S, S2W, N2W	East-First [79]
40	E2N, E2S, W2N, W2S, S2E, N2E	West-First [22]
41	E2N, E2S, W2N, W2S, N2W, N2E	North-Last [22]
42	E2N, E2S, W2N, S2E, N2W, N2E	Negative-First [22]
46	E2N, W2N, S2W, S2E, N2W, N2E	South-First [78]
48	E2S, W2S, S2W, S2E, N2W, N2E	North-First [78]

To this end, this work provides a method for enumerating all the Turn-Model [22] based routing algorithms and an analysis of their robustness in different fault scenarios.

3.2.1. Enumerating all Uniform 2D turn models

Since eight turns exist in the 2D turn model, total of $2^8 = 256$ turn models can be formed. However, for a turn model to be useful, it should provide full connectivity in the NoC-meaning any node should be able to be reached from any other node in the network- and deadlock freeness [19]. To evaluate these properties for 256 existing routing algorithms, the concept of the routing graph can be used.

Definition 3. *A Routing Graph, RG(V, E), is a directed graph, where the set V, denotes the set of all the input/output ports in the network (including local port of the router) and E represents the set of (v_i, v_j) where v_i is a vertex (input or output port) depending on v_j port. A vertex v in Routing Graph is denoted as $node_{i,p,dir}$, describing direction $dir \in \{in, out\}$ of port $p \in \{N, E, W, S, L\}$ of node i in the network.*

There are two different types of links represented as edges in the routing graph:

- *Inter-router edges*, describing physical link connections between routers (from an output port of a router to an input port of an neighboring router).
- *Intra-router links*, describing allowed connections inside the router (from an input port of a router to an output port in the same router). An intra-router link can be: 1) From or to local port: describing dependency between the router's north, east, west and south ports and the local port connected to the processing element (PE) of the same router. 2) Straight connections: describing dependency between ports in straight connections inside the router (e.g from North input to South output port of the same router). 3) Turns: dependency of the ports in perpendicular direction (e.g. from North input to West output port of the same router). These connections should follow turn model restrictions in the routing algorithm.

Since routing graphs show the dependency of resources in the network, they can be utilized for evaluation of deadlock freeness. A deadlock situation in the network under a specific routing algorithm, will result in a cycle in a routing graph. In this case, if there is no cyclic dependency of resources ($i.e.$ deadlock) in the NoC, a deadlock can not occur in the network. Using simple

Algorithm 2: Filtering out Turn-Models algorithm

```
deadlock_free_TM = [];
for turn model TM ∈ full_set_of_turn_modes do
    RG_TM = Generate_Routing_Graph(TM);
    deadlock_free = True;
    if RG_TM has cycle then
        deadlock_free = False;
        break;
    end
    for Source_Node ∈ NoC_Nodes do
        for Destination_Node ∈ NoC_Nodes do
            if Source_Node != Destination_Node then
                if !RG.has_path(Source_Node, Destination_Node) then
                    deadlock_free = False;
                    break;
                end
            end
        end
    end
    if deadlock_free then
        deadlock_free_TM.append(TM);
    end
end
```

graph algorithms, cycles in routing graphs can be easily detected. Also, to check connectivity of a routing algorithm, its simply enough to check if there exist a path from every source to every destination in the routing graph. It might be the case that certain routing algorithm implementations do not support such paths (due to their limitation). However, it is easy to implement such paths using source or table-based routing. Algorithm 2 describes the filtering process of algorithms which result in deadlock. After filtering out the turn models which result in deadlock or un-connected nodes, 50 deadlock free turn models with full connectivity remain (see Fig. 3.1 for visualization off all the 50 turn models).

3.2.2. Robustness evaluation of all 2D turn models

The Degree of Adaptiveness (DoA) introduced in [22] metric can be used to classify the turn models which only considers the minimal paths with length

Figure 3.1 visualization of all 2D deadlock free turn models which provide full connectivity in the network

equal to Manhattan distance from the source node to destination node. A general form of DoA metric can be formulated as:

$$DoA = \frac{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} NoSP_{i,j,rg}}{\text{number of pairs of nodes}} \tag{3.1}$$

Where N is the number of nodes in the network and $NoSP_{i,j}$ is defined as:

$$NoSP_{i,j,RG} = \begin{cases} \text{number of shortest paths in } RG \text{ from } node_{i,L,out} & i \neq j \\ \text{to } node_{j,L,in} & \\ 0 & \text{otherwise} \end{cases}$$

Table 3.2 DoA and DoA_{Ex} for all 2D routing algorithms (being shown in Fig. 3.1)

	4 turns			5 turns		6 turns	
Turn Model Num	0, 13	3, 5, 8, 10	1, 2, 4, 6, 7, 9, 11, 12	14, 15, 16, 17, 28, 33, 36, 37	18-27, 29-32, 34, 35	42, 43, 45, 47	38, 39, 40, 41, 44, 46, 48, 49
DoA	1			1.23		1.43	
DoA_{Ex}	1	1.41	1.63	2.11	2.41	3.83	4.33

The resulting DoA for the turn models are presented in Table 3.2. It comes naturally that turn models with higher number of turns, would provide higher DoA. However, as mentioned before DoA only covers the minimal paths and hence, can be used for minimal path routing turn model classification. Extending this metric (DoA_{Ex}) to cover non-minimal paths in the network ($i.e.$ include all the simple paths in the network-paths that do not have repeating nodes in them) provides a slightly different picture than the original DoA. DoA_{Ex} makes it possible to classify the turn models even further. The DoA_{Ex} metric can be described as follows:

$$DoA_{Ex} = \frac{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} NoSP'_{i,j,RG}}{\text{number of pairs of nodes}} \tag{3.2}$$

Where N is the number of nodes in the network and $NoSP'_{i,j}$ is defined as:

$$NoSP'_{i,j,RG} = \begin{cases} \text{number of simple paths in } RG \text{ from } node_{i,L,out} & i \neq j \\ \text{to } node_{j,L,in} & \\ 0 & \text{otherwise} \end{cases}$$

Table 3.2 presents DoA and DoA_{Ex} metrics for the turn models in Fig. 3.1. This table shows that inside each class of turn models (four, five, and six-turn turn models), there are sub-classes that have different characteristics. As an example, turn model no. 3 shares two turns with XY and two turns with YX which allows it to have non-minimal de-routes. Similarly, under non-minimal routing, turn model no. 1 and 2 have even further advantage in providing path diversity.

To evaluate the robustness of the 50 deadlock free turn models with full connectivity routing algorithms, Alg. 3 is used. The proposed method calculates the average connectivity metric of each of the turn models over all possible combinations of n_{broken} broken links in the network, for n_{broken} starts from zero and goes up to the number of links in the network.

Algorithm 3: average connectivity calculation algorithm

```
for n_broken ∈ [0, n_total] do
    list_of_configurations = list of all 3×3 2D mesh NoCs with n_broken broken links
    sum_con = 0
    forall conf_broken ∈ list_of_configurations do
        Generate RG based on conf_broken
        sum_con += Connectivity_rg
    end
    avg_con[n_broken] = sum_con/len(list_of_configurations)
end
return avg_con
```

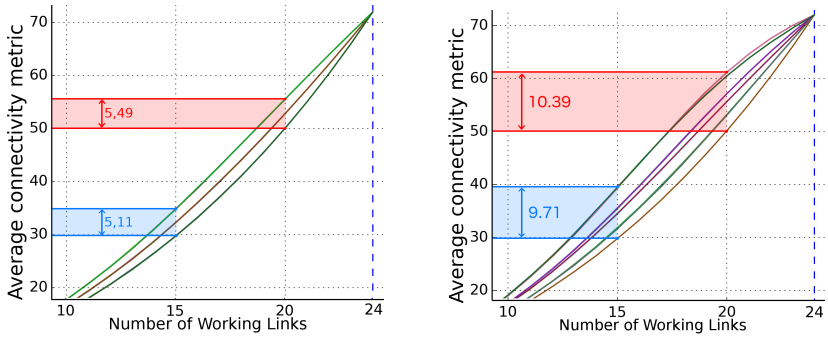


Figure 3.2 Comparison of avg. connectivity metric of turn models under a) minimal, b) non-minimal routing by number of available links.

Fig. 3.2 shows the result of Alg. 3. The average connectivity of routing algorithms in Fig. 3.2 a. for minimal path routing falls on three lines which correspond to three classes of DoA. Similarly the graphs for non-minimal routing fall on six lines, corresponding to six classes of DoA_{ex}. This means that, in fact, uniform routing algorithms with higher degree of adaptivity can provide better connectivity on average with same number of broken links. This however, will not provide any insight about the fact that the implemented routing mechanisms which operate under the given turn-model can find this path. This problem arises from the fact that the routing logic only has information regarding the current router or from its immediate neighbors. This lack of information might lead to sending the packet in a direction (using a turn allowed by the routing algorithm) which puts the packet into a path that

can not reach the destination. In approaches such as table-based routing mechanisms, the information can be adjusted so there can be guarantee of reachability, considering the fact that routing tables are not scalable. Many other approaches specifically algorithmic routing implementations suffer from lack of such guarantees. In section 3.3.1 a novel method has been proposed to tackle this problem. However, assuming the existence of such routing mechanisms, it is possible to rely on the connectivity metric. It is important to note that for an equal number of broken links, the connectivity metric gap between the graphs grows from minimal to non-minimal path routings. This is due to the fact that non-minimal path routings provide more path diversity and hence, provide more possibility for the net work to stay connected. An analytical approach for reliability assessment of routing algorithms has been described in [80]. However, this work focuses on turn failures in routers and only considers negative-first turn model for evaluation.

Its equally important to compare these algorithms performance (Latency and Throughput) and energy. Performance and energy results of all the turn model based routing algorithms are published online [81]. [81] also contains the full list of all 3D deadlock-free turn models is provided.

The same method can be applied to compare other routing algorithms. By using the proposed method, it is possible to single out turn models which provide better robustness under any fault configuration.

3.3. Local Fault Handling

One of the most important tasks in local fault management is online handling of the faults. Even one transient fault in control part of the router or packet (destination address or flit type), might lead to a total network failure. This section proposes mechanisms to prevent such failures. In order to mitigate such failures, in the router, critical locations in the router should be identified. These critical router locations are FIFO input where the flow-control signals are handled and Routing-Logic input where routing decisions are made. This work uses packet dropping as a mechanisms for handling faults in such critical locations. Similar approach has been proposed in [48], once an error is detected by the parity checker located at switch's input "the transmission can be interrupted immediately without storing the flit and routing it to an output". However, implementation details of such process is not outlined. Removing the faulty flit in the router is not a trivial task. Specially if the faulty flit is the header or tail flit of a packet. [82] uses packet dropping for such purpose as well. And they mention the case of corruption of header. However, this work does not address the problem of tail-less packets. Since routing logic usually terminates the connection using tail flit, once a tail flit is removed, the

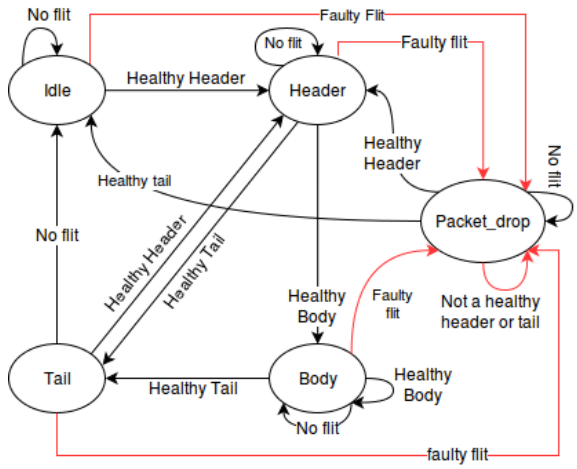


Figure 3.3 FIFO packet dropping Finite State Machine

Table 3.3 Area overhead of proposed packet dropping mechanisms

unit name	original unit area (μm^2)	updated unit area(μm^2)	area overhead(%)
FIFO	14357	16045	11.7%
LBDR	1744	2940	68.5%

successive packet might follow the faulty packet and leave an undesirable port of the router, leading to deadlock and network congestion.

Fig. 3.3 shows the Finite State Machine (FSM) being in charge of controlling the packet dropping. The FSM starts in Idle state and upon receiving a packet, goes through states named after the flit type. In case the current flit is faulty (detected using parity checker for data path), the FSM moves to the packet dropping state and either completely removes the packet (in case the fault is in the header flit) or removes the rest of the packet (in case the fault is in the body or tail flit) and patches it up with a dummy tail. In the later case the application layer would notice the miss-match of the value of packet length in the header flit and the actual length of the packet and will handle the situation.

For the routing logic, since the decision is made using the header flit, it is relatively easy to perform a packet drop process. Upon detection of a fault in the output of the FIFO, or in the case that the routing logic can not produce a valid request (due to broken network links), the routing logic would initiate a packet drop phase. The packet dropping in the routing logic would consist of sending fake grant signals to the FIFO without producing any requests to the arbiter until the tail of the packet is dropped.

The overhead of the proposed methods is reported in Table 3.3. These results should be analyzed considering the fact that the LBDR unit is small in

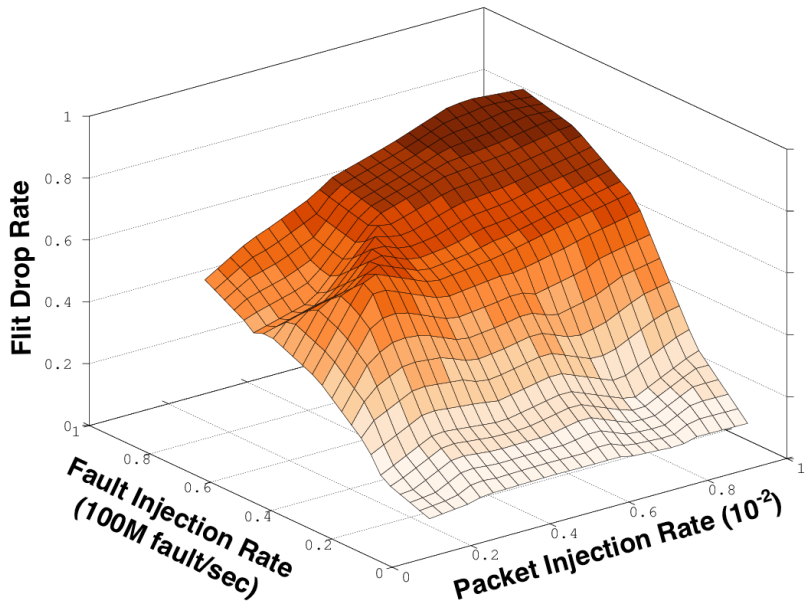


Figure 3.4 Packet Drop rate in different packet and fault injection rates

size and the FIFO unit is only four flits deep and does not have any virtual channels.

Fig. 3.4 shows the experimental results of packet drop rate in different packet injection under different fault injection rates. In this experiment, all the faults are injected on the network link. Its important to note that while network's packet drop rate can reach to 80%, the network does not suffer from congestion and similar failures mentioned before.

3.3.1. Reachability and partitioning in on-chip networks

Providing guarantees of reachability of a packet to its destination is a critical topic in fault tolerant NoCs. If a packet can not each its destination due to a fault configuration in the network, It either can cause a network-wide congestion or in presence of packet-dropping mechanisms, the data will be lost which is not acceptable in hard-real time systems.

This work proposes a mechanism (called NoCDepend) to keep minimal information about non-reachable areas of the network in each output port in order to provide guarantee packet reachability. This guarantee is provided under arbitrary number of faults as long as a path exist from the source node to the destination. The proposed mechanism is independent of the routing algorithm and can be adapted to any turn-model based routing algorithm.

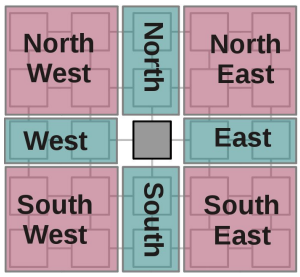


Figure 3.5 8 regions for each node used for back-propagation algorithm

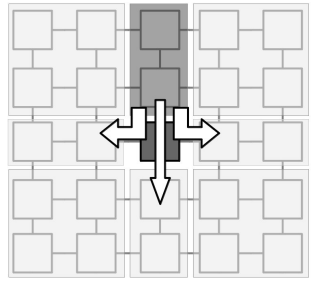


Figure 3.6 Information propagation example for non-rectangle in north

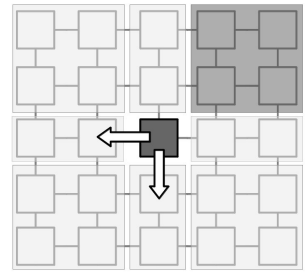


Figure 3.7 Information propagation example for non-rectangle in north-east

The information about unreachable areas is coded as coordinates of the lower left and upper right (or can be alternatively coded as lower right and upper left) corners of a rectangle representing the area. The rectangle information for each router output is calculated by the Global System Health Monitoring Unit (SHMU) and is propagated to the routers via an IJTAG infrastructure.

To detect if a packet can reach its destination from a current routing path position, it is enough to compare the destination address with the coordinates of the rectangles stored in the output port. In case a destination is not reachable, the packet should not be injected in the network or can be easily dropped as early as possible, notifying the SHMU the extent of the failure. Later on, SHMU should request a re-mapping and re-scheduling of the application to avoid this scenario.

Calculation of the non-reachable areas can be done using a simple back propagation algorithm (For more information please refer to Appendix B). The principle of back-propagation works based on dividing the space for each node to 8 regions (see Fig. 3.5). The information about non-reachable region would be propagated to the proper outputs based on the requirements of the routing algorithm. This means that if there is an allowed turn in the turn-model the reachability information should be transfered in the opposite direction to the routers upstream which use those turns. Figures3.6 and 3.7 show examples of propagation of information. However, performing this algorithm in hardware imposes a large hardware overhead (specially for storing the rectangle information).

Alternatively non-reachable areas can be calculated off-line in the System Health Monitoring Unit (SHMU) using the concept of Routing Graph (RG) introduced in section 3.2.1. Using the RG, it is possible to calculate the unreachable node coordinates for each output port for each router. These unreachable area rectangles would be compressed and merged in order to reduce their number to the available memory slots in each output port. Later,

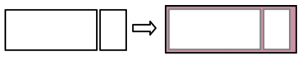


Figure 3.8 Rectangle merging scenario

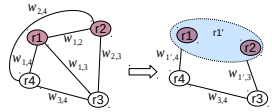


Figure 3.9 Lossy merging heuristics

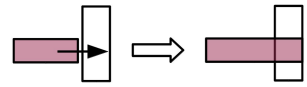


Figure 3.10 Rectangle Expansion Process

these information would be sent back to the routers which will have a few registers for keeping the final minimized information. Different merging options are discussed in the subsections bellow:

Loss-Less merging

There are two basic cases for loss-less merging: 1. There are two rectangles that have same height or same width and which are adjacent on that side, or 2. one rectangle is contained in another one. In first case, we can merge both rectangles into one larger rectangle (see Fig. 3.8) and in the second, we can delete the rectangle, being a subset of the other one.

Lossy merging

Lossy merging only occurs when the number of rectangles exceeds the available number of list entries of a router output port. In this case, some reachable nodes have to be sacrificed in order to meet the list size constraints. Given a graph $G(V,E)$ where the set V of vertices represents the non-reachable rectangle list of one router output and the weight w_{ij} of an edge $e_{ij} \in E$ between two vertices v_i and v_j represents the number of reachable nodes that would be lost if these two vertices merge.

Greedy Merging: Adjacent vertices of edges e_{ij} of the Graph G having smallest weight w_{ij} are merged as long as $|V| > MaxNumAreas$.

Full Enumeration: if m rectangles have to be joined into k list entries, for small m and k a full enumeration of solutions (m^k) could be evaluated an the one with the least loss is selected.

Heuristic Search: The m vertices are randomly distributed on k buckets and then an optimization is performed by the Fiduccia-Mattheyses [83] group-migration heuristic and the least costly found solution is finally selected.We would like to cluster Graph G with m vertices into n clusters where n is equivalent to the number $MaxNumAreas$ of list entries. This process can be interpreted as a classic graph clustering problem and can be solved with local search. The process is shown in Fig. 3.9. In this case, the cost function for any particular solution would be the number of reachable nodes that we will loose

if we choose that solution. Pseudo code of this algorithm is shown in Algorithm 4.

Algorithm 4: Heuristic Search for Lossy Optimization

```
1  solution = randomly assign vertices to clusters
2  Cost= cost(solution)
3  Best Cost= cost(solution)
4  Best Solution = solution
5  while i < numberofiterations do
6      choose random Vertex
7      choose random Cluster
8      solution = move chosen vertex to chosen Cluster
9      Cost= cost(solution)
10     if Cost ≤ BestCost then
11         Best Cost = cost(solution)
12         Best Solution =solution
13     end
14 end
```

Rectangle Expansion

Rectangles which are adjacent and one can be expanded inside the other one. This means that some unreachable areas will be included in two different rectangles. This will increase the hit rate when searching in the list in case of a a fully serialized implementation of the detector circuit. Examples of this process are shown in Fig. 3.10.

The experiments show that a large number of faults with different configurations can be handled using the proposed method by applying only small hardware overhead (4%) to the router. The only drawback of this method would be the lossy merges; when a reachable node falls into a non-reachable area due to limited register size in the router. This method can be combined with any turn-model based routing method and is independent of the underlying routing mechanism.

The same method can be used in order to partition the chip into criticality domains in mixed criticality SoCs. Details of partitioning method are described in [84]. The advantage of this process is that there is no need for extra hardware overhead and multiple criticality domains with full or partial isolation can be formed by organizing the unreachable rectangles. The proposed method can provide arbitrary number of partitions in the circuit and at the same time provide all the faulty link management capabilities in the original system.

3.4. Reliability Evaluation and Improvement of Fault Tolerance Mechanisms

The last but not least part in the local fault management, is local resource management. After classifying a unit as faulty, it should be isolated. Also, lifetime of the router should be prolonged as long as possible (with degraded performance) using resource management. However, it is important to note that the insertion of fault tolerance mechanisms has an impact on system reliability. To this end, the several architectures have been proposed in [18] with the aim of keeping one path in the router functional.

To evaluate each architecture, this work proposes the use of block-diagram reliability analysis [9]. In order to evaluate reliability of a router, we consider a router functional when at least a path in the router is functional. A path is defined as one input and one output channel. In this work, an input channel consists of an input FIFO and the routing unit and the output channel consists of the output multiplexer and the arbiter unit in charge of the output channel. To put it more formally, the reliability of a path is calculated as:

$$R_{path} = R_{Channel_{in}} \times R_{Channel_{out}}$$

Further on, for the baseline handshaking router we can consider that an input channel consists of a FIFO and an LBDR unit and an output channel consists of an $arbiter$ and a crossbar switch:

$$R_{Channel_{in}} = R_{FIFO} \times R_{LBDR}$$

$$R_{Channel_{out}} = R_{arbiter} \times R_{xbar}$$

It is possible to normalize reliability of different modules towards FIFO (the largest unit in the router), using their relative area. Lehtonen et. al. [85] have used similar approach to investigate NoC-level architecture reliability. In this work we consider the following for unit x:

$$R_x = \frac{Area_{FIFO}}{Area_x} \times R_{FIFO}$$

We can include all the additional reconfiguration hardware required for a path to function into the general router reliability formula and compare different architectures impact on the router reliability.

In [18] we proposed several architectures which reuse the NoC router resources (by adding extra multiplexers) in order to improve the router's reliability (for more information please refer to Appendix G). The proposed architectures are compared in terms of their reliability using the proposed method. These architectures have been examined against double modular redundancy (considering an existing fault detection mechanism for each unit)

and triple modular redundancy techniques. For one architecture, the results show that with 77% area overhead to the original design, the reliability of the router can be improved dramatically (compared to methods such as DMR with fault detection and TMR). Also the proposed method will slightly increase the critical path delay due to addition of all the multiplexers in the router.

3.5. Chapter summary

As the systems move towards feature miniaturization, it becomes more and more important to provide scalable local fault management mechanisms that can protect the system from failure and provide graceful degradation in case of permanent damages. This chapter targets local and hybrid fault management approaches in NoC routers which can react to the faults with much lower latency compared to global schemes. Local fault management schemes are completely oblivious from the global state of the system. In contrast to local fault management's limited view, Hybrid schemes are reconfigured globally while having all the advantages of closeness to the hardware. The inherent fault tolerance of the routing algorithm can be used as the first step of managing the faults in the network. One of the most popular routing techniques is the use of turn model based routing algorithms. However, even uniform turn-model based routing algorithms are not fully identified to begin with. Later a framework for evaluating their fault tolerance is required as well. To this end, this chapter enumerated all 2D uniform turn model based routing algorithms. An extension of adaptivity metric was introduce that can classify behavior of turn model based routing algorithms using non-minimal path routing. Last but not least, in this chapter the selected turn models were evaluated in terms of robustness in presence of faulty links in the network. The result of this evaluation would be used to select the best turn-model that outperforms other turn models in tolerating link failures, at the beginning of the system lifetime. Also the global system manager would only evaluate the effectiveness of a handful of turn models (which perform better than others) for each critical link failure.

Another pressing issue in NoC routers is the damage that even a single transient fault can cause in a NoC; it can either cause a packet miss-route or lead to a network-wide congestion. Such packets should be removed from the network to prevent possible network shutdown. This chapter provided two lightweight solutions for handling fault locally using packet dropping in FIFO and routing logic. The proposed mechanisms adds 11% to the FIFO size and 68% to the Logic Based Distributed Routing (LBDR) logic respectively. The experiments show that using this method, the network will preserve its integrity even if up to 80% of the packets are faulty and dropped.

Another equally important issue is the guarantee of reachability. It might not be feasible to route the packets in the network under any fault configuration.

This means that such packets would be stuck in the network which can lead to network-wide congestion. A light-weight mechanisms is required to restrict injection of such packets into the network. Furthermore, since the routers are making their routing decisions locally, they might route the packet in a direction allowed by the routing algorithm which leads the packet to a direction in which the destination is not reachable due to the faults in the network. A mechanism is required to check the possibility of reachability of each node from each router's output port. This chapter introduced a mechanism for guarantee of service in the Network-on-Chips with small hardware overhead (%4) to the router.

Also it is possible to use the regularity of a NoC router in order to re-use isolated units to provide performance degradation instead of a total router shut-down. In this chapter, an evaluation framework for investigating the effects of fault tolerance mechanism on system's reliability was introduced along with solutions which provides graceful degradation and enhancing system's reliability.

4. Global Fault Management

Considering the growing number of on-chip components, it is not possible to manage all the faults locally. The local fault management mechanisms do not have a global view of the system and can not take into account the effect of the local decisions on the system behavior. This might lead to situations that a series of local reconfigurations based on the limited available information, will lead to the non-optimal solution from global point of the view. Alternatively many of such approaches rely on configuration from a higher abstraction layers. Therefore it is crucial to have a global fault manager which collects the fault information from the lower abstraction layers, maintains a holistic view of the system and performs system-wide reconfiguration either by hardware reconfiguration or by performing partial or full application mapping/scheduling. Also by focusing on the local fault manager, the effects of such faults on the system will be ignored. This means that while a local unit is mitigating a fault or dropping a packet in the network etc., The effects of such actions on the application behavior is not considered. Moreover the collaboration of local and global fault management systems should be investigated in order to identify the shortcomings of each of these mechanisms in more detail and to optimize the distribution of such tasks over different layers of abstraction. To examine the effects of such global manager, it is necessary to perform high-level simulation of its behavior. In the next section, the available simulation tools for NoC based systems are analyzed. The results of such investigation shows that while there are many high quality solutions for simulation of execution of application on NoC based SoC, but a general simulator which support the faulty tolerance mechanisms and Mixed-Criticality applications is missing in the literature. To this end, this chapter provides the following contribution:

- A new simulator for NoC based SoCs for modeling global fault management under mixed-criticality constraints

This chapter is based on the following publication:

- 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.

And borrows from the following supporting publication:

- S. P. Azad, B. Niazmand, J. Raik, G. Jervan, T. Hollstein: ”Holistic Approach for Fault-Tolerant Network-on-Chip based Many-Core Systems”,2nd International Workshop on Dynamic Resource Allocation and Management in Embedded, High Performance and Cloud Computing DREAMCloud 2016 (arXiv:cs/1601.04675), DREAMCloud/2016/05

4.1. Literature Review

Many different approaches exist in the literature for providing global fault management for NoC based systems. [86, 87, 7] recalculate different application mappings for the NoC under different constraints (mainly targeting performance and energy). In [86], the authors have proposed a system-level Fault-Aware Resource Management (FARM) approach using spare processing elements to replace faulty ones. [88] proposed a simulated annealing based mapping algorithm for minimizing energy consumption, while considering link failures in the network. [89] uses a Mobile Master which keeps the view of the system health and an Application manager which is in charge of application mapping. [90] addresses a fault prediction method for the network links and acting upon the information at the OS level.

Aside with proposed mapping and scheduling algorithms and methods in the literature, different frameworks for mapping and scheduling of applications on NoC based systems are proposed in the literature, covering different mapping algorithms from Meta-heuristics such as GA and SA [92, 96] and heuristics [91, 93, 97] to different static and dynamic mappings [94, 95] and random mappings[92, 97] using different cost functions such as performance [92, 93, 94, 95, 96, 97], average communication [91] and task execution time [98]. These frameworks, utilize adaptive (minimal and non-minimal) [91, 94, 96, 97] and deterministic routing algorithms [92, 93, 94, 95] and are written in different programming languages. However, to the best of the authors knowledge, none of the above mentioned frameworks have support for mixed critical applications and only one has limited support for fault tolerance.

To this end, this work introduces an open-source framework for mapping and scheduling of applications on a NoC based system which explores the design space in different levels. Support of mixed-critical applications and fault tolerance mechanisms, are the main motivations of this work. Table 4.1 summarizes the features and characteristics of the frameworks mentioned above along with the proposed framework (”SoCDep^2 ”). As it can be seen in Table 4.1, ”SoCDep^2” includes most of the features provided by the other

Tool Name	Cost Function	Mapping Algorithm	Routing Algorithm	GUI	Open source	Programming Language	Scheduling Support	Mixed-Criticality Support	Fault Tolerance Support
SUNMAP [91]	average communication delay, area, power	Minimum-path mapping (based on NMAP)	Dimension ordered,minimum path,traffic splitting across all paths	+	–	C++/SystemC	Not mentioned	–	–
SMAP [92]	Performance	Spiral-based, GA, random	XY,rest not mentioned	+	–	MATLAB	+	–	–
xENoC [93]	Performance	Based on SUNMAP	XY,Diagonal XY	–	–	SystemC	Not mentioned	–	–
Atlas [94]	Performance, Power	Static and dynamic	XY,turn model (West-First) routing	+	+	Java/SystemC	+	–	Limited to HERMES
HeMPS [95]	Performance	Static and dynamic	XY	+	+	C++/SystemC	+	–	–
GSNoC [96]	Performance, Energy Consumption	GA, SA	Minimal and non-minimal routing,adaptive routing,turn model routing	+	–	C++/SystemC	+	–	–
NoCTweak [97]	Performance, communication energy	Random, NMAP	Minimal routing,deterministic and adaptive routing,turn model routing	–	+	C++/SystemC	+	–	–
Project Cheddar [98]	Task execution time	Not mentioned	Not mentioned	+	+	Ada	+	–	–
SoCDep^2 [proposed method]	Scheduling Makespan	LS, ILS, SA, Min-Min, Max-Min, MCT, MET, NMAP	Minimal and non-minimal turn model based,Adaptive and Deterministic	+	+	Python	+	+	+

Table 4.1 Comparison of the available tools in terms of different criteria

tools and the noticeable advantages are that (1) it addresses support for NoC based many-core systems fault tolerance investigation, such as injection of faults, fault monitoring, testing and classifying faults under mixed-criticality constraints, (2) it provides support for a wide variety of routing algorithms, mapping algorithms and scheduling and (3) it is open-source, making it possible for different research groups to reproduce the experiments and contribute as well.

To tackle dependable task deployment on NoCs, the following sub tasks should be performed: 1) Application modeling, 2) architecture modeling, 3) System Health Monitoring Unit modeling, 4) mapping and scheduling and finally 5)to examine the system performance under different faulty environments.

Fig. 4.1 shows the considered simulator architecture. It includes the NoC based SoC target architecture which has a Processing Element (PE) in each tile. The system also contains a System Health Monitoring Unit (SHMU) and a Mapper Scheduler Unit (MSU). The SHMU is in charge of collecting fault information and maintaining a System Health Map (SHM) and MSU uses this information to deploy the application -represented in form of a Task Graph (TG) or a Clustered Task Graph (CTG)- on the system. The MSU models the target architecture as an Architecture Graph (AG) and represents the routing in the system by Routing Graphs (RG). The current mapping is stored in a shared memory called Current Mapping Memory(CMM) which will be used by SHMU to determine the impact of the faults in the system. The system determines a list of most probable mappings in a memory called Most Probable Mapping (MPM) Memory.The following sections provide detailed description of different parts of the framework.

4.2. Application Modeling

In this work, the application model is described as a Task Graph (TG). A task graph is a directed acyclic graph TG(V,E) where V is the set of tasks and E is a set of edges between tasks which describes tasks data dependency. The weight of each edge represents the amount of data transfer between tasks. The proposed framework provides the user with the possibility to specify TG as manual, random dependent or independent or a benchmark application.

Each task (t_i) in TG has the following attributes:

- ID number of the task in TG
- Task's Worst Case Execution Time (WCET)
- Task's deadline
- Criticality Level: tasks in mixed-critical systems are divided into two major classes, High critical tasks (H) which the system can not tolerate missing their deadlines and Low critical (L) tasks which the system will continue

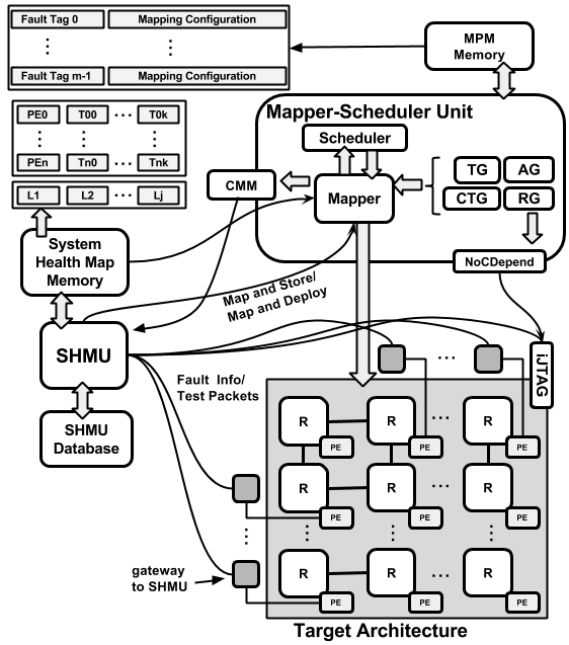


Figure 4.1 NoC based SoC System Model Architecture

to work with degraded performance in case they miss their deadline. However, two more classes (gateways) are added to the model which are used for network partitioning [84]. These gateways are not present in the original task graph and are inserted in order to handle the communication between High and Low critical tasks. All the communication between the tasks, i.e. the edges, are assumed to fall either under High critical or Low critical domain. Tasks are divided into the following classes:

  - High critical tasks (H): both incoming and outgoing edges are critical.
  - Gateway to Critical tasks (GH): the incoming edges are not critical communication, but their outgoing edges are considered critical.
  - Gateway to Non-Critical tasks (GNH): the incoming edges are critical communication, but their outgoing edges are considered non-critical.
  - Low critical tasks (L): the incoming and outgoing edges are both considered non-critical.

- ID of the cluster that the task is assigned to (in case of having clustering)
- ID of the node in Architecture Graph (AG, which represents the NoC topology see section 4.3) that the task is mapped onto

- Release time: Time which the task becomes available. Release time is not necessary for all the applications.
- Task type: Application or Test. Test applications are described in section 4.5.5.

Each edge e_{ij} represents communication between tasks t_i and t_j. Each edge e_{ij} in the TG, has the following attributes:

- Criticality level of the communication between two tasks.
- ID of the links from the Architecture Graph (AG, which represents the NoC topology see section 4.3) that the edge is mapped onto.
- The communication weight between two tasks, measured in terms of the number of flits.

In addition, This framework supports possibility of different slack(s) insertion for high-critical tasks' re-execution.

4.3. Architecture Modeling

In this work, the target architecture is modeled using an Architecture Graph (AG). An Architecture Graph is a directed graph AG(V, E) where V is the set of nodes including a router and a Processing Element (PE), and E represents the set of physical links between different routers. Fig 4.2 depicts an example of an architecture graph for a 3×3 NoC with Mesh topology. For each node the following attributes are associated:

- task list: list of mapped tasks onto the node
- scheduling: allocated time-slots for each task mapped on the node.
- list of unreachable areas for the node: these are the areas of the network (in the form of rectangles) that can not be reached from this node (For more information please refer to [16]).
- Node partition: Each node can be either in critical or non-critical partition or can be a gateway node between the partitions. This will determine which type of tasks can be mapped on the node(see [84] for detailed information).

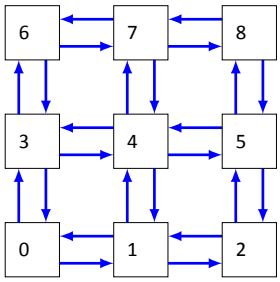


Figure 4.2 Architecture graph example for 3×3 mesh network

Similarly, the links keep the information about mapped packets (edges in the TG) and their schedule. The "SoCDep^2 " framework supports both 2D and 3D topologies, including 2D Mesh, 2D Torus, Line, Ring and 3D Mesh.

4.3.1. System Health Monitoring Unit Model

System Health Monitoring Unit (SHMU) is in charge of collecting fault information from the NoC (as mentioned in Chapter 1) and maintains a general view of the system health status(faulty/non-faulty or performance degradation). The considered fault model includes permanent and transient faults on links, router turns and Processing Elements (PEs). These information are stored in a shared memory called System Health Memory (SHM).

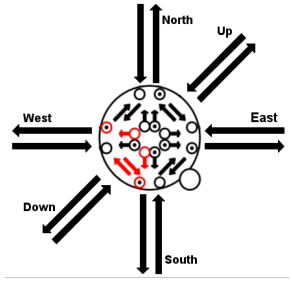


Figure 4.3 Visualization example of a 3D router health

The placement of SHMU in the system is a critical point as well. It can be either mapped on one or more nodes of the network, or can be separate unit embedded in the system. Keeping SHMU as a dedicated hardware has disadvantage of turning this unit into single point of failure. On the other hand, mapping SHMU tasks on system's processing elements requires guarantees of connectivity to other parts of system for collecting fault information performing system reconfiguration. However, placement of the SHMU is not the focus of this work. In this work the SHMU is considered as a stand-alone unit(as shown in Fig. 4.1) and communicates with the mapper-scheduler unit (MSU) via a dual port shared memory(System Health Map). This work also assumes that there exist a fault propagation mechanism that transfers the fault information to SHMU from local fault detectors.

An example visualization of a node in SHM for a 3D NoC is shown in Fig. 4.3. The big circle shows the router and all the turns in the router and the small circle on bottom right corner shows the Processing Element (PE). Broken components are visualized as red. The outgoing and incoming arrows towards the router are representing the links between the current router and its adjacent routers and single arrows inside the router show turns in one plane. The $\odot$ symbol describes the outgoing arrow from the plane, and $\bigcirc$ symbol describes the incoming arrow towards the plane. Combining these with four directions will result in eight 3D turns. As an example, $\odot \rightarrow$, describes Up-to-East turn. Other eight 3D turns are shown using the circles close to outer border of the router. A $\bigcirc$ near the North port represents a North-Down turn.

4.4. Routing Algorithm Modeling

In order to route packets to their destinations, "SoCDep^2" utilizes concept of Routing Graph (RG) introduced in section 3.2. A Routing Graph RG(V,E) is a directed acyclic graph where V is the set of all the ports in the network (input and output ports of the routers, including the local ports) and E is the set of

all connections between the input and output ports [5]. Using routing graphs enables the tool to use graph algorithms for finding paths between any source destination.

The connections in the set E in RG can be either 1) any internal connection inside a router between the input and output ports (including turns and other paths inside the router) and 2) the physical link connection between neighboring routers. By setting the internal connections, it is possible to model any turn model based routing algorithm under a specific fault configuration. It is also possible to feed in user defined turn models and set up non-uniform routing algorithms such as odd-even routing [99].

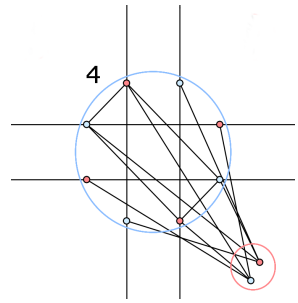


Figure 4.4 Example of routing graph [5] of a single 2D router under XY routing algorithm

Fig. 4.4 illustrates an example of a router in a 2D Mesh network under XY routing. The blue nodes represent input ports and red nodes represent output ports of the router. The nodes enclosed in red circle describe the local ports and the nodes enclosed in the blue circle are the router ports that are connected to adjacent routers. The edges inside the blue circle describe the turns and straight paths inside the router, meanwhile the outgoing and incoming edges to the blue circle describe the connectivity of this router to adjacent routers.

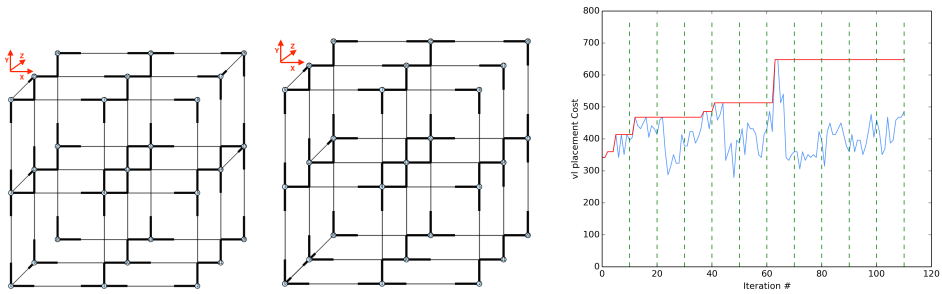
The concept of routing graph is used in reachability analysis for NoCDepend mechanism which uses path-finding in RG in order to determine non-reachable regions of network from each router port. Similarly, RG is used in adaptivity analysis and robustness of routing algorithms under different fault configuration. Once the system manager identifies a faulty link, it can find the most effective routing algorithm in terms of connectivity and reconfigure the routers to the new routing scheme.

4.5. Optimization Algorithms

This section the available optimization algorithms for vertical link placement, task clustering and mapping and scheduling in $SoCDep^2$ are discussed.

4.5.1. Vertical Link Placement

In 3D NoC topologies, one of the factors that can affect the performance of the system is the number and placement of vertical links. Due to the large area overhead of the vertical links, its not possible to have a full 3D mesh network and the number of these links are limited. This problem highlights the need for placement optimization of such links. One of the criteria that can



(a) *(b)* *(c)*

Figure 4.5 VL placement: a) Initial and b) Final VL placement c) Visualization of the connectivity metric development during iterative local search

be considered for targeting optimization of vertical links is the connectivity of each source-destination pair in the network. This can be achieved by initial selection of a routing algorithm for the 3D NoC and moving the vertical links in order to find the optimal or high quality placement solution. "SoCDep^2" uses the RG to evaluate the connectivity for each node and maximizes this metric by running an LS or ILS heuristic. The optimization algorithm considers the sum of the number of connected source-destination pairs as cost function and tries to maximize it using LS heuristics.

In this subsection, an example of Vertical Link (VL) placement is described. In this scenario, a $3 \times 3 \times 3$, 3D-Mesh NoC with Negative-First turn model routing algorithm is used. We consider the number of pairs of nodes that can communicate under the constraints of the routing algorithm, with current configuration of the VLs as our metric for a greedy ILS heuristic. Fig. 4.5, shows the initial (with connectivity metric: 342) and final VL placement (with connectivity metric: 648) and increase in connectivity metric during the iterative local search optimization. Each green vertical line in Fig. 4.5 c. shows the beginning of a new local search cycle.

4.5.2. Task Clustering

By growing number of tasks in the TG, the search space of the mapping algorithm also grows. If the task graph has m tasks and the network has n nodes then the size of the search space for placing tasks on the network nodes will be will be: n^m. To reduce the problem size, one approach is to cluster the tasks together and mapping task clusters to the network nodes. To obtain better mapping result, it is crucial to optimize task clustering. This has been performed using greedy heuristics. In case of mixed-critical applications with partitioned network, it is necessary to cluster tasks of the same criticality

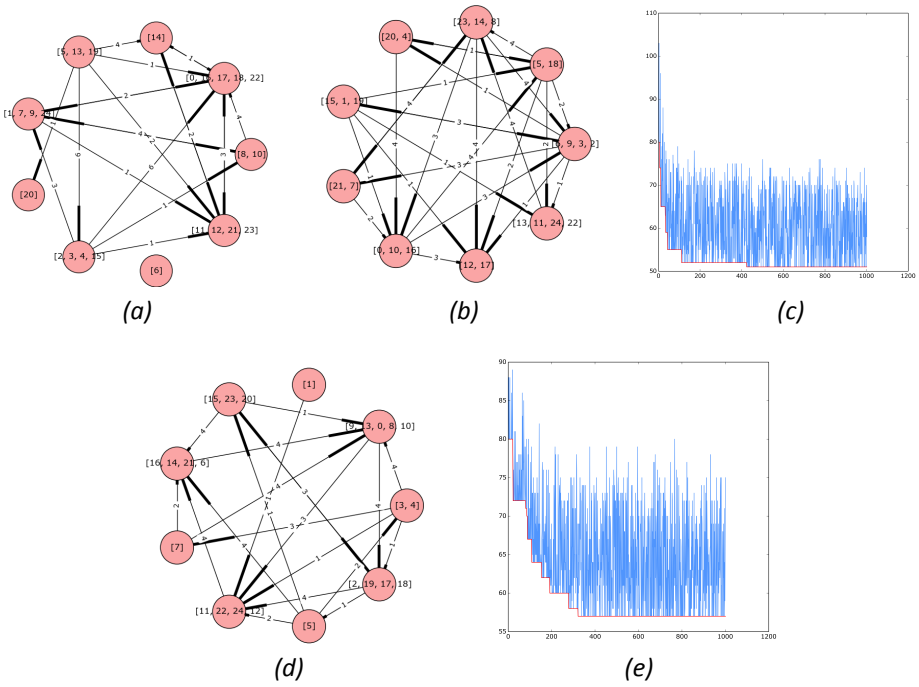


Figure 4.6 a) initial random clustering, b) optimized clustering and c) cost function progression using random task move, d) optimized clustering and e) cost function progression using task migration

together and optimization steps should be restricted in order to prevent their mixture.

To perform the clustering optimization, we use Clustered Task Graph as a directed graph CTG(V,E) where V is a set of clusters in which each cluster contains a set of tasks, and E is the set of edges between two clusters, encapsulates all the communications between tasks in those clusters. The communication weight of an edge $E = (c_1, c_2)$ in CTG, is the sum of weights of all the outgoing communication from tasks in cluster c_1 towards tasks in cluster c_2. The following optimization objectives are available to the user - where Com_E for representing the set of communication weight of all the edges in CTG and $Util_C$ being the set of utilizations of clusters. The term $Util_{C_i}$ represents the utilization of cluster $i \in [0, N_c]$ where N_c is the number of clusters-:

- $C_1 = \sigma(Com_E) + \sigma(Util_C)$: sum of standard deviations of all communication weights and all cluster utilizations
- $C_2 = max(Com_E) + max(Util_C)$: sum of maximum value of communication and cluster utilization

- $C_3 = C_1 + C_2$
- $C_4 = \frac{1}{N}\sum_{i=1}^{N} Util_{C_i}$: Average cluster utilization
- $C_5 = max(Com_E)$: Maximum communication edge weight between clusters.

In case of using greedy search algorithms, it is important to carefully design the algorithm moves (the change in the solution in each step). To this end, two different moves have been investigated in this work; 1) Random task movement, where a task would be randomly selected and moved from its current cluster to an randomly selected cluster and 2)Group migration, where two tasks are randomly selected from two different clusters and are migrated to their cluster to the other cluster. Fig. 4.6 shows the optimization process of an initial random Clustered Task Graph (Fig. 4.6 a.) using random task movement (Fig. 4.6 b and c) and task migration (Fig. 4.6 d and e) for cost function C_2. It became obvious that each of the task movements has its own advantages and disadvantages. Since greedy heuristics are used, random task movement can not move across a path where at least one move is not feasible under greedy constraints (*i.e.* at least one move will result in worse than the current solution). Also starting with task migration, the optimization gets stuck in situations where only a single task move is required to reach a better solution. To overcome this problem, a hybrid of these moves or use of other search methods such as Tabu Search is required (this investigation is left for future works).

4.5.3. Mapping and Scheduling

The task of mapping consists of assigning clusters from CTG (and the tasks in that cluster) to a PE within AG and assigning a the edges in the Task Graph (TG) to a path of links and routers in the network. This process is followed by scheduling of the tasks on PEs and packets on links and routers. To assign paths in the network to TG edges, the routing graph RG is used. Using RG, paths under deterministic/adaptive, minimal/non-minimal routing from the source to destination can be extracted using conventional graph algorithms. The process of the mapping can be formulated as:

$$M(App, Arch) : TG(V, E) \rightarrow AG(P(F), C(F))$$

or in case of using task clustering:

$$M(App, Arch) : CTG(V, E) \rightarrow AG(P(F), C(F))$$

Where $Arch : AG(PE(F), C(F))$ describes a fault prone architecture graph. $PE(F)$ is a set of $\{pe_i(F), \forall i \in [0, N_{pe}]\}$ and $pe_i(F)$ is describing the ith

Processing Element under fault configuration F. And $C(F)$ is the set of $\{c_j(F), \forall j \in [0, N_c]\}$ where each $c_j(F)$ is representing the jth communication link under fault configuration F. It does not matter to directly map TG on AG or perform mapping on CTG, in the end a PE would be assigned for each task on TG for execution, that is the main reason that we do not change our representation for mapping for different cases. This work only focuses on deterministic routing algorithms, since adaptive routing would result in probabilistic scheduling of the packets.

For TGs with data dependency between tasks, different mapping algorithms have been implemented in the tool:

- **Greedy Heuristics:** are mostly LS and ILS. The main idea for using such algorithms is to find a feasible solution with acceptable quality based on the current system health, every time a system failure occurs.
- **Meta-Heuristics:** The meta-heuristics are used as a benchmark for comparing greedy heuristic solutions. Simulated annealing is chosen for this purpose. Different cooling schedules and cost functions have been implemented for SA.

Once a mapping is performed, scheduling of the tasks will follow in order to provide the evaluation metrics for the cost of a mapping. All the conflicts over the shared resources (such as network links, router resources etc.) would be resolved during scheduling. The process of scheduling will start from the root of the TG and will perform ASAP scheduling on each used resource considering the constraints from the units to which this scheduling is depending. For example, scheduling of a flit on a link depends on the scheduling of the upstream router. Using $Makespan_L$ to denote the list of scheduling lengths for all the links, $Makespan_N$ to denote the list of scheduling lengths for all the nodes, $Util_N$ as set of utilization of each node in the network and $Util_L$ as set of utilization of each link in the network. Different cost functions are available to the user:

- $C_1 = \sigma(Makespan_L)$: standard deviation of makespans of packets on the links. This is mainly used for balancing the schedule length on all the network links.
- $C_2 = \sigma(Makespan_N)$: standard deviation of makespans of tasks on the nodes. This cost function is used for balancing the schedule length on all the network nodes.
- $C_3 = C_1 + C_2$
- $C_4 = Max(Makespan_L) + Max(Makespan_N)$: sum of maximum makespans of packets on the links and tasks on the nodes.
- $C_5 = C_3 + C_4$

- $C_6 = \sigma(Util_N)$: standard deviation of utilization of nodes. Which is used for load balancing on network nodes.
- $C_7 = \sigma(Util_L)$: standard deviation of utilization of links. Which is used for load balancing on network links.
- $C_8 = C_6 + C_7$

The cost function severely penalizes the mapping/scheduling solutions that violate critical tasks' deadline. However, these solutions are not totally discarded.

For the simulated annealing algorithm, if the new solution has equal or smaller cost it will always be accepted. In case the new solution has higher cost than the current solution, the probability of choosing the new solution is calculated using Metropolis criterion [100]:

$$P_t = e^{\frac{C_{current} - C_{new}}{T_t}}$$

Where $C_{new}, C_{current}$ represent the cost of the new and current solutions respectively and T_t represents the temperature of the process in step$t$. The following cooling methods are investigated for temperature control:

- linear: the temperature decreases by a fixed amount at each step. More formally, temperature at step t can be calculated as $T_t = T_0 - nt$ [101].
- Exponential: the temperature in step t is calculated as $T_t = T_0 \alpha^t$ [101]
- Logarithmic: the temperature in step $t > 1$ is calculated as $T_t = \frac{T_0}{\log t + 1}$ [101]
- Aart the cooling mechanism is based on method introduced in [102].
- Huang: its similar to Aart's cooling schedule except that using two counters, it dynamically changes the size of the queue based on the behavior of the system.
- Custom made adaptive: the algorithm monitors a moving window of the cost progression and interpolates a line through the data points. In case the variation is reduced (slope of the line is smaller than a predefined value) in the window, the algorithm starts cooling down with a predefined cooling factor (α) [103]. This will allow the process to start from relatively high temperatures and the system will adapt to the situation of the system

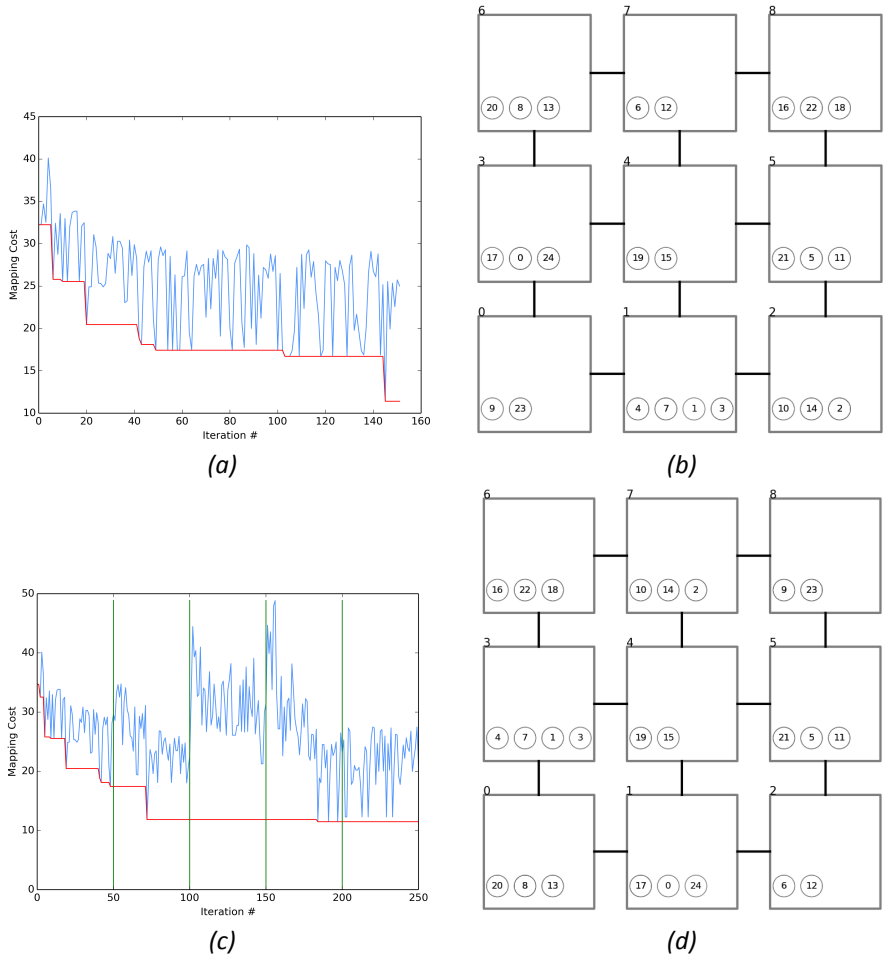


Figure 4.7 Cost progression for a) local search and c) iterative local search algorithm, Final mapping for b) local search and d) iterative local search

Figures 4.7 and 4.8 show examples of mapping and cost function progression of the greedy heuristics and meta-heuristics under C_8 respectively. The Green lines in Fig. 4.8 visualizes the temperature value at each step. Fig. 4.8 experiments use annealing factor $\alpha = 0.99995$. For custom annealing the experiment considers 2000 moves window before each decrease in temperature and 0.01 as the threshold slope. The process stops if after no improvement in the past 30,000 steps. Huang schedule uses similar parameters, however, the process terminated after exceeding 130,000 steps. The experiments show that although the greedy heuristics do not provide the same quality of solution as the Meta-heuristics, the solution quality-execution time ratio of the greedy heuristics makes them very practical in online global application management.

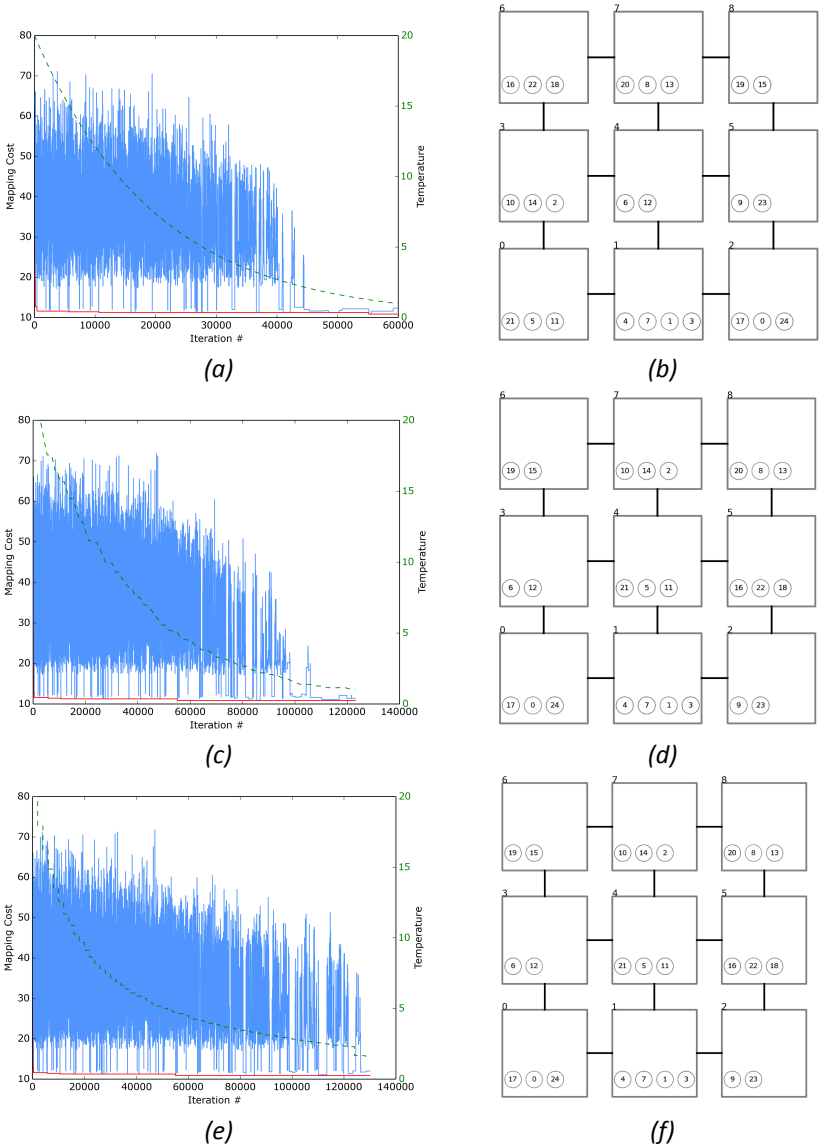


Figure 4.8 Cost reductions for SA using a) Exponential c) Custom and e) Huang's annealing and Final mappings using SA b) Exponential, d) Custom and f) Huang's annealing

For independent TGs, the following heuristics for mapping are also available:

- **Min-Min:** sorts the tasks based on Worst Case Execution Time (WCET) and maps the shortest task on the list, to the node with the smallest completion time (fastest node for that task).
- **Max-Min:** sorts the tasks based on WCET and maps the longest task on the list, to the node with the smallest completion time (fastest node for that task).
- **MET:** chooses tasks randomly from the TG and maps them on the fastest node.
- **MCT:** chooses tasks randomly from the TG and maps them onto the node with the smallest completion time.

However, one of the requirements of such approaches is to provide support for different PE speed in the system. This has been accomplished by adding a speedup factor in the architecture graph as node attribute.

In the end, the tool automatically generates Gantt charts of the final scheduling of the tasks. Please refer to Appendix A for examples of the generated scheduling Gantt charts.

4.5.4. Providing partial mapping

To decrease the time of application deployment it is crucial to be able to update the mapping cost function in order to be able to find the closeness of a solution to the previous mapping. This will reduce the amount of task migration on the system. To this end, the proposed framework incorporates the idea of hamming distance of strings (for example see [104]) to generate an additional term *"distance"* to the mapping cost function. To calculate the distance of two mapping solutions, a simple list would be generated where element *i* of the list represents the processing elements id where task T_i is mapped:

PE_g	PE_h	...	PE_i	PE_j

Where $g,h,i,j \in [0,n]$ and n represents number of nodes in the network. If we have $mapping\_list_0$ and $mapping\_list_1$ the distance of the two mappings can be calculated using Alg. 5.

Using the distance as a metric, it is possible to penalize the mappings that provide good quality solutions (for the initial cost function) but require more task migration. Once the mapping optimization finishes, it would be necessary to only apply partial task migration and skip full-system reconfiguration. More formally, if the calculation of a reconfiguration takes T_{RL} where

Algorithm 5: average connectivity calculation algorithm

```
distance = 0
for i ∈ [0, len(mapping_list_0)] do
    if mapping_list_0[i] != mapping_list_1[i] then
        distance += 1
    end
end
return distance
```

$$T_{RL} = T_{MapAlg} + T_{Mapp}$$

Where T_{MapAlg} is the time required for calculation of the mapping algorithm and T_{Mapp} is the time for application deployment on the system. Using the proposed method this time would be reduced to:

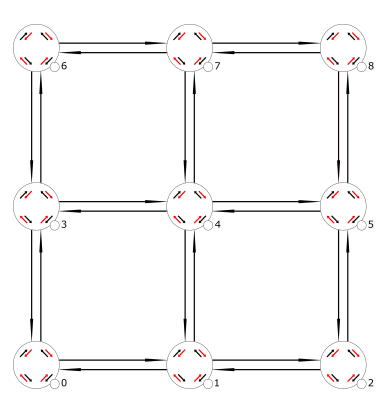
$$T_{RL} = T'_{MapAlg} + T_{ParExt} + T_{ParMap}$$

Where T'_{MapAlg} is the time required for mapping optimization while using the distance metric. T_{ParExt} is the time required to find the difference between current mapping and the old mapping (to find how many tasks should be migrated). And T_{ParMap} is the time required to apply the necessary task migrations.

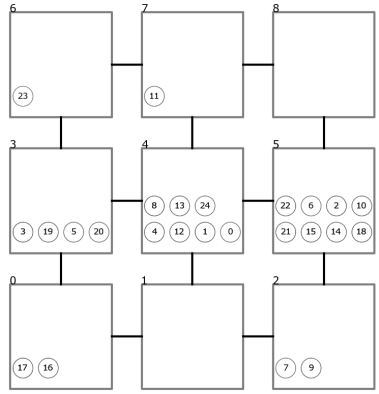
An example of this process is depicted in Fig. 4.9 and Fig. 4.10, where the cost function is heavily penalized by the distance factor. Fig 4.9 1 shows the initial state of the system where 25 tasks are mapped on a fully functional 3×3 mesh network. Figures 4.9b-c and 4.10a-c show network's links degradation in the system health map and the mapping provided for that configuration using such heavy distance penalization. By naming the successive mappings in Figures 4.9 and 4.10, M_0 (for network with no failures) to M_5 (for network with five link failures), its possible to calculated the distance between mappings are as follows:

- $Distance(M_0, M_1) = 5$
- $Distance(M_1, M_2) = 3$
- $Distance(M_2, M_3) = 3$
- $Distance(M_3, M_4) = 3$
- $Distance(M_4, M_5) = 7$

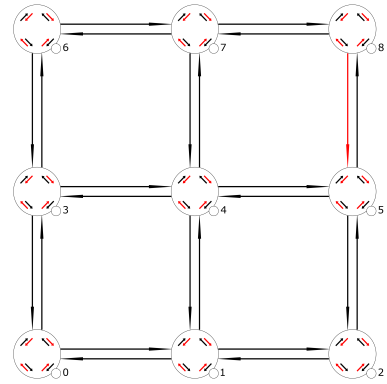
This means that it is possible to go from mapping M_0 for a fault free network depicted in Fig. 4.9 a.2 to M_1 with the one failed link (see Fig. 4.9 b.1 for the system health map representation) depicted in Fig. 4.9 b.2 by migrating only



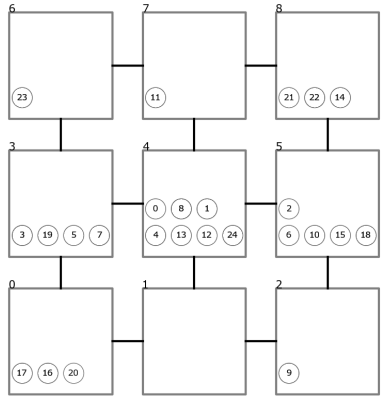
(a.1)



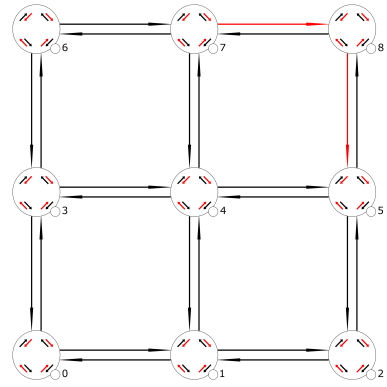
(a.2)



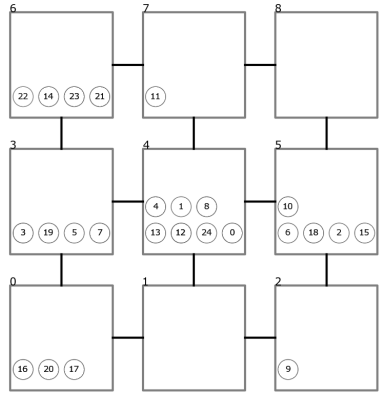
(b.1)



(b.2)

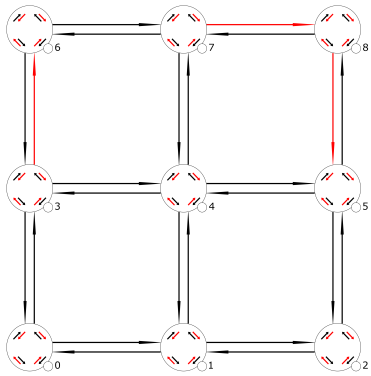


(c.1)

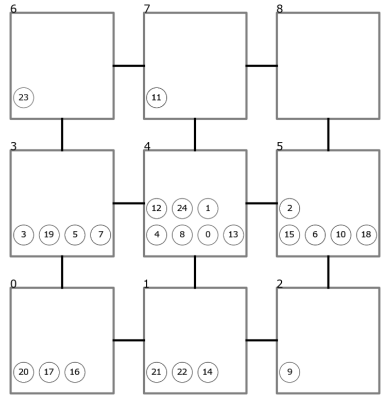


(c.2)

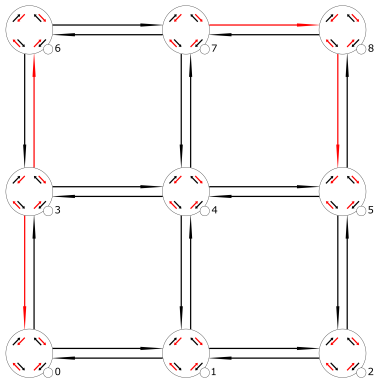
Figure 4.9 Successive mappings using distance between mappings. Figures a.1, b.1 and c.1, show the link failures in the system health map. figures a.2, b.2 and c.2 show the task assignment for each processing element under the fault condition.



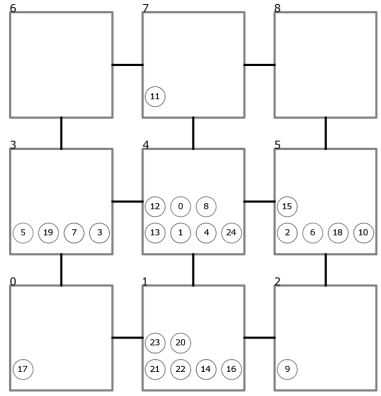
(a.1)



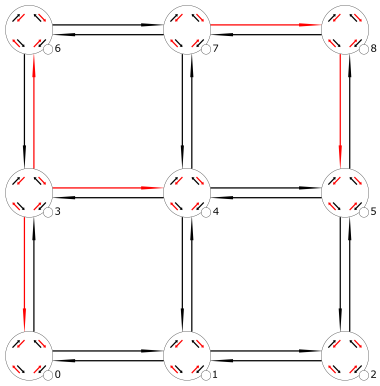
(a.2)



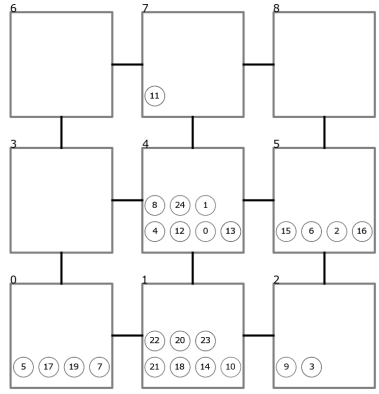
(b.1)



(b.2)



(c.1)



(c.2)

Figure 4.10 Successive mappings using distance between mappings. Figures d.1, e.1 and f.1 show the link failures in the system health map. figures d.2, e.2 and f.2 show the task assignment for each processing element under the fault condition.

5 tasks. These experiments show the considerable advantage of the proposed partial mapping extraction from 25 task migrations (in case of a system-wide re-mapping) to 3-7 task migrations which reduces (4.8 task migrations on average resulting in 80.8% reduction compared to complete reconfiguration of the system).

4.5.5. Improving the mapping latency

It is important that the SHMU assigns a severity metric to the received fault information and takes proper action accordingly. This means that SHMU can either ignore the fault (in case of no severe effect on the system), or can issue an order to Mapper-Scheduler Unit (MSU) to prepare another mapping. A mapping can be ordered in case that SHMU predicts a fault becoming permanent in near future and assigns high severity to it. This mapping either should be used immediately (in case of reported faults having severe effect) or in case of prediction of occurrence in short future should be stored in a memory designated for Most Probable Mappings. It should be noted that no scheduling information is stored at this stage. [44] has proposed a prediction method for fault prediction and pre-processing of routing table updates. However this work only focuses on the link faults.

In this work, it is assumed that there exist a prediction method (using machine learning or other methods) for predicting future faults. Then its possible to define Most Probable Faults Set (MPFS) as a set of f_i most probable faults that can occur to each $AG(PE(F), C(F))$ based on diagnosis information received from the architecture under a certain mapping $M(App, Arch(F))$. These most probable faults would be stored in the SHMU database. More formally, set of next Most Probable Mappings (MPM) is defined as:

$$MPM : \{M(App, Arch(F + f_i)) | \forall f_i \in MPFS\}$$

To calculate these mappings, SHMU updates the System Health Map with a certain fault (which is predicted to happen), then issues a map and store order to the Mapping-Scheduling unit. Upon receiving this, the MSU will calculate the mapping and store it with a fault tag (which encapsulates the fault configuration) in MPM memory. Afterwards, SHMU would return the SHM to its original state.

Upon receiving a ”Map and Store” order from SHMU, MSU starts calculating the mapping by choosing proper heuristic according to information received from SHMU. For each mapping, we define a list which ith element of it represents the processing element id chosen for mapping for the ith task. Each mapping also has a fault tag that describes the fault configuration associated with it. This will result in the following data-structure for each mapping in MPM:

Table 4.2 Tool performance with growing size of TG

TG Size	AG Size	O.P.S.	CTG Size	CTG Opt. Time (sec)	R.P.S.	Map Opt. Time (sec)
10	3×3	$3.4e9$	9	1	$3.8e8$	1
20	3×3	$1.2e19$	9	1	$3.8e8$	6
40	3×3	$1.4e38$	9	2	$3.8e8$	12
80	3×3	$2.1e76$	9	2	$3.8e8$	29
160	3×3	$4.7e159$	9	4	$3.8e8$	73

Fault Tag	p_{t_0}	p_{t_1}	...	p_{t_m}

Different approaches can be taken for Fault Tag generation. The easiest is to hash the SHM into a fixed size string, since this tag is not going to be re-used to re-calculate the fault information, classic one way hashes or data compression can be used. The size of this memory depends on the application size and system requirements. Bigger number of mappings require more number of entries in MPM which results in bigger memory size.

Upon receiving a "Map and Deploy" order from SHMU, MSU first checks the fault configuration with the pre-calculated mappings tags in MPM. In case of having a hit in MPM, MSU goes further and extracts the difference with the current mapping in order to perform a partial mapping (and scheduling, since no scheduling information is kept in MPM) and then deploys the mapping on PEs (refer to section 4.5.4). In case of not finding the fault tag in MPM, MSU starts calculating the mapping for the current fault configuration, performs partial mapping extraction and deploys it on PEs.

We define reconfiguration latency T_{RL} as the time needed for mapper/scheduler to reconfigure the architecture and map the application after a remap order is received. In cases that the occurred fault is not in MPFS, the reconfiguration latency would be:

$$T_{RL} = T_{MapAlg} + T_{ParExt} + T_{ParMap}$$

Where T_{MapAlg} is the time needed for computation of a new mapping and scheduling algorithm (Since scheduling information is not kept in the MPM). T_{ParExt} is the time needed for extracting partial mapping and T_{ParMap} describes the time required for applying partial mapping on the architecture (this partial mapping consist of migrating tasks). If there is a hit in MPM, the reconfiguration latency T_{RL} would be:

$$T_{RL} = T_{fetch} + T_{Schd} + T_{ParExt} + T_{ParMap}$$

Where T_{fetch} refers to time required to fetch the mapping from the memory and T_{Schd} is needed for calculating scheduling times. T_{Schd} is needed because we are not storing scheduling times for the tasks to save memory. By using ASAP

Table 4.3 Tool performance with growing size of AG

TG Size	AG Size	O.P.S.	CTG Size	CTG Opt. Time (sec)	R.P.S.	Map Opt. Time (sec)
80	2×2	$1.46e48$	4	2	256	17
80	3×3	$2.1e76$	9	2	$3.8e8$	30
80	4×4	$2.1e96$	16	3	$1.84e19$	39
80	5×5	$6.8e111$	25	3	$8.8e34$	50
80	6×6	$3.1e124$	36	3	$1e56$	70

scheduling algorithm the system would regenerate the same scheduling from the mapping which is of the complexity of $O(n)$ since the task graph nodes are already sorted for ASAP scheduling. This would enable the system to recover from a failure $T_{MapAlg} - (T_{fetch} + T_{Schd})$ time units faster. It is important to note that in case of performing mapping, the system has to run the scheduling algorithm once in each mapping step in order to calculate the cost of mapping since scheduling length is an important factor in the mapping cost function. This would make the gap between these two terms rather significant. Even though these mapping heuristics are fast, they need tens (sometimes hundreds) of steps to find a suitable solution. Tables 4.2 and 4.3 show the performance of the mapping and scheduling algorithm for different sizes of TG and AG on a 64-bit linux machine, running on a 3.4 GHz Intel©Core™-i7-6700 CPU with 32GB of Memory. It is clear that the required time for running such algorithms on the field on limited resources would require much longer execution time, which makes this proposal much valuable. Another important note is that this improvement is for the best case scenarios, In the cases of MPM miss, the same long mapping scheduling calculation process is needed. Therefore, the performance of this approach is heavily dependent on how well the system can predict the Most Probable Faults.

Test Application Mapping

The testing environment in "SoCDep^2" works based on the concept introduced in [105] where a graph-based methodology for testing the components of a system using unreliable components is proposed. In case of a Network-on-Chips, such graph nodes which test each other are the cores (Processing Elements) of each tile. In this testing model, nodes of the network detect faults either in one-stop (called one-step diagnosable system) or each node can test the other nodes in a sequential manner (called sequentially diagnosable system)[105]. Both approaches are supported by "SoCDep^2".

Based on the approach presented in [105], a Test Task Graph (TTG) is generated and is mapped in one step on the network nodes (since it is clear which testing task belongs to which network node). Fig. 4.11 shows an example of a sequentially diagnosable test task graph for a 3×3 Mesh NoC. Each node in the network has a test task and will send the results according to the data dependency to other nodes. The mapping of the tasks are predefined in the task graph however, the scheduling is not decided. It is important to make sure that the scheduling of these tasks do not interrupt the normal system scheduling.

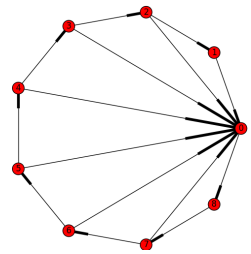
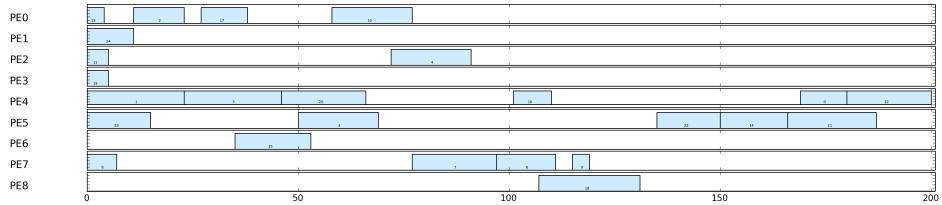
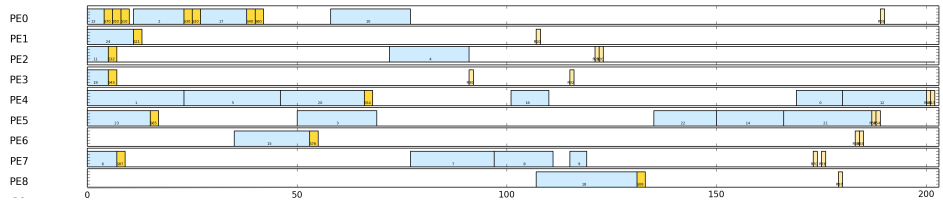


Figure 4.11 example of sequentially diagnosable system



(a) Scheduling of the application on the processing elements



(b) Scheduling of the test application on the processing elements in idle times of Nodes

Figure 4.12 Example of scheduling test tasks in idle times of the processors

The test tasks will be scheduled in idle times on each core (this is done after scheduling the TG) to provide the guarantee that it will not disrupt the normal schedule. Fig. 4.12 shows the insertion of test tasks in idle time of processing elements (test tasks are marked in yellow). Similarly, the idle time of the routers is utilized for scheduling the communication packets between the test tasks over the network. The packets are scheduled to be injected in the network without disrupting the application's packet flow.

4.5.6. Environment Simulator

To test each of the above mentioned approaches, it is crucial to simulate the system under different fault injection rates and observe the degradation of the system during its lifetime. In this section the characteristics of the simulation

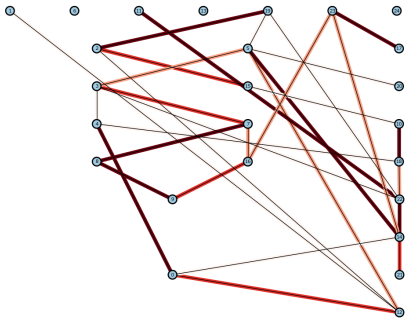


Figure 4.13 Task graph for simulator example

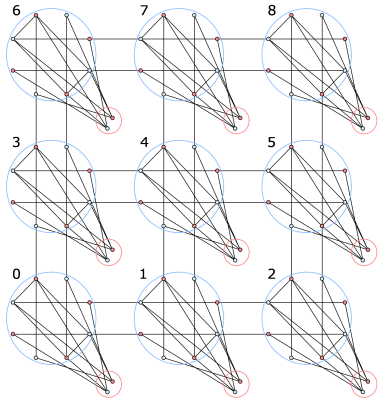


Figure 4.14 Routing graph for simulator example

support (using SimPy library [106]) of SoCDep^2 is discussed and example of the behavior of the system would be presented.

To elaborate the process of the tool, a randomly generated task graph is considered with 25 tasks with 35 data transactions (represented by the edges on the task graph). The resulting task graph is presented in Fig 4.13. The tasks dependencies are shown by graph edges and tasks are executed from top down. Darker and thicker edges show higher dependency between the tasks.... The application is mapped on a 3×3 2D mesh network (see Fig. 4.2) is considered under X-Y routing algorithm. The system starts with all components being fault free. Based on the above information Routing Graph of the 3×3 network is generated (presented in Fig. 4.14).

Mapping is performed using local search (LS) algorithm in 1000 moves starting from a random mapping where the mapping cost function is $C_4 = Max(Makespan_L) + Max(Makespan_N)$. The initial random mapping, and optimized mapping along with the cost function progress are shown in 4.15 a-c. The final scheduling of the tasks on the network nodes (Processing Elements) is shown in Fig. 4.15 d. As shown in Fig. 4.15 d, the mapping uses almost all the processing elements (except PE1).

At this point the simulator starts running the application and initiates fault injection with mean time of 0.1 fault per clock cycle. Every time the system health monitoring unit receives information about a permanent failure, a new mapping and scheduling is prepared and the simulator starts executing the application from the beginning.

To see the performance degradation of the system, the system status is evaluated after half of the network links are marked as permanently damaged. Fig. 4.16 a-c, show the system health map, the mapping and mapping cost progression. Fig. 4.16-d, depicts scheduling of the application on the processing

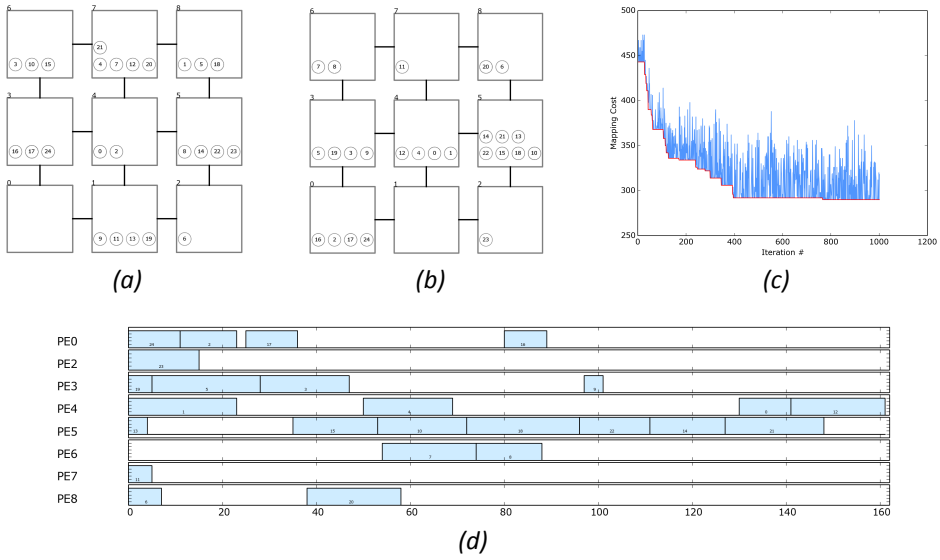


Figure 4.15 a) initial random mapping, b) optimized mapping, c) mapping cost function progress and d) scheduling of the tasks on the PEs for a fault-free system

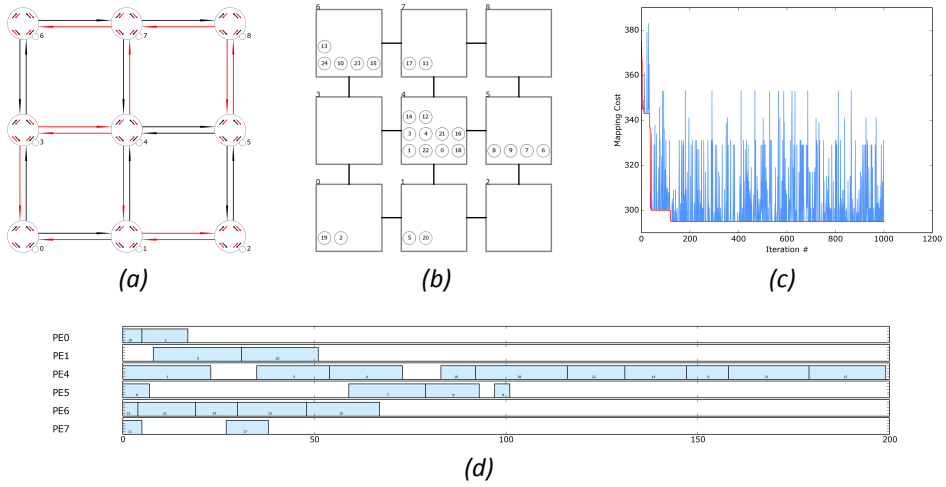


Figure 4.16 a) system health map representation, b) mapping, c) mapping cost progress and d) scheduling of the tasks on the PEs after 12 permanent link failures.

elements. Note the concentration of tasks in several processing elements and the increase in the scheduling time due to lack of available resources. Full scheduling Gant charts for the nodes, routers and links for both cases

are available in Appendix 4.6. The colors on the links and routers represent individual packets going through the network.

4.6. Chapter summary

Considering growing size of the components on chips, it is not possible to manage all the faults locally.

The local fault management mechanisms do not have a global view of the system and can not take into account the effect of the local decisions on the system behavior. This might lead to situations that a series of local reconfigurations based on the limited available information, will lead to the non-optimal solution from global point of the view. Moreover, many of such local approaches rely on configuration information from a global system manager. Therefore it is crucial to have a global fault manager which collects the fault information from the lower abstraction layers, maintains a holistic view of the system and performs system-wide reconfiguration either by hardware reconfiguration or by performing partial or full application mapping/scheduling. Also by only focusing on the local fault manager, the effects of such faults on the system will be ignored. Moreover the collaboration of local and global fault management systems should be investigated in order to identify the shortcomings of each of these mechanisms in more detail and to optimize the distribution of such tasks over different layers of abstraction. To examine the effects of such global manager, Its necessary to perform high-level simulation of its behavior.

This chapter discussed a framework for investigating the global fault management approaches by simulating them under different fault environment. The proposed framework provides possibility of modeling different applications and system architectures. The $SoCDep^2$ framework utilizes a System Health Monitoring unit which maintains the health map of the system's components. This information would be used to calculate different mapping solutions under faulty system configuration. The proposed framework provides the user with a wide range of mapping algorithms such as local search, Iterative local search, Simulated Annealing (used for benchmarking) for tasks with data dependencies along with Min-Min, Max-Min, MET and MCT for independent task mappings.

Two approaches for reducing the down-time of the system and increasing the availability of the system are proposed: 1) A partial mapping technique is used for reducing the number of task migrations (up to 80%) and 2) An infrastructure for calculating and storing probable mappings based on the available fault information. The chapter concluded by introduction to the simulator included in the framework for testing the proposed approaches under different fault injection ratios.

Conclusions

Growing the number of components in the systems and shrinking feature size has affected systems susceptibility to faults. The shortened system's life and increased number system failures during its lifetime has highlighted the need for novel fault tolerance methods. In recent years Network on Chip (NoC) communication paradigm has been accepted widely in System on Chips. NoCs provide an scalable solution for communication bottleneck in multi/many-core systems by transmit information in form of packets in an on chip network and eliminating long wires over the chip. Considering the trends in the component lifetime, NoC based systems require light-weight fault tolerance mechanisms which can provide (1) guarantees for hard real-time systems and (2) a graceful performance degradation in presence of faults. This work focuses on providing scalable solutions for cross-layer fault management for NoC based SoCs. The proposed mechanism consists of local and hybrid hardware-based fault management mechanisms which provide fast response time along with a system health monitoring unit which can fill the shortcomings of limited information available to local fault management systems. The local fault management schemes are hardware based solutions that are oblivious to the system's state and act based on the information obtained from their close vicinity. Hybrid approaches provide the advantages of closeness to the hardware but they are configured by the global system manager which turns them into one of the most effective assets in cross-layer fault management schemes. The main contributions of this work can be listed as:

- The fault detection mechanisms for control part of the circuit considered in this work (concurrent online checkers) provide detailed fault localization information but have a large hardware overhead. In this work a minimization framework for such checkers is proposed. A new metric, namely coverage density has been introduced for selecting the checkers during the optimization process. Also new concept of dominant checkers have been introduced for speeding up the process by reducing the search space of the problem. The experimental result shows that using the proposed methods, it is feasible to reduce the area of the checkers bellow 150% of the pseudo-combinational part of the control part of the circuit while guaranteeing 100% fault coverage. Which in-turn makes the

checker circuits an scalable solution for low latency fault detection in NoCs.

- The large number of fault diagnosis signals in the router (specially coming from the concurrent online checkers) require more lightweight hardware based classification methods. This work extends the existing works for hardware-based online fault classification to make them more practical for abstracted checker information as well as the links. The proposed fault classifier only adds 1% to the area of the router. This low hardware overhead, enables deploying fault classifiers on all the key fault information signals.

- The current metric available in the literature for comparing adaptivity of different turn-model-based routing algorithms only considers minimal path routing in the networks. This work extends the degree of adaptiveness used in minimal path routing to cover non-minimal path routing as well. This metric provides the possibility of further classification of turn model behavior under different link-failure scenarios.

- In the current state of the art, several 2D uniform turn-model based routings are identified. However, an enumeration and evaluation of all possible possible cases was missing. This work provides an enumeration of all deadlock free, 2D turn-model based routing algorithms which provide full connectivity in 2D mesh networks. Later, these turn models are compared in terms of their robustness to all configurations of link failure in the network. Using this information, the global system health manager can select an optimal routing model given a fault configuration.

- Another pressing issue in the field of NoC based SoC is partial or full-scale network failure due to faults in the data-path of routers; If a router fails to route or miss-route a packet or parts of a packet due to faults, it is possible that all the communication in the upstream routers to be stalled. This work provides a local fault handling mechanism to protect the network from such congestion scenarios using packet dropping. The experiments show that even if 80% of packets are dropped due to faults on the network links, the network can still operate without congestion.

- It is possible that a routing algorithm provides a viable path from a source to destination but the local routing units might fail to find the path due to their lack of global information. This will lead to congestion in the network due to packets being routed in undesired (but absolutely legal) directions. This work proposes a novel method to guarantee that the injected packets in the network would reach their destination in faulty networks. This method works on arbitrary number of faulty links in combination with any turn-model-based routing algorithm.

- It is crucial to evaluate the effects of faults on NoC based SoC systems at the application layer along with effectiveness of global fault management mechanisms to handle such situations. This work provides a new simulator for NoC based SoCs for modeling global fault management under mixed-criticality constraints.

- Since router architectures are very regular, it is possible to reuse functioning resources that have been cut-off due to their neighboring units. Such a task requires a local resource management and evaluation of the effects of the supporting infrastructure. This work proposes an evaluation framework to compare their effect on reliability of the router.

The contributions in this work, highlights the feasibility and effectiveness of cross-layer dependability approaches in NoC based SoCs. However, there are still many open issues regarding dependability in such systems which will be considered as future work:

- Currently the information extracted from the checkers are abstracted, classified and forwarded to the higher abstraction layer. However, as we saw in case of link faults, faults in the control-part can paralyze the network. New mitigation mechanisms are required for the control part which can use checkers' fault information.

- Quality of service in NoC based mixed-critical systems is of utmost importance. Specially considering the effects of faults in such systems. Methods of providing guarantees regarding maximum network latency and throughput in presence of faults should be investigated.

- The dependability of systems can not be considered without considering system's security. It is important to consider NoC security while addressing fault tolerance and dependability.

REFERENCES

[1] P. Maris Ferreira, H. Cai, and N. Lirida, "Reliability Aware AMS / RF Performance Optimization," in *Performance Optimization Techniques in Analog, Mixed-Signal, and Radio-Frequency Circuit Design*, M. H. F. Mourad Fakhfakh, Esteban Tlelo-Cuautle, Ed., Oct. 2014. [Online]. Available: https://hal-centralesupelec.archives-ouvertes.fr/hal-01222104

[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] P. Saltarelli, B. Niazmand, J. Raik, R. Hariharan, G. Jervan, and T. Hollstein, "A framework for comprehensive automated evaluation of concurrent on-line checkers," in *Digital System Design (DSD), 2015 Euromicro Conference on*, Aug 2015, pp. 288–292.

[4] 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.

[5] T. Kogge, "Graph-based methods for property evaluation of on-chip routing algorithms and implementation of a scalable network-on-chip dependability layer," Bachelor Thesis, Technischen Universität Darmstadt, October 2015.

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

[7] C. L. Chen, Y. H. Chen, and T. Hwang, "Communication driven remapping of processing element (pe) in fault-tolerant noc-based mpsocs," in *2016 2nd International Conference on Wireless and Telematics (ICWT)*, Aug 2016, pp. 666–671.

[8] C. L. Chou and R. Marculescu, “Farm: Fault-aware resource management in noc-based multiprocessor platforms,” in *2011 Design, Automation Test in Europe*, March 2011, pp. 1–6.

[9] E. Dubrova, *Fault-Tolerant Design*, 1st ed. 978-1-4614-2113-9: Springer-Verlag New York, 2013.

[10] S. R. Nassif, N. Mehta, and Y. Cao, “A resilience roadmap,” in *2010 Design, Automation Test in Europe Conference Exhibition (DATE 2010)*, March 2010, pp. 1011–1016.

[11] D. K. Schroder and J. A. Babcock, “Negative bias temperature instability: Road to cross in deep submicron silicon semiconductor manufacturing,” *Journal of Applied Physics*, vol. 94, no. 1, pp. 1–18, 2003. [Online]. Available: https://doi.org/10.1063/1.1567461

[12] A. B. Achballah, S. B. Othman, and S. B. Saoud, “Problems and challenges of emerging technology networks−on−chip: A review,” *Microprocessors and Microsystems*, vol. 53, pp. 1 – 20, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0141933116303593

[13] S. P. Azad, B. Niazmand, 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)*, May 2017, pp. 1–2.

[14] 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)*, April 2017, pp. 48–53.

[15] S. P. Azad, B. Niazmand, K. Janson, T. Kogge, J. Raik, G. Jervan, and T. Hollstein, “Comprehensive performance and robustness analysis of 2d turn models for network-on-chips,” in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2017, pp. 1–4.

[16] T. Hollstein, S. Azad, T. Kogge, H. Ying, and K. Hofmann, “Nocdepend: A flexible and scalable dependability technique for 3d networks-on-chip,” in *Design and Diagnostics of Electronic Circuits Systems (DDECS), 2015 IEEE 18th International Symposium on*, April 2015, pp. 75–78.

[17] 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.

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

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

[20] "Project bonfire network-on-chip," https://github.com/Project-Bonfire, 2015.

[21] Plasma CPU. http://plasmacpu.no-ip.org.

[22] C. J. Glass and L. M. Ni, "The turn model for adaptive routing," in *Computer Architecture, 1992. Proceedings., The 19th Annual International Symposium on*, 1992, pp. 278–287.

[23] S. Rodrigo, S. Medardoni, J. Flich, D. Bertozzi, and J. Duato, "Efficient implementation of distributed routing algorithms for nocs," *Computers Digital Techniques, IET*, vol. 3, no. 5, pp. 460–475, September 2009.

[24] C. Chen, Y. Fu, and S. Cotofana, "Towards maximum utilization of remained bandwidth in defected noc links," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 2, pp. 285–298, Feb 2017.

[25] B. Bhowmik, J. K. Deka, and S. Biswas, "Towards a scalable test solution for the analysis of interconnect shorts in on-chip networks," in *2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Sept 2016, pp. 394–399.

[26] J. Liu, J. Harkin, Y. Li, and L. Maguire, "Online fault detection for networks-on-chip interconnect," in *2014 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, July 2014, pp. 31–38.

[27] 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*, April 2007, pp. 1–6.

[28] S. R. Suja and M. Deivakani, "Testing of fifo buffer of noc router using bist," in *2017 IEEE International Conference on Electrical, Instrumentation and Communication Engineering (ICEICE)*, April 2017, pp. 1–6.

[29] A. Fatima and S. M. Waseem, "Cellular automata based built-in-self test implementation for star topology noc," in *2017 11th International Conference on Intelligent Systems and Control (ISCO)*, Jan 2017, pp. 45–48.

[30] M. Balboni and D. Bertozzi, “Transparent lifetime built-in self-testing of networks-on-chip through the selective non-concurrent testing of their communication channels,” in *Proceedings of the 2Nd International Workshop on Advanced Interconnect Solutions and Technologies for Emerging Computing Systems*, ser. AISTECS ’17. New York, NY, USA: ACM, 2017, pp. 12–17. [Online]. Available: http://doi.acm.org/10.1145/3073763.3073765

[31] A. Prodromou, A. Panteli, C. Nicopoulos, and Y. Sazeides, “NoCAlert: An On-Line and Real-Time Fault Detection Mechanism for Network-on-Chip Architectures,” pp. 60–71, 2012.

[32] 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 Transactions on Architecture and Code Optimization*, vol. 13, no. 2, pp. 1–26, jun 2016. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2952301.2930670

[33] 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, pp. 104:1—-104:30, mar 2014. [Online]. Available: http://doi.acm.org/10.1145/2514871

[34] Jinfu Xu, Yanxi Hang, Pengfei Guo, and Qiang Dai, “An unified online fault-tolerant mechanism for FIFO faults in network-on-chip router,” in *2016 13th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*. IEEE, oct 2016, pp. 1419–1421. [Online]. Available: http://ieeexplore.ieee.org/document/7998756/

[35] L. Huang, X. Zhang, M. Ebrahimi, and G. Li, “Tolerating transient illegal turn faults in NoCs,” *Microprocessors and Microsystems*, vol. 43, pp. 104–115, jun 2016. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0141933116000284

[36] C. Killian, C. Tanougast, F. Monteiro, and A. Dandache, “Smart reliable network-on-chip,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 2, pp. 242–255, feb 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6468169/

[37] L. Fiorin and M. Sami, “Fault-Tolerant Network Interfaces for Networks-on-Chip,” *IEEE Transactions on Dependable and Secure Computing*, vol. 11, no. 1, pp. 16–29, jan 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6560056/

[38] N. Alves, Y. Shi, J. Dworak, R. I. Bahar, and K. Nepal, “Enhancing online error detection through area-efficient multi-site implications,” pp. 241–246, 2011.

[39] A. Alaghi, N. Karimi, M. Sedghi, and Z. Navabi, “Online NoC Switch Fault Detection and Diagnosis Using a High Level Fault Model,” pp. 21–29, 2007.

[40] D. Appello, A. Fudoli, V. Tancorre, P. Bernardi, F. Corno, M. Rebaudengo, and M. S. Reorda, “A bist-based solution for the diagnosis of embedded memories adopting image processing techniques,” *Journal of Electronic Testing*, vol. 20, no. 1, pp. 79–87, Feb 2004. [Online]. Available: https://doi.org/10.1023/B:JETT.0000009315.57771.94

[41] 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.

[42] 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.

[43] C. Killian, C. Tanougast, F. Monteiro, and A. Dandache, “Smart reliable network-on-chip,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 2, pp. 242–255, Feb 2014.

[44] J. Silveira, C. Marcon, P. Cortez, G. Barroso, J. M. Ferreira, and R. Mota, “Scenario preprocessing approach for the reconfiguration of fault-tolerant noc-based mpsocs,” *Microprocessors and Microsystems*, vol. 40, pp. 137 – 153, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0141933115001180

[45] “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.

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

[47] E. Cota, L. Carro, and M. Lubaszewski, “Reusing an on-chip network for the test of core-based systems,” *ACM Trans. Des. Autom. Electron. Syst.*, vol. 9, no. 4, pp. 471–499, Oct. 2004. [Online]. Available: http://doi.acm.org/10.1145/1027084.1027088

[48] 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. 6 pp.–.

[49] F. Silva, W. Magalhães, J. Silveira, J. M. Ferreira, P. Magalhães, O. A. Lima, and C. Marcon, "Evaluation of multiple bit upset tolerant codes for nocs buffering," in *2017 IEEE 8th Latin American Symposium on Circuits Systems (LASCAS)*, Feb 2017, pp. 1–4.

[50] A. I. Jutman, 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.

[51] R. Bishnoi, V. Laxmi, M. Gaur, R. Bin Ramlee, and M. Zwolinski, "Fault tolerant routing implementation mechanism for irregular 2d mesh nocs," in *NORCHIP, 2014*, Oct 2014, pp. 1–4.

[52] M. Ebrahimi, M. Daneshtalab, J. Plosila, and H. Tenhunen, "Mafa: Adaptive fault-tolerant routing algorithm for networks-on-chip," in *Digital System Design (DSD), 2012 15th Euromicro Conference on*, Sept 2012, pp. 201–207.

[53] 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.

[54] R. Alizadeh, M. Saneei, and M. Ebrahimi, "Fault-tolerant circular routing algorithm for 3d-noc," in *Technology, Communication and Knowledge (ICTCK), 2014 International Congress on*, Nov 2014, pp. 1–7.

[55] F. Dubois, A. Sheibanyrad, F. Pétrot, 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.

[56] 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.

[57] J. Lee and K. Choi, "A deadlock-free routing algorithm requiring no virtual channel on 3d-nocs with partial vertical connections," in *Networks on Chip (NoCS), 2013 Seventh IEEE/ACM International Symposium on*, April 2013, pp. 1–2.

[58] S. Rodrigo, J. Flich, A. Roca, S. Medardoni, D. Bertozzi, J. Camacho, F. Silla, and J. Duato, "Addressing manufacturing challenges with cost-efficient fault tolerant routing," in *Networks-on-Chip (NOCS), 2010 Fourth ACM/IEEE International Symposium on*, May 2010, pp. 25–32.

[59] 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.

[60] M. Imai and T. Yoneda, "Improving dependability and performance of fully asynchronous on-chip networks," in *2011 17th IEEE International Symposium on Asynchronous Circuits and Systems*, April 2011, pp. 65–76.

[61] H. S. Kia and C. Ababei, "A new fault-tolerant and congestion-aware adaptive routing algorithm for regular networks-on-chip," in *2011 IEEE Congress of Evolutionary Computation (CEC)*, June 2011, pp. 2465–2472.

[62] M. Ebrahimi, M. Daneshtalab, and J. Plosila, "High performance fault-tolerant routing algorithm for noc-based many-core systems," in *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, Feb 2013, pp. 462–469.

[63] S. Zhang, G. Han, and F. Zhang, "Very fine-grained fault-tolerant routing algorithm of noc based on buffer reuse," in *2013 IEEE 4th International Conference on Software Engineering and Service Science*, May 2013, pp. 758–762.

[64] M. Valinataj and S. Mohammadi, "A fault-aware, reconfigurable and adaptive routing algorithm for noc applications," in *2010 18th IEEE/IFIP International Conference on VLSI and System-on-Chip*, Sept 2010, pp. 13–18.

[65] C.-C. Su and K. G. Shin, "Adaptive fault-tolerant deadlock-free routing in meshes and hypercubes," *IEEE Transactions on Computers*, vol. 45, no. 6, pp. 666–683, Jun 1996.

[66] K. hsuan Chen and G. ming Chiu, "Fault-tolerant routing algorithm for meshes without using virtual channels," Journal of Information Science and Engineering, Tech. Rep., 1997.

[67] Z. Jiang, J. Wu, and D. Wang, "A new fault-information model for adaptive minimal routing in 3-d meshes," *IEEE Transactions on Reliability*, vol. 57, no. 1, pp. 149–162, March 2008.

[68] D. Wang, "A rectilinear-monotone polygonal fault block model for fault-tolerant minimal routing in mesh," *Computers, IEEE Transactions on*, vol. 52, no. 3, pp. 310–320, March 2003.

[69] J. Wu and D. Wang, "Fault-tolerant and deadlock-free routing in 2-d meshes using rectilinear-monotone polygonal fault blocks," in *Parallel Processing, 2002. Proceedings. International Conference on*, 2002, pp. 247–254.

[70] E. Wachter and F. Moraes, "Mazenoc: Novel approach for fault-tolerant noc routing," in *SOC Conference (SOCC), 2012 IEEE International*, Sept 2012, pp. 364–369.

[71] E. Wachter, A. Erichsen, A. Amory, and F. Moraes, "Topology-agnostic fault-tolerant noc routing method," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, March 2013, pp. 1595–1600.

[72] J. Wang, L. Huang, G. Li, X. Wang, and T. Mak, "A fault-tolerant routing algorithm for noc using farthest reachable routers," in *Dependable, Autonomic and Secure Computing (DASC), 2013 IEEE 11th International Conference on*, Dec 2013, pp. 153–158.

[73] B. Fu, Y. Han, H. Li, and X. Li, "A new multiple-round dor routing for 2d network-on-chip meshes," in *Dependable Computing, 2009. PRDC '09. 15th IEEE Pacific Rim International Symposium on*, Nov 2009, pp. 276–281.

[74] C.-T. Ho and L. Stockmeyer, "A new approach to fault-tolerant wormhole routing for mesh-connected parallel computers," *Computers, IEEE Transactions on*, vol. 53, no. 4, pp. 427–438, April 2004.

[75] A. Kologeski, C. Concatto, F. Kastensmidt, and L. Carro, "Atards: An adaptive fault-tolerant strategy to cope with massive defects in network-on-chip interconnections," in *VLSI and System-on-Chip (VLSI-SoC), 2012 IEEE/IFIP 20th International Conference on*, Oct 2012, pp. 24–29.

[76] J. Duato, S. Yalamanchili, and N. Lionel, *Interconnection Networks: An Engineering Approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002.

[77] A. M. Shafiee, M. Montazeri, and M. Nikdast, "An innovational intermittent algorithm in networks-on-chip (noc)," *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 2, no. 9, pp. 2907 – 2909, 2008. [Online]. Available: http://waset.org/Publications?p=21

[78] Y. Miura, K. Shimozono, S. Watanabe, and K. Matoyama, "An Adaptive Routing of the 2-D Torus Network Based on Turn Model," in *Computing and Networking (CANDAR), 2013 First International Symposium on*, Dec 2013, pp. 587–591.

[79] S. Mubeen and S. Kumar, "Designing efficient source routing for mesh topology network on chip platforms," in *Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on*, Sept 2010, pp. 181–188.

[80] S. Moriam and G. P. Fettweis, "Reliability assessment of fault tolerant routing algorithms in networks-on-chip: An analytic approach," in *Proceedings of the Conference on Design, Automation & Test in Europe*, ser. DATE '17. 3001 Leuven, Belgium, Belgium: European Design and Automation Association, 2017, pp. 61–66. [Online]. Available: http://dl.acm.org/citation.cfm?id=3130379.3130394

[81] " Turn Model website ," http://turnmodel.pld.ttu.ee/, 2016.

[82] E. A. Rambo, C. Seitz, S. Saidi, and R. Ernst, "Designing networks-on-chip for high assurance real-time systems," in *2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC)*, Jan 2017, pp. 185–194.

[83] C. M. Fiduccia and R. M. Mattheyses, "A linear-time heuristic for improving network partitions," in *Proceedings of the 19th Design Automation Conference*, ser. DAC '82. Piscataway, NJ, USA: IEEE Press, 1982, pp. 175–181. [Online]. Available: http://dl.acm.org/citation.cfm?id=800263.809204

[84] T. Hollstein, S. Azad, T. Kogge, and B. Niazmand, "Mixed-criticality noc partitioning based on the nocdepend dependability technique," in *Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2015 10th International Symposium on*, June 2015, pp. 1–8.

[85] T. Lehtonen, P. Liljeberg, and J. Plosila, "Fault tolerance analysis of noc architectures," in *2007 IEEE International Symposium on Circuits and Systems*, May 2007, pp. 361–364.

[86] C.-L. Chou and R. Marculescu, "Farm: Fault-aware resource management in noc-based multiprocessor platforms," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, March 2011, pp. 1–6.

[87] B. N. K. Reddy, M. H. Vasantha, and Y. B. N. Kumar, "A gracefully degrading and energy-efficient fault tolerant noc using spare core," in *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, July 2016, pp. 146–151.

[88] S. Tosun, V. Ajabshir, O. Mercanoglu, and O. Ozturk, "Fault-tolerant topology generation method for application-specific network-on-chips,"

Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 34, no. 9, pp. 1495–1508, Sept 2015.

[89] M. Fattah, M. Palesi, P. Liljeberg, J. Plosila, and H. Tenhunen, "Shifa: System-level hierarchy in run-time fault-aware management of many-core systems," in *Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE*, June 2014, pp. 1–6.

[90] J. Silveira, M. Bodin, J. Ferreira, A. Cadore Pinheiro, T. Webber, and C. Marcon, "A fault prediction module for a fault tolerant noc operation," in *Quality Electronic Design (ISQED), 2015 16th International Symposium on*, March 2015, pp. 284–288.

[91] S. Murali and G. De Micheli, "Sunmap: a tool for automatic topology selection and generation for nocs," in *Design Automation Conference, 2004. Proceedings. 41st*, July 2004, pp. 914–919.

[92] S. Saeidi, A. Khademzadeh, and A. Mehran, "Smap: An intelligent mapping tool for network on chip," in *Signals, Circuits and Systems, 2007. ISSCS 2007. International Symposium on*, vol. 1, July 2007, pp. 1–4.

[93] J. Joven, O. Font-Bach, D. Castells-Rufas, R. Martinez, L. Teres, and J. Carrabina, "xenoc - an experimental network-on-chip environment for parallel distributed computing on noc-based mpsoc architectures," in *Parallel, Distributed and Network-Based Processing, 2008. PDP 2008. 16th Euromicro Conference on*, Feb 2008, pp. 141–148.

[94] "Atlas: A noc generation and evaluation framework," https://corfu.pucrs.br/redmine/projects/atlas.

[95] E. Carara, R. de Oliveira, N. Calazans, and F. Moraes, "Hemps - a framework for noc-based mpsoc generation," in *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*, May 2009, pp. 1345–1348.

[96] H. Ying, T. Hollstein, and K. Hofmann, "Gsnoc; the comprehensive design platform for 3-dimensional networks-on-chip based many core embedded systems," in *High Performance Computing and Simulation (HPCS), 2013 International Conference on*, July 2013, pp. 217–223.

[97] A. T. Tran and B. Baas, "Noctweak: a highly parameterizable simulator for early exploration of performance and energy of networks on-chip," 2012, http://www.ece.ucdavis.edu/vcl/pubs/2012.07.techreport.noctweak/.

[98] F. Singhoff, J. Legrand, L. Nana, and L. Marcé, "Cheddar: A flexible real time scheduling framework," in *Proceedings of the 2004 Annual ACM SIGAda International Conference on Ada: The Engineering of Correct and*

Reliable Software for Real-time &Amp; Distributed Systems Using Ada and Related Technologies, ser. SIGAda '04. New York, NY, USA: ACM, 2004, pp. 1–8. [Online]. Available: http://doi.acm.org/10.1145/1032297.1032298

[99] G.-M. Chiu, "The odd-even turn model for adaptive routing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 7, pp. 729–738, Jul 2000.

[100] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953. [Online]. Available: http://link.aip.org/link/?JCP/21/1087/1

[101] Y. Nourani and B. Andresen, "A comparison of simulated annealing cooling strategies," *Journal of Physics A: Mathematical and General*, vol. 31, no. 41, p. 8373, 1998.

[102] P. J. Van Laarhoven, E. H. Aarts, and J. K. Lenstra, "Job shop scheduling by simulated annealing," *Operations research*, vol. 40, no. 1, pp. 113–125, 1992.

[103] T. Hollstein, "Entwurf und interaktive hardware-/software-partitionierung komplexer heterogener systeme," Ph.D. dissertation, Darmstadt University of Technology, 2000.

[104] G. De Micheli and L. Benini, *Networks on Chips: Technology and Tools*, ser. Systems on Silicon. Elsevier Science, 2006. [Online]. Available: https://books.google.ee/books?id=IHHTmSBcoGIC

[105] F. P. Preparata, G. Metze, and R. T. Chien, "On the connection assignment problem of diagnosable systems," *Electronic Computers, IEEE Transactions on*, vol. EC-16, no. 6, pp. 848–854, Dec 1967.

[106] " Simpy library," http://simpy.readthedocs.io/.

ABSTRACT
Cross-Layer Dependability Management in Network on Chip based System on Chip

The shrinking technology node's feature size and growing number of transistors per chip, results in integration of more components on a single chip. Such massive integration of components can not be handled by the traditional and modern buses (resulting in the communication bottleneck). Networks-on-Chip have emerged as a scalable communication solution for handling the communication bottleneck.

At the same time, the shrinking technology feature size have resulted in increase in the effects of faults along with occurrence of new types of faults. Such increase in the number faults results in significant reduction in lifetime of the system.

The reduced lifetime of the systems and the communication paradigm shift towards Networks-on-Chip, motivates research on dependability of such on-chip networks. It is important to augment the system with mechanisms that prolong the lifetime of the system at the cost of graceful degradation of the system performance.

Providing support mechanisms for graceful degradation to the network require mechanisms that consider the global state of the system. These mechanisms enable system-wide decisions. However, using only global mechanisms imposes unacceptable latency for handling rather small events. In order to address this shortcoming, local hardware-based solutions are also required. Such mechanisms are inherently fast due to their proximity to the actual hardware and can act with no or minor support from global mechanisms.

This work focuses on a cross-layered approach which consists of three main layers; fault information acquisition, local and hybrid mechanisms and global mechanisms. In order to model the global effects of the faults, this work provides a new simulator for network-on-chip based systems-on-chip for modeling global fault management. A minimization framework for reducing area overhead of online fault detection mechanisms is presented. Later, a light-weight extension of the an already existing hardware-based fault classifier method is proposed. The small area overhead of the fault classifier enables massive deployment of such mechanisms in the hardware. Also a new metric is

proposed for classification of non-minimal path, uniform 2D Turn-Model based routing algorithms. The robustness of all such routing algorithms is evaluated under all possible link failure configurations in the network. A mechanism for reachability guarantee of the packets in the network under arbitrary link failure and any turn-model based routing algorithm (as long as the network is connected under the routing algorithm) is proposed. In order to reduce the possibility of network-wide failure due to faults in the data-path of routers, two packet dropping mechanisms have been proposed. And last but not least, this work provides an evaluation framework for router's reliability after deployment of all the dependability mechanisms. This is accompanied by architectures which provide local resource management and provide a boost in the router reliability.

The investigations in this work demonstrate the effectiveness of such cross-layer approaches for dependability in network-on-chip based system-on-chip. Details of required mechanisms along their advantages and drawbacks are discussed. It is shown how the area overhead of all the dependability mechanisms will influence the reliability of the system as a whole and proposes mechanisms for system's reliability improvement.

KOKKUVÕTE
Kiipvõrkudel põhinevate süsteemide kihtideülene usaldatavuse haldus

Aina miniatuursemaks muutuv tehnoloogia ja üha suurem arv transistoreid kiibis toob kaasa paljude komponentide ehk tuumade integreerimise ühele kiibile. Tuumade massilist integreerimist ei saa läbi viia traditsiooniliste ega kaasaegsete siiniarhitektuuride abil, sest see tooks endaga kaasa side kitsaskohti. Nimetatud kitsaskohtade lahendamiseks on välja töötatud kiipvõrgu lähenemine kui skaleeritav sidelahendus. Samal ajal on vähenenud tehnoloogia mõõtmete tõttu toimunud rikete arvu kasv kiipides ja samuti on esile kerkinud uued rikketüübid. See omakorda on viinud süsteemi eluea olulise vähenemiseni. Süsteemide lühenenud eluiga ühelt poolt ja kommunikatsiooni paradigma suundumus kiipvõrkude suunas teiselt poolt motiveerib uurima selliste kiipide töökindlust. Oluline on süsteemi täiustada mehhanismidega, mis pikendavad süsteemi eluiga süsteemi jõudluse vähenemise hinnaga.

Nimetatud toetusmehhanismid eeldavad struktuure, mis arvestavad süsteemi seisukorraga. Need struktuurid võimaldavad vastu võtta kogu süsteemi hõlmavaid otsuseid. Kuid probleemiks on siin, et vaid globaalsete struktuuride kasutamisel tekib lokaalsete sündmuste käsitlemise puhul vastuvõetamatu latentsus. Selle puuduse kõrvaldamiseks on vaja ka kohalikke riistvarapõhiseid lahendusi. Sellised mehhanismid on oma olemuselt kiired, kuna nad on lähedased tegelikule riistvarale ja võivad tegutseda globaalsete mehhanismide toetuseta.

Käesolev dissertatsioon keskendub kihiülesele lähenemisele, mis hõlmab kolme peamist kihti: rikketeabe omandamine, kohalikud ja hübriidsed mehhanismid ning globaalsed mehhanismid.

Rikete globaalse mõju modelleerimiseks pakub käesolev töö välja uue kiipvõrgu süsteemide simulaatori.

Lisaks esitatakse töös rikete tuvastamise mehhanismide minimeerimise keskkond. Seejärel pakutakse välja meetod rikete klassifitseerimiseks. Rikete klassifikaatori väike pindala võimaldab selliste mehhanismide ulatuslikku kasutuselevõttu riistvaras. Samuti on töös välja pakutud uus mõõdik mitte-minimaalsel teel põhinevatele, 2D marsruutimisalgoritmide klassifitseerimiseks.

Kõikide selliste marsruutimisalgoritmide jõudlust hinnatakse võrgu kõigi võimalike ühenduste rikete konfiguratsioonide all. Pakutakse välja võrgu kõigile ruutersõlmedele ligipääsu tagamise mehhanism meelevaldse võrguühenduse rikke korral.

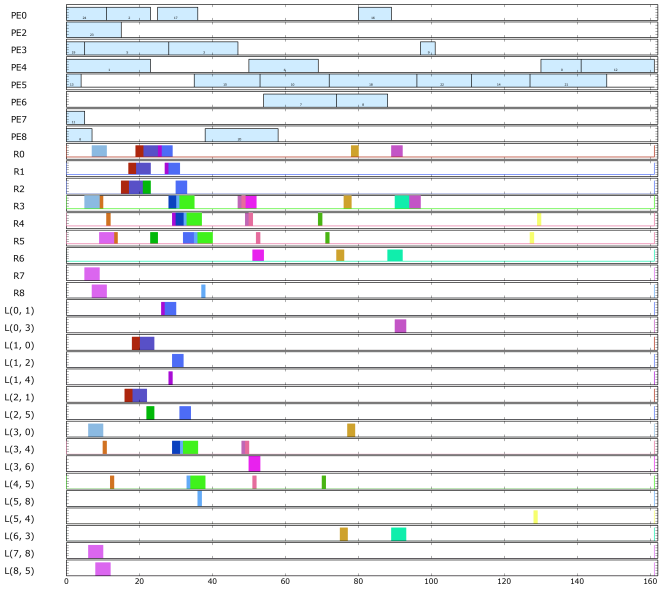
Selleks, et vähendada võimalust võrguruuterite andmevoo riketest tulenevaks kogu võrku hõlmavaks veaks pakutakse töös välja kaks pakettide kustutamise mehhanismi. Ning lisaks annab käesolev töö metodoloogia kiipvõrgu ruuteri usaldusväärsuse hindamiseks peale kõikide eelpool nimetatud töökindluse mehhanismide kasutuselevõttu. Sellega kaasnevad arhitektuurid, mis tagavad kohaliku ressursside haldamise ja võimaldavad suurendada ruuteri usaldusväärsust.

Käesolevas dissertatsioonis esitatud uurimistöö näitab kihtideülese lähenemisviisi kiipvõrkude usaldusväärsuse tagamisel. Käsitletud on usaldusväärsuseks vajalike mehhanismide üksikasju ning arutletud nende eeliste ja puuduste üle. Töös on näidatud, kuidas kõigi usaldusväärsusmehhanismide pindalanõudlus mõjutab kogu süsteemi usaldusväärsust tervikuna ja on välja pakutud mehhanisme kogu süsteemi töökindluse parandamiseks.

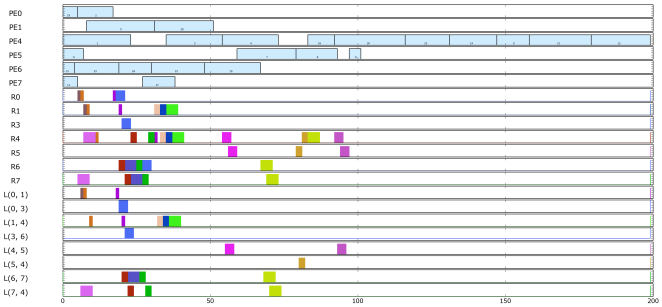
APPENDICIES

Appendix A: Scheduling Example

This appendix describe full scheduling reports for mapping of a 25-task task graph on a fault free 3×3 network and a heavily faulty 3×3 mesh network. [for more information regarding this example, please refer to section 4.5.6].



scheduling of the tasks on the fault free system



scheduling of the tasks on the system under 12 broken links

Appendix B: Publication A

T. Hollstein, S. P. Azad, T. Kogge, H. Ying and K. Hofmann, ”NoCDepend: A Flexible and Scalable Dependability Technique for 3D Networks-on-Chip,” 2015 IEEE 18th International Symposium on Design and Diagnostics of Electronic Circuits & Systems, Belgrade, Serbia, 2015.

NoCDepend: A flexible and scalable Dependability Technique for 3D Networks-on-Chip

Thomas Hollstein, Siavoosh Payandeh Azad, Thilo Kogge
Department of Computer Engineering
Tallinn University of Technology
Tallinn, Estonia

Haoyuan Ying, Klaus Hofmann
Integrated Electronic Systems Lab
Technische Universität Darmstadt
Darmstadt, Germany

***Abstract*—In order to be able to handle an arbitrary amount of static communication segment faults in NoC-based MPSoCs, a flexible fault tolerance mechanism has to be applied. In this contribution, we present a flexible and scalable approach for fault-tolerance in NoCs, which - in contrast to existing circumvention techniques - can in principle handle any number of static faults in the routing network. It doesn't require routing restrictions (as static routing/source routing) and can basically be combined with any static or adaptive minimal or non-minimal routing algorithm. The needed additional hardware effort is low and the increase of the time for computation of routing decisions is reasonably low as well. The presented dependability technique can work hand-in-hand with a task scheduler/mapper and is applicable in critical, mixed-critical and non-critical application scenarios.**

I. Introduction and Related Work

Sophisticated fault tolerance mechanisms are needed in order to be able to make use of any NoC-based MPSoC by tolerating any combination of faulty routing segments in the network. In the recent years, several NoC fault tolerance approaches have been presented. We can classify the main approaches as follows: **Local circumvention** ([1],[2],[3],[4],[5]): bypass the faulty node locally. These methods are able to handle a limited number of faulty routing segments, otherwise the reachability of the destination nodes is not guaranteed. Additional subclasses are Rectangular fault block models ([6], [7]) and Fault rings and chains ([8]). **Global path-based methods** ([9],[10]): search for possible paths from source to destination, store path information in the source node and apply non-adaptive source routing (doesn't scale well with the network size). **Intermediate node solutions** ([11]: Use intermediate routers legal turns to improve reachability. This means that the source node sends the packet to an intermediate node which will forward it to the destination node (might end up in higher-level deadlocks; additional HW effort in NoC interface). **Data splitting** ([12]): partially serialize the faulty links in order to maintain the network operation. However this method will drastically degrade the network performance (disturbs pipeline principle, causing congestion) and has considerable area overhead. **Hybrid methods:** In [13] a combined method is propose, but this still needs partially dynamic path search to guarantee the routability.

In this contribution a new hybrid fault tolerance method for NoCs is presented, which does not rely on virtual channels, supports any amount of faults in the network (as long as the network is not disconnected and there exists a path from source to destination under routing algorithm constraints), has very low latency during run-time, doesn't need path search packets and provides full support of any adaptive minimal or non-minimal routing algorithm (turn-model-based or planar-adaptive) with a very low hardware and run-time overhead.

This paper is organized as follows: In Section II the general NoCDepend methodology is introduced and an insight on the needed additional Hardware and Run-time effort is given. Subsequently a concrete demonstration and validation of the presented methods based on a well-approved NoC Simulator is shown, followed by some final conclusions.

II. NoCDepend Methodology

Assumptions: The proposed method is not dependent on a specific routing algorithm, and it is assumed, that the MPSoC has the following necessary components already built-in: *A Built-In-Self-Test (BIST)* for Mesh-based NoC communication resources, *A central System Health and Resource Manager (SHRM)*, running on-chip in a processing tile of the Multi-processor System-on-Chip (MPSoC). The SHRM is described in the following section. *iJTAG* [14] implementation for communicating test results to SHRM and later on distribution of information to nodes.

A. Fault Model and System Health Management

The fault-model of the presented method, considers inter-router communication link faults. However it can easily be enhanced to consider intra-router faults (turn faults) as well. System-Health-Map (SHM) (Fig. 1). The SHM is co-located

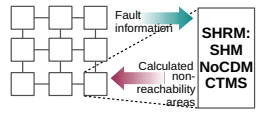


Fig. 1: System-Health and Task Deployment Management

with a Central Task Mapper and Scheduler (CTMS) on a tile in the network, which can re-locate application tasks, if the required communications cannot be carried out as a result of defected communication links between pairs of routers.

The NoCDepend manager NoCDM is calculating individual non-reachability information *for every individual NoC router output* and is sending them as configuration information back to every router, where they are evaluated by the locally running routing engines. **Every router output can store a limited number of $MaxNumAreas$ non-reachability zones/areas**

978-1-4799-6780-3/15 $31.00 © 2015 IEEE
DOI 10.1109/DDECS.2015.30

in the network, which cannot be reached via this output. The additional latency and hardware effort in the routers is kept low and the method guarantees, that all packets can be routed and any amount of defective links can be handled.

B. Basic Principle: Non-reachability

The NoCDepend method is designed to be applied to a mesh-based directed Graph $G_{Noc}(R, C, F_R)$. The graph should be a sub-graph of a N-Dimensional mesh, but its applicable to other topologies as well. In this paper a two-dimensional NoC is assumed. The set of nodes R represents the routers in the mesh. The edge set C represents outgoing router links $C_{x,y,dir}$. F denotes the applied routing function. An edge has the attributes $h_{x,y,dir}$ and $NR_{x,y,dir}$. The link health status $h_{x,y,dir}$ can either be 1 (healthy) or 0 (faulty) and $NR_{x,y,dir} = \{NR_{x,y,dir,1}, ...NR_{x,y,dir,MaxNumAreas}\}$ is a limited list of network areas, which is newly introduced here and defines, which target nodes in the NoC cannot be reached via this outgoing router link. It turned out that it is useful to assume an axis aligned cuboid as shape for the unreachable network areas, because they can be represented for every dimension with two points. In this paper the illustration is limited to rectangles as shape for unreachable areas in 2D-Nocs. Every axis-aligned N-dimensional cuboid can be represented by 2 coordinates min and max.

Example 1: Fig.(2) illustrates a fault scenario: Since the outgoing link $C_{2,2,E}$ from router $R_{2,2}$ to router $R_{3,2}$ is broken, $h_{2,2,E} = 0$ and $NR_{2,2,E,1}$ (r1) is the first rectangle in the non-reachability list of this output containing the information, which part of the network cannot be reached by this link. Assuming a non-minimal west-first routing scheme, this would be the region which can be represented by a rectangle spanned by the routers $R_{3,0}$ and $R_{5,5}$. Since the adaptivity of the routing algorithm allows easy circumvention toward eastern direction, only router $R_{2,2}$ needs to store this information in its eastward outgoing link and no propagation to neighbor routers is required.

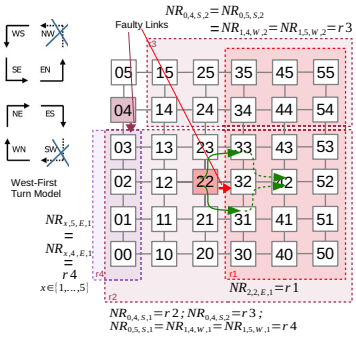


Fig. 2: Example: Faults and Faulty Area Propagation

If link $C_{0,4,S}$ is broken then the areas $r2$ and $r3$ cannot be reached any more via this link. Then the routers $R_{0,5}, R_{1,4}$ have to be informed by $R_{0,4}$, that the rectangular region $r4$ (cannot be reached via $C_{0,4,\{N,E,S\}}$) cannot be reached via $R_{0,4}$. Therefore $r4$ has to be entered in the non-reachability lists $NR_{0,5,S}$ and in $NR_{1,4,W}$. $R_{0,5}$ has to propagate $r4$ to $R_{1,5}$ where this information has to be attributed to the West output $NR_{1,5,W}$. Routers $R_{1,4}$ and $R_{1,5}$ have to propagate to eastern direction, that $r4$ cannot be reached, since the turn model doesn't allow any S-to-W turns. This information has to be propagated through the whole row 5 and row 4 in eastern direction, since all routers need to know the non-reachability of $r4$. This implies, that the routers $R_{x,5}$ and $R_{x,4}$ (with $x \in \{1, .., 5\}$) cannot reach $r4$. Tasks which have to communicate to $r4$ have to be re-mapped by the CTMS.

C. Algorithm for Fault Propagation

As shown in section II-B, a local inter-router propagation for unreacheable NoC zones is needed. Subsequently a generic algorithm is presented, which can be combined with any non-minimal or minimal routing algorithm. A turn-model assumption is used for the explanation, but the algorithm can also be applied separately to the defined routing layers in planar adaptive routing approaches. The *NoCDepend:FaultPropagation* algorithm (1) will be executed, once the CTMS receives BIST information about a new broken inter-router connection. Given a function $PrecInp(C_{x,y,dir})$, which returns a list of router inputs $I_{x,y,idir}$, which precede/can reach the output under application of the currently active routing algorithm F and a function $SuccOutp(I_{x,y,idir})$, which returns all possible outputs directions (full list independent of destination address) being reachable via the applied routing algorithm for any packet coming in at the router input $I_{x,y,idir}$, the algorithm for propagation of non-reachability information in general form is described in Algorithm 1.

In this algorithm $MaxNumAreas$ is the maximum number of non-reachable rectangles, which can be stored for an outgoing port of a router and being used for reachability check by the routing algorithm. If this number is exceeded, a lossy compression has to be carried out (lines 3 and 20 of Alg. 1). Once some additional fault information occurs on the output of one router, the algorithm checks for every input of this router, if parts of the newly introduced non-reachable areas can be reached via other outputs. Only the remaining newly unreachable areas will be propagated back to the output of the adjacent router (or the local port). The algorithm terminates, once no further propagation is needed ($ProcessingList$ empty). The communication to the local port is not shown in the algorithm, but based on the accumulated non-reachability information it is already known in the SHM if packet destinations (of tasks being mapped to this node) are in non-reachable areas. In this case this task must be re-mapped to another processing location.

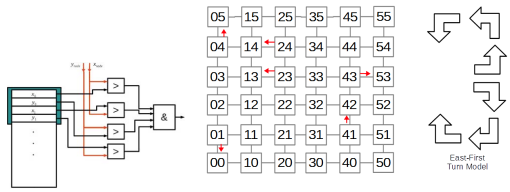
Lemma 1: If there exists no path from a specific source router S to a destination router D under the constraints of the currently used routing algorithm, then this information is available at the the source router and the NoCDM can directly relocate/remap the sending or receiving application task to another node.

Proof 1: A short proof outline on the completeness of the non-reachability information (no dead-end paths) can be done with mathematical induction as follows: 1) If a specific destination D of a routing path, outgoing from a source location S cannot be reached by the last router R on a minimal path via a broken link, then this non-reachability information has to be propagated to all inputs (and their predecessors = adjacent

Algorithm 1: Generic Non-Reachability Propagation in the SHRM

receive broken link information for $C_{x,y,dir}$ from BIST
compute non-reachable areas $NRnew_{x,y,dir}$ of output
$NRnew_{x,y,dir} :=$
$Minimize(NRnew_{x,y,dir}, MaxNumAreas)$
$NR_{x,y,dir} := NRnew_{x,y,dir}$ // new assignm. to $NR_{x,y,dir}$ since directly adjacent link is broken
ProcessingList := $C_{x,y,dir}$
while *ProcessingList not empty* **do**
  $C_{xf,yf,dirf}$:= GetAndRemoveHead(ProcessingList)
  forall the $I_i \in PrecInp(C_{xf,yf,dirf})$ **do**
    // potentially possible incoming directions - all other inputs don't need to be updated
    $NRprop := NRnew_{xf,yf,dirf}$
    // init rectangle list which has to be propagated to Predecessor of I_i
    for $O_j \in \{SuccOutp(I_i) \setminus Output(dir_f)\}$ **do**
      // for all outputs without the one where the new fault propagation came from
      forall the *non-reachable router destinations* $D(x_d, y_d)$ *in rectangles* $\in NRprop$ **do**
        if *destination routable via* O_j *according to routing algorithm* F **then**
          if *ReachabilitySearch*(R_{x_d,y_d}, O_j) *successful* **then**
            $NRprop := NRprop \setminus R_{x_d,y_d}$
            // R_{x_d,y_d} is reachable from I_i via other output port O_j and has to be removed from the list
  $NRnew_{x_p,y_p,dir_p} := NRprop$
  $NRreallynew :=$
  $NRnew_{x_p,y_p,dir_p} \setminus (NRnew_{x_p,y_p,dir_p} \cap NR_{x_p,y_p,dir_p})$
  // only non-reachable areas which were not yet contained in the output's non-reachable areas list
  if $NRreallynew \neq \emptyset$ **then**
    $NR_{x_p,y_p,dir_p} := NR_{x_p,y_p,dir_p} \cup NRreallynew$
    $NRnew_{x_p,y_p,dir_p} := NRreallynew$
    $NRcompressed :=$
    $Minimize(NR_{x_p,y_p,dir_p}, MaxNumAreas)$
    $NRadditionally := NRcompressed \setminus NR_{x_p,y_p,dir_p})$
    $NR_{x_p,y_p,dir_p} := NR_{x_p,y_p,dir_p} \cup NRadditionally$
    $NRnew_{x_p,y_p,dir_p} :=$
    $NRnew_{x_p,y_p,dir_p} \cup NRadditionally$
    ProcessingList = ProcessingList $\cup$ C_{x_p,y_p,dir_p}
  else
    {nothing has to be done since no new information}

routers), which cannot reach D on a non-minimal path via another valid output of the same router R, applying allowed turns only according to the currently valid turn model (initial condition).
2) If a router R on any location of any possible path from S to D knows that D cannot be reached via one of its outputs, this information has to be propagated to all inputs of R(and their predecessors = adjacent routers), which cannot reach D on a non-minimal path via another valid output of the same router R, applying allowed turns only according to the actually valid turn model (conclusion from n to n+1). This leads to the following conclusion: If there is no routing path to a destination D, then this information is known on any point of all possible paths at any router position in the NoC. If this information reaches S, then there is no working path from S to D and the initiating the communication has to be placed on another network location.



(a) **(b)**

Fig. 3 (a) Proposed detector circuit, (b) Example fault scenario

Furthermore the deadlock-freeness of the proposed method has to be confirmed:
Lemma 2: The NoCDepend method returns deadlock-free routing schemes, if the underlying basic routing algorithm is deadlock-free.
Proof 2: The used routing algorithm can be represented as a directed routing graph G_R for the concrete non-defect 2D/3D NoC If the underlying routing algorithm is deadlock-free then G_R is deadlock-free, since cycle-free. If some edges (defect links) are removed from G_R, resulting in G_R^*, then G_R^* is still cycle-free and therefore the resulting defective scheme is deadlock-free as well (partial edge usage due to unreachable areas cannot cause cycles).
The deadlock-freeness has been verified applying a self-developed graph-based prooving engine.

D. Additional Hardware and Routing-Time Complexity

Assuming, the iJTAG standard and/or the NoC itself is used for communication between SHRM and routers, no significant additional effort is needed for communication of link fault information and non-reachability information to be stored in router outputs. In the 2D-case non-reachable areas are represented as rectangles r_{dir_i} with Cartesian coordinates of two corners as a tuple $(x_{lowerleft}, y_{lowerleft}, x_{upperright}, y_{upperright})$. This lists should have limited length and be rather short in order to keep the overhead in terms of area and static power consumption quite limited.
It is fairly easy to check if a node falls inside a rectangle just by use of simple comparators. The detector circuit can be implemented either fully serial or parallel. The schematics for the detector circuit for one list entry is shown in Figure 3a. This circuit must be small, since it is needed in all routers (alternative: access arbitration).

III. Results

In this project, the GSNoC simulator [15] in combination with the XHiNoC [16] architecture and an implementation of a minimized East-First deadlock-free routing algorithm has been used for validation of the NoCDepend method. A link fault model has been implemented and a set of registers for non-reachable areas has been integrated to every output. A detector is identifying non-routable packets (which should not arrive - dead end situation).
Algorithmic Results: Figure 3b shows the configuration of faulty links in a 6×6 mesh network. In this case it is assumed

Node	East-List	West-List	South-List	North-List
00	[45,42]	–	–	[05,05]
10	[45,42]	[05,05]	–	–
20,30	[45,42]	–	–	–
40	–	–	–	[45,42]
50	–	[45,42]	–	–
01	[45,42]	–	[00,50]	[05,05]
11	[45,42]	[05,05], [00,00]	–	–
21,31	[45,42]	–	–	–
41	–	–	–	[05,52]
51	–	[45,42]	–	
02	–	–	[00,00]	–
12	–	[00,00], [05,05]	–	
22	–	–	–	[04,13]
32,42,52	–	–	–	[03,13]
03	[55,50]	–	[00,00]	[05,05]
13	[55,50]	[05,05], [00,00]	–	–
23	[55,50]	[05,10]	–	[04,14]
33,43	[55,50]	[03,13]	–	[04,14]
53	–	[03,13]	–	[04,14]
04	–	–	–	[05,55]
14	–	[05,05], [00,00]	–	–
24	–	[05,10]	[03,13]	–
34,44,54	–	[04,14]	[03,13]	–
05	–	–	[00,00]	–
15	–	[00,00]	–	–
25,35,45,55	–	–	[04,14]	–

TABLE I Unreachable rectangles for the example of Fig. 3b

that two rectangle entries exist per outgoing link in each router (according to cost limits). Red arrows represent faulty links. Example: the outgoing link of node 24 to the west is broken. After the propagation phase (Alg. 1, called once for every detection of a faulty link), a list of non-reachable areas for each concerned router output is calculated in the central SHRM (Table I). These rectangles are sent to the real NoC's specific router output ports (in the NoC simulator). A series of simulations of random based Generic Scalable Pseudo Applications (GSPA) [17] was performed using the GSNoC simulator. The resulting non-reachability information is shown in table I.

Correctness Analysis: Exhaustive simulations have been carried out on 2D 6x6 and 10x10 NoCs, applying XY routing (deterministic, minimal routing) and West-First routing (adaptive, non-minimal routing) schemes. The simulations have been carried out on the basis of a large amount of a) random source, random destination and b) all-sources to all-destinations packet injections. If the non-reachability information for a specific destination has been already available at the injection point, then the injection has been prevented (in a real application this would be a case for task-remapping in the SHRM). Exhaustive Simulation results validate, that in cooperation with a task/mapper scheduler 100% reachability can be guaranteed (as long as there exists a path from source to destination under routing algorithm constraints or prior detection of the need of re-mapping). The higher the degree of adaptivity of the routing, the less often re-mapping is required. Applying the non-minimal West-First routing algorithm, and increase of the simulation time and average packet latency can be observed for deterministic all-to-all traffic patterns

Analysis of Implementation Overhead: In order to examine the direct implementation overhead of the non-reacheability rectangle storage and evaluation logic we have implemented the fast parallel evaluation method in VHDL and synthesized it on a 65nm ASIC library using SYNOPSYS. Table II shows the resulting overheads storing and evaluating two non-reacheable rectangles in every router output link: 4% in area, 3% in static power and 5% in dynamic power. Since the evaluation of rectangle matches can be done in parallel to logic computation in the routing algorithm no additional delay is generated here (can vary depending on the length of critical path within the routing logic).

	Orig. Router	with NoCDepend	overhead
area (μm^2)	157662	165149	4%
Dynamic power (mW)	114.3	118.5	3%
Static power (μW)	725.3	762.8	5%

TABLE II Area and power overhead in 65nm technology

IV. Conclusions

In this paper, we presented a scalable and functionally safe dependability technique for NoCs, which can tolerate any number of faulty links. The method is resource-efficient and universal and can be combined with any routing algorithm in 2D and 3D. The implementation results show a significant improvement in reachability: if the destination is reachable under routing algorithm constraints, NoCDepend will enable this. In combination with a central task mapper/scheduler 100% reachability can be achieved.

References

[1] M. Imai and T. Yoneda, "Improving dependability and performance of fully asynchronous on-chip networks," in *Asynchronous Circuits and Systems (ASYNC), 2011 17th IEEE International Symposium on*, April 2011, pp. 65–76.

[2] H. Kia and C. Ababei, "A new fault-tolerant and congestion-aware adaptive routing algorithm for regular networks-on-chip," in *Evolutionary Computation (CEC), 2011 IEEE Congress on*, June 2011, pp. 2465–2472.

[3] M. Ebrahimi, M. Daneshtalab, and J. Plosila, "High performance fault-tolerant routing algorithm for noc-based many-core systems," in *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*, Feb 2013, pp. 462–469.

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

[5] S. Zhang, G. Han, and F. Zhang, "Very fine-grained fault-tolerant routing algorithm of noc based on buffer reuse," in *Software Engineering and Service Science (ICSESS), 2013 4th IEEE International Conference on*, May 2013, pp. 758–762.

[6] Z. Jiang, J. Wu, and D. Wang, "A new fault-information model for adaptive and minimal routing in 3-d meshes," *Reliability, IEEE Transactions on*, vol. 57, no. 1, pp. 149–162, March 2008.

[7] C.-C. Su and K. Shin, "Adaptive fault-tolerant deadlock-free routing in meshes and hypercubes," *Computers, IEEE Transactions on*, vol. 45, no. 6, pp. 666–683, Jun 1996.

[8] K. hsuan Chen and G. ming Chiu, "Fault-tolerant routing algorithm for meshes without using virtual channels," Journal of Information Science and Engineering, Tech. Rep., 1997.

[9] E. Wachter, A. Erichsen, A. Amory, and F. Moraes, "Topology-agnostic fault-tolerant noc routing method," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, March 2013, pp. 1595–1600.

[10] J. Wang, L. Huang, G. Li, X. Wang, and T. Mak, "A fault-tolerant routing algorithm for noc using farthest reachable routers," in *Dependable, Autonomic and Secure Computing (DASC), 2013 IEEE 11th International Conference on*, Dec 2013, pp. 153–158.

[11] B. Fu, Y. Han, H. Li, and X. Li, "A new multiple-round dor routing for 2d network-on-chip meshes," in *Dependable Computing, 2009. PRDC '09. 15th IEEE Pacific Rim International Symposium on*, Nov 2009, pp. 276–281.

[12] A. Kologeski, C. Concatto, F. Kastensmidt, and L. Carro, "Atards: An adaptive fault-tolerant strategy to cope with massive defects in network-on-chip interconnections," in *VLSI and System-on-Chip (VLSI-SoC), 2012 IEEE/IFIP 20th International Conference on*, Oct 2012, pp. 24–29.

[13] D. Wang, "A rectilinear-monotone polygonal fault block model for fault-tolerant minimal routing in mesh," *Computers, IEEE Transactions on*, vol. 52, no. 3, pp. 310–320, March 2003.

[14] "Ieee draft standard for access and control of instrumentation embedded within a semiconductor device," *IEEE P1687/D1.71, March 2014*, pp. 1–347, Sept 2014.

[15] H. Ying, T. Hollstein, and K. Hofmann, "Gsnoc - the comprehensive design platform for 3-dimensional networks-on-chip based many core embedded systems," in *High Performance Computing and Simulation (HPCS), 2013 International Conference on*, July 2013, pp. 217–223.

[16] F. Samman, T. Hollstein, and M. Glesner, "Adaptive and deadlock-free tree-based multicast routing for networks-on-chip," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 18, no. 7, pp. 1067–1080, July 2010.

[17] H. Ying, A. Jaiswal, T. Hollstein, and K. Hofmann, "Deadlock-free generic routing algorithms for 3-dimensional networks-on-chip with reduced vertical link density topologies," *Journal of Systems Architecture*, vol. 59, no. 7, pp. 528 – 542, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1383762113000295

Appendix C: Publication B

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,” 2016 11th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), Tallinn, Estonia, 2016.

SoCDep^2: A framework for dependable task deployment on many-core systems under mixed-criticality constraints

Siavoosh Payandeh Azad *, Behrad Niazmand*, Peeter Ellervee*,
Jaan Raik*, Gert Jervan*, Thomas Hollstein*†
Department of Computer Engineering
*Tallinn University of Technology, Estonia
† Frankfurt University of Applied Sciences
email: {siavoosh, bniazmand, LRV, jaan.raik, gert.jervan, thomas}@ati.ttu.ee

***Abstract*—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 is introduced. This 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 typologies, any turn model based routing algorithm, fault monitoring mechanisms and different fault models (Link, Turn, Node).**

***Keywords*—Dependability, Scheduling, Mapping, Many-Core.**

I. Introduction

The ever-increasing trend in using many-cores platforms in safety critical systems has made their dependability aspects more significant. The goal of this work is to introduce a framework for design space exploration of dependable (mixed- or non-critical) task deployment on Network-on-Chip (NoC) based many-core systems. A dependable system requires support for fault detection, maintaining a holistic view of the system's health status and a separate unit in charge of re-mapping (fault isolation and re-configuration). The proposed framework is called SoC Dependable application Deployment (SoCDep^2), which addresses all the above mentioned requirements. The architecture of the framework is demonstrated in Fig. 1. The system is composed of the following components:

- **Target Architecture:** which is assumed to be a 2D or 3D Network-on-Chip based many-core system.
- **System Health Map (SHM):** a shared memory that contains diagnostic information.
- **System Health Monitoring Unit (SHMU)**: gathers fault information from different components and updates SHM.
- **Mapper-Scheduler Unit (MSU)**: maps and schedules the application tasks to target architecture, based on fault configuration of the system.

To simulate such infrastructure, a Graphical User Interface (GUI) based open-source tool is written in Python language that encapsulates SHMU, SHM and MSU, and provides dependable mapping and scheduling solutions under mixed-criticality constraints.

978-1-5090-2520-6/16/$31.00 ©2016 IEEE

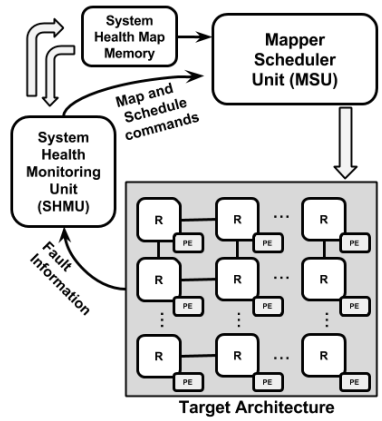


Fig. 1: Proposed SoCDep^2 framework architecture

The rest of this paper is organized as follows: Section II describes the related works and compares them to the proposed approach. Sections III to IX will discuss details of different parts of the proposed framework. Section X describes some experimental results obtained using the tool and Section XI concludes our work and describes our plans for future work.

II. Related Work

This section is dedicated to the review of the state-of-the-art in existing tools and frameworks which support mapping of applications onto NoC-based architectures, addressing mixed-criticality and fault-tolerance aspects and performing optimization algorithms, with brief explanation of their specific features. However, the focus of this work is only on NoC-based architectures, thus, works that exclusively focus on bus-based many-core systems are not in the scope of this paper. Finally, a comparison of the proposed framework and other existing frameworks is provided.

TABLE I: Comparison of the available tools in terms of different criteria

Tool Name	Cost Function	Mapping Algorithm(s)	Routing Algorithm(s)	GUI	Open Source	Programming Language	Scheduling Support	Mixed-Criticality Support	Fault Tolerance Support
SUNMAP [1]	average communication delay, area, power	Minimum-path mapping (based on NMAP)	Dimension ordered, minimum path, traffic splitting across all paths	+	–	C++/ SystemC	Not mentioned	–	–
SMAP [2]	Performance	Spiral-based, GA, random	XY, rest not mentioned	+	–	MATLAB	+	–	–
xENoC [3]	Performance	Based on SUNMAP	XY, Diagonal XY	–	–	SystemC	Not mentioned	–	–
Atlas [4]	Performance, Power	Static and dynamic	XY, turn model (West-First) routing	+	+	Java/ SystemC	+	–	Limited to HERMES
HeMPS [5]	Performance	Static and dynamic	XY	+	+	C++/SystemC	+	–	–
GSNoC [6]	Performance, Energy Consumption	GA, SA	Minimal and non-minimal routing, adaptive routing, turn model routing	+	–	C++/SystemC	+	–	–
NoCTweak [7]	Performance, communication energy	Random, NMAP	Minimal routing, deterministic and adaptive routing, turn model routing	–	+	C++/SystemC	+	–	–
Project Cheddar [8]	Task execution time	Not mentioned	Not mentioned	+	+	Ada	+	–	–
SoCDep^2 [proposed method]	Scheduling Makespan	LS, ILS, SA, Min-Min, Max-Min, MCT, MET, NMAP	Minimal and non-minimal turn model based, Adaptive and Deterministic	+	+	Python	+	+	+

LS:Local Search, ILS:Iterative Local Search, SA: Simulated Annealing, GA:Genetic Algorithm, MCT:Minimum Completion Time, MET:Minimum Execution Time, GUI:Graphical User Interface

Different frameworks for mapping and scheduling of applications on NoC based systems are proposed in the literature, covering different mapping algorithms from Meta-heuristics such as GA and SA [2, 6] and heuristics [1, 3, 7] to different static and dynamic mappings [4, 5] and random mappings [2, 7] using different cost functions such as performance [2–7], average communication [1] and task execution time [8]. These frameworks, utilize adaptive (minimal and non-minimal) [1, 4, 6, 7] and deterministic routing algorithms [2–5] and are written in different programming languages. In addition, some of these frameworks are open-source projects, such as [4, 5, 7, 8]. However, to the best of our knowledge, none of the above mentioned frameworks have support for mixed critical applications and only one has limited support for fault tolerance. An open-source framework for mapping and scheduling of applications on a NoC based system that provides the possibility of design space exploration in different levels, while supporting mixed-critical applications along with fault tolerance mechanisms, is the main motivation of this work.

Table I summarizes the frameworks explained above. In addition to the state-of-the-art, we have also listed our proposal, "SoCDep^2 ". As it can be seen in Table I, "SoCDep^2" includes most of the features provided by the other tools and the noticeable advantages are that (1) it addresses fault tolerance aspects of NoC based many-core systems, such as injection of faults, fault monitoring, testing and classifying faults under mixed-criticality constraints and (2) it is open-source, making it possible for different research groups to reproduce the experiments and contribute as well. In the following sections, a detailed description of the mentioned features and other modules of the proposed framework are provided.

III. Application Modeling

In "SoCDep^2", the application is modeled as a Task Graph (TG). A task graph is a directed acyclic graph TG(V,E) where V is the set of tasks and E is a set of edges between tasks. An edge between two tasks describes their data dependency. The proposed framework provides the user with the possibility to specify TG as manual, random dependent or independent or a benchmark application.

Each task (t_i) in TG has the following attributes:

- ID number of the task in TG
- Task's Worst Case Execution Time (WCET)
- Task's deadline
- Criticality Level: tasks in mixed-critical systems can be divided into two major classes, High critical tasks (H) which the system can not afford missing their deadlines and Low critical (L) tasks which the system can tolerate their deadlines' miss and continue to work with degraded performance. However, two more classes (gateways) are added which are used for network partitioning (for more information please refer to [9]). All the communication between the tasks, i.e. the edges, are assumed to fall either under High critical or Low critical domain.
- ID of the cluster that the task is assigned to
- ID of the node in Architecture Graph (AG, which is the NoC) that the task is mapped onto
- Release time: Time which the task becomes available. Release time is not necessary for all the applications. In case of having task dependency, release times can be set to zero.
- Task type: the type of task, which can be either Application or Test.

Each edge e_{ij} represents communication between task t_i and

t_j. Each edge e_{ij} in the TG, has the following attributes:

- Criticality level of the communication between two tasks.
- ID of the link from the Architecture Graph (AG, which is the NoC) that the edge is mapped onto.
- The communication weight between two tasks, measured in terms of the number of flits.

In addition, "SoCDep^2 " provides possibility to insert different slack(s) for high-critical tasks' re-execution and packet retransmission.

IV. Architecture Modeling

In the proposed framework, the target architecture is modeled as an Architecture Graph (AG). An Architecture Graph is a directed graph AG(V, E) where V is the set of nodes consisting of a router and a Processing Element (PE), and E represents the set of links between different nodes. For each node the following attributes are associated:

- A list of all tasks mapped onto the node
- The allocated time-slots for tasks which are mapped on the node (showing the sequence of their execution on the AG).
- A list of unreachable areas for the node (For more information please refer to [10]).
- Node region, used for network partitioning. Each node can be in critical or non-critical region or it be a gateway itself (see [9] for detailed information).

Similarly, the links keep the information about mapped packets (TG edges) and their allocated time-slots. The organization of the nodes in a Network-on-Chip is dictated by its topology. "SoCDep^2 " supports both 2D and 3D topologies, including 2D Mesh, 2D Torus, Line, Ring and 3D Mesh.

V. System Health Monitoring Unit

System Health Monitoring Unit (SHMU) is a unit that gathers fault information from the NoC based architecture (by means of either online fault detection or testing mechanisms) and keeps a general view of the system health status (faulty/non-faulty or performance degradation). The used fault model covers permanent and transient faults on links, router turns and Processing Elements (PEs) and aging (only for PEs). This status information is stored in a shared memory called System Health Memory (SHM). For detailed information regarding SHMU please refer to [11].

VI. Routing

In order to route packets to their destinations, "SoCDep^2" utilizes a Routing Graph (RG) [12]. This enables the tool to use conventional graph algorithms for such purposes. A Routing Graph RG(V,E) is a directed acyclic graph where V is the set of all the ports in the network (input and output ports of the routers, including the local ports) and E is the set of all connections between the ports.

The connections in the set E in RG can be either (a) from/to the local port of a router to/from input/output ports of the same router, (b) straight connections between opposite direction ports (such as from North to South or from West to East), (c) turns which connect an input port node to an adjacent output port of the same router (defined as 90-degree turns, for instance from North input to East output, denoted as N2E) or (d) connections from output ports of one router to input ports of another router (representing physical links in the network). By setting these connections, it is possible to model any turn model based routing algorithm under a specific fault configuration. It is also possible to feed in user defined turn models and set up non-uniform routing algorithms such as odd-even routing [13].

Using Routing Graphs, NoCDepend method [10] is implemented in "SoCDep^2" for reachability calculations and network partitioning [9] configuration.

VII. Task Clustering

"SoCDep^2" provides the user with the possibility to cluster tasks together and perform clustering optimization using greedy heuristics in order to reduce the search space for the mapping algorithm. In case of having mixed-crticial applications, clustering might result in a poor quality solution and the user can skip the clustering step, however, for either hard or soft real time applications, clustering can be very useful. These clusters will later be mapped onto the Processing Elements (PEs). In general, we can describe a Clustered Task Graph as a directed graph CTG(V,E) where V is a set of clusters in which each cluster contains a set of tasks, and E is the set of edges between two clusters, which encapsulates all the communications between tasks in those clusters. The communication weight of an edge in CTG, is the sum of all the communication weights of the encapsulated TG edges. The following optimization objectives are available to the user:

- $C_1 = \sigma(Com_E) + \sigma(Util_C)$
- $C_2 = max(Com_E) + max(Util_C)$
- $C_3 = C_1 + C_2$
- $C_4 = \frac{1}{N}\sum_{i=1}^{N} Util_{C_i}$
- $C_5 = max(Com_E)$

Where Com_E represents the set of communication weight of all the edges in CTG and $Util_C$ is the set of utilization of clusters. The term $Util_{C_i}$ represents the utilization of cluster $i \in [0, N]$ where N is the number of clusters.

VIII. Mapping and Scheduling

The task of mapping can be described as assigning a PE from AG to each cluster (and the tasks in each cluster) from CTG and assigning a path in the network to each edge in the Task Graph (TG). This process is followed by As Soon As Possible (ASAP) scheduling of the tasks on PEs and packets on links. To assign paths in the network to TG edges, RG is used which returns result of a deterministic/adaptive, minimal/non-minimal routing from the source to destination using conventional graph algorithms. However this work only focuses on deterministic routing algorithms, since adaptive routing would result in probabilistic scheduling of the packets (which should eventually be reduced to a deterministic case of it to be simulated). For dependent TGs, different mapping algorithms have been implemented in the tool:

- **Greedy Heuristics:** are mostly LS and ILS. The main idea for using such algorithms is to find a feasible solution with acceptable quality based on the current system health, every time a system failure occurs.
- **Meta-Heuristics:** The meta-heuristics are not going to be used to calculate a mapping solution, but are used as a benchmark for comparing greedy heuristic solutions. At the moment the only fully implemented meta-heuristic algorithm is SA. Different cooling schedules and cost functions have been implemented for SA.

Different cost functions are available to the user:

- $C_1 = \sigma(Makespan_L)$
- $C_2 = \sigma(Makespan_N)$
- $C_3 = C_1 + C_2$
- $C_4 = Max(Makespan_L) + Max(Makespan_N)$
- $C_5 = C_3 + C_4$
- $C_6 = \sigma(Util_N)$
- $C_7 = \sigma(Util_L)$
- $C_8 = \sigma(Util_N) + \sigma(Util_L)$

where $Makespan_L$ denotes the list of scheduling lengths for all the links, $Makespan_N$ denotes the list of scheduling lengths for all the nodes, $Util_N$ is a list of utilization of each node in the network and $Util_L$ is a list of utilization of each link in the network. The cost function severely penalizes the mapping/scheduling solutions that violate critical tasks' deadline. However, these solutions are not totally discarded.

For independent TGs, the following heuristics for mapping are also available:

- **Min-Min:** sorts the tasks based on Worst Case Execution Time (WCET) and maps the shortest task on the node with the smallest completion time.
- **Max-Min:** sorts the tasks based on WCET and maps the longest task on the node with the smallest completion time.
- **MET:** chooses tasks randomly from the TG and maps them on the fastest node.
- **MCT:** chooses tasks randomly from the TG and maps them onto the node with the smallest completion time.

In the end, the tool automatically generates Gantt charts of the final scheduling of the tasks.

IX. GUI, Documentation and Maintenance

In addition to the visualizations generated by "SoCDep^2", it includes a GUI, written in Python using Tkinter library, which makes it simple for the user to set different parameters in the tool, instead of changing them manually in the CONFIG file. The project is available under GNU-GPL2 license in GitHub [14]. All the documentation is available in project wiki.

X. Experimental results

In this section, some examples are discussed which go through all the steps of the process of mapping an application onto the architecture. Also, an example of vertical link placement process is provided. All these experiments are reproducible by running the script provided by the authors at [15]. In the end of this chapter, some performance metrics of the tool will be discussed.

A. Example Scenario

In this section we use a simple scenario and will use the visualizations generated automatically by the tool to display the process. Figure 2a shows the randomly generated TG for our experiment which contains 25 tasks and 35 edges. The AG for this example is shown in Fig. 2b which is a 3×3 2D Mesh topology (all the links shown in the figure are bi-directional). For the sake of simplicity we have not used a mixed criticality example and any partitioning in the network. Using initial faulty turns in the routers, SHM implements the deterministic XY routing algorithm. The SHM and Routing Graph (RG) representation are visualized in Fig. 2c and Fig. 2d. Considering the fact that each of the tasks of the TG can be mapped onto any Processing Element (PE), it gives us the problem size of $7,17 \times 10^{23}$. To reduce the size of the problem, the tasks are grouped in 9 clusters (same as the number of nodes in the network) which reduces the problem size to $3,87 \times 10^8$. In order to find a good quality clustering solution, a simple LS is applied with the sum of the maximum communication weight between clusters and maximum utilization (duration of the tasks in each cluster) on cluster nodes (see C_2 in section VII) as cost function. The process of optimization is shown in Fig. 2g which reduces the cost from 77 to 47 in 70 steps. The final Clustered Task Graph (CTG) (see Fig 2f) will be mapped randomly on the Architecture Graph (AG) under the constraints of Routing Graph (RG). The result of this initial mapping is shown in Fig. 2h.

Next, mapping optimizations are performed to distribute the tasks equally between the Processing Elements and also to distribute the communication weight evenly between the links (i.e. reducing standard deviation of the processing elements and links utilization, see C_8 from section VIII). Fig 3 shows different mapping heuristics using the same cost function starting from the initial mapping of Fig. 2h with cost of 32.24 and ending up with different mappings having different costs, The ILS optimization usually has more chance of hitting a good quality solution (with cost of 11.48) due to its wider search range, however, LS can also give a good quality solution (with cost of 11.41) but the solution's quality is rather based on luck. Simulated Annealing (SA) optimization has been used for benchmarking our heuristics using different annealing schedules. The tool provides the user with seven different annealing schedules for SA optimization, however, only three of these schedules are discussed here due to lack of space. Fig. 3e shows the cost development of SA using exponential annealing schedule (with an $\alpha = 0.9995$ which is relatively fast using only 6000 steps and converging to a local minimum with cost of 11.43). And Fig. 3g shows the cost reduction of the mapping using SA with an adaptive annealing mechanism designed by the authors (which took more than 65k steps, converging to a solution with cost of 10.87). The last annealing schedule (see Fig. 3i) to be discussed here is Huang's schedule (which took around 90k steps, converging to a solution with cost of 10.87). The problem size is relatively small which resulted in short convergence in heuristics (this problem size was chosen for demonstration purposes, bigger scenarios can be tested

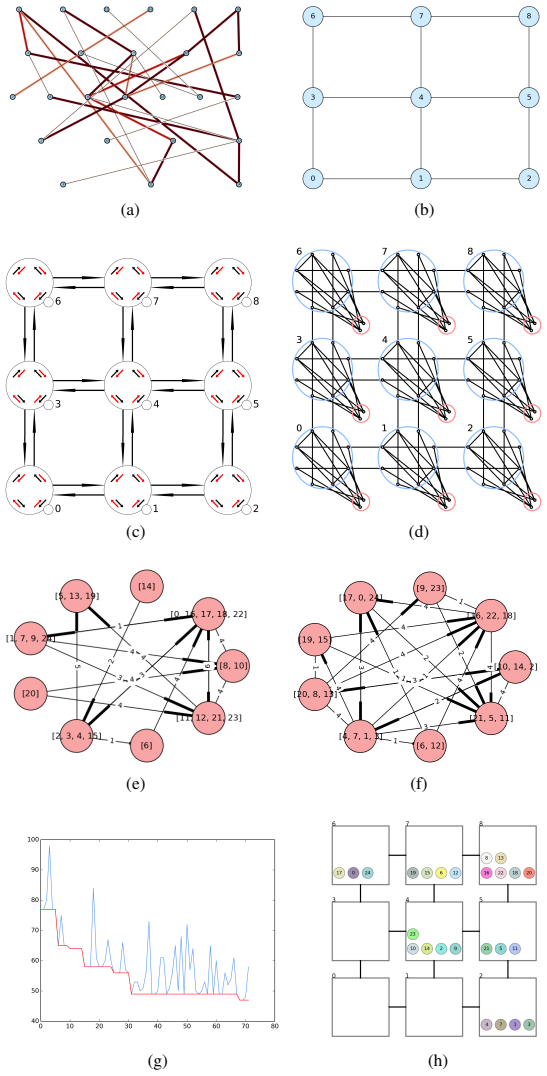


Fig. 2: a) Random dependent Task Graph with 25 tasks and 35 edges b) Architecture Graph modeling a 3×3 Mesh NoC c) SHM visualization: Using initial faulty turns, XY routing has been implemented d) Routing Graph e) Initial random clustering f) Clustered task graph after ILS optimization g) CTG cost during ILS optimization h) Initial random mapping of CTG on AG

using the tool). By using these heuristics (LS and ILS), a new mapping is calculated (based on SHM configuration) after each fault occurrence. This provides a mechanism for graceful degradation of resource utilization which will continue until total system failure. This scenario can be simulated using [14].

B. Tool Performance

To show the performance of the tool, two experiments are carried out using a randomly generated TG, where the

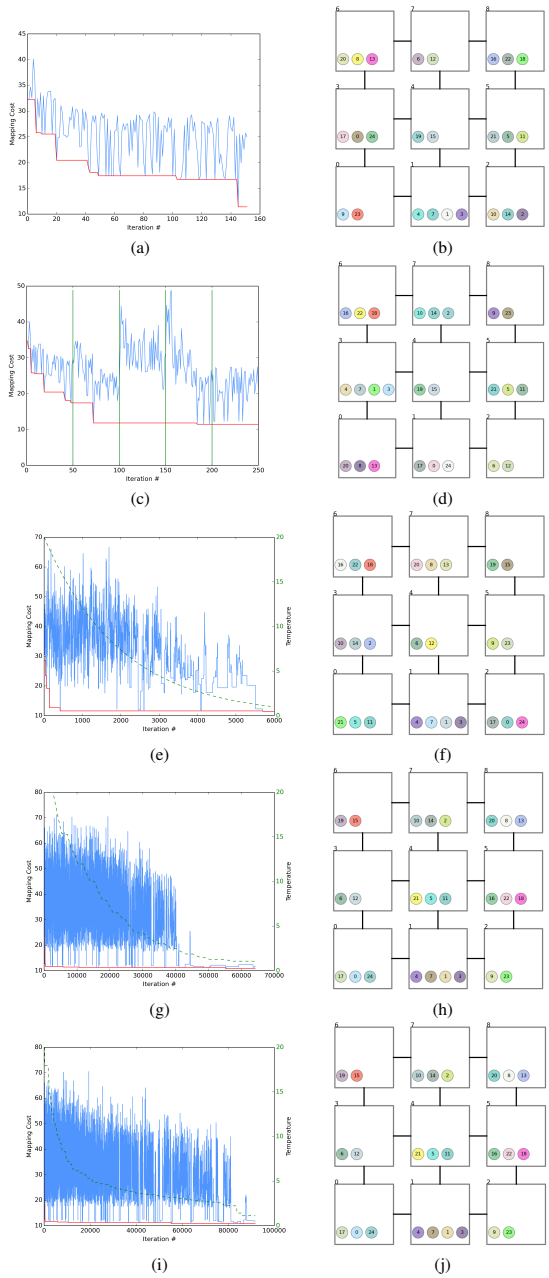


Fig. 3: a) Cost reduction for LS, b) Final Mapping (LS) c) Cost reduction for ILS d) Final Mapping (ILS) e) Cost reduction for SA (Exponential annealing with $\alpha = 0.9995$) f) Final Mapping (SA) g) SA (Custom Made adaptive annealing method) h) Final mapping using SA with costume made annealing i) SA using Huang's annealing schedule j) Final mapping using Huang annealing schedule

TABLE II: Tool performance with growing size of TG

TG Size	AG Size	O.P.S.	CTG Size	CTG Opt. Time (sec)	R.P.S.	Map Opt. Time (sec)
10	3×3	$3.4e9$	9	1	$3.8e8$	1
20	3×3	$1.2e19$	9	1	$3.8e8$	6
40	3×3	$1.4e38$	9	2	$3.8e8$	12
80	3×3	$2.1e76$	9	2	$3.8e8$	29
160	3×3	$4.7e159$	9	4	$3.8e8$	73

TABLE III: Tool performance with growing size of AG

TG Size	AG Size	O.P.S.	CTG Size	CTG Opt. Time (sec)	R.P.S.	Map Opt. Time (sec)
80	2×2	$1.46e48$	4	2	256	17
80	3×3	$2.1e76$	9	2	$3.8e8$	30
80	4×4	$2.1e96$	16	3	$1.84e19$	39
80	5×5	$6.8e111$	25	3	$8.8e34$	50
80	6×6	$3.1e124$	36	3	$1e56$	70

number of edges is 150% of the number of nodes. These TGs will be clustered to CTGs with number of nodes equal to the number of tiles in the AG. Later, CTG optimization will be performed using only 1000 steps and finally LS mapping optimization will be performed using 1000 mapping steps. Tables II and III show the time spent to perform each optimization along with the Original Problem Size (O.P.S) and Reduced Problem Size (R.P.S.) due to clustering. These results are obtained by running the tool on a 64-bit linux machine, running on a 3.4 GHz Intel©Core™-i7-6700 CPU with 32GB of Memory. However, as the problem size grows, regardless of the clustering, every task and every communication between the tasks should be scheduled in each mapping step which results in a longer optimization process.

XI. Conclusions

In this paper, a framework for dependable mixed critical task deployment on NoC based many-core systems which is accompanied by an open-source tool for simulations was introduced. Different parts of the framework were discussed and comparisons with similar frameworks were made. The proposed approach enables fault detection, isolation and reconfiguration by incorporating health monitoring and mapping/scheduling units. As future work, we plan to connect the tool to a cycle-accurate NoC simulator, in order to acquire cycle-accurate results for the simulation of the considered application, along with system's evaluation in terms of average latency, throughput and communication energy. Furthermore, the issue of heat dissipation should be addressed, as the mapping-scheduling process should not create hotspots in the network, especially in case of scheduling testing applications. Different fault classification methods are under investigation and will soon be added to the tool. Moreover, implementation and emulation of the whole NoC on a Field Programmable Gate Array (FPGA) which can be used to verify the results provided by the tool, is in progress.

Acknowledgments

The work has been supported by EU FP7 STREP BASTION, EUs H2020 RIA IMMORTAL, EUs 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] S. Murali and G. De Micheli, "Sunmap: a tool for automatic topology selection and generation for nocs," in *Design Automation Conference, 2004. Proceedings. 41st*, July 2004, pp. 914–919.

[2] S. Saeidi, A. Khademzadeh, and A. Mehran, "Smap: An intelligent mapping tool for network on chip," in *Signals, Circuits and Systems, 2007. ISSCS 2007. International Symposium on*, vol. 1, July 2007, pp. 1–4.

[3] J. Joven, O. Font-Bach, D. Castells-Rufas, R. Martinez, L. Teres, and J. Carrabina, "xenoc - an experimental network-on-chip environment for parallel distributed computing on noc-based mpsoc architectures," in *Parallel, Distributed and Network-Based Processing, 2008. PDP 2008. 16th Euromicro Conference on*, Feb 2008, pp. 141–148.

[4] "Atlas: A noc generation and evaluation framework," https://corfu.pucrs.br/redmine/projects/atlas.

[5] E. Carara, R. de Oliveira, N. Calazans, and F. Moraes, "Hemps - a framework for noc-based mpsoc generation," in *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*, May 2009, pp. 1345–1348.

[6] H. Ying, T. Hollstein, and K. Hofmann, "Gsnoc; the comprehensive design platform for 3-dimensional networks-on-chip based many core embedded systems," in *High Performance Computing and Simulation (HPCS), 2013 International Conference on*, July 2013, pp. 217–223.

[7] A. T. Tran and B. M. Baas, "Noctweak: a highly parameterizable simulator for early exploration of performance and energy of networks on-chip."

[8] F. Singhoff, J. Legrand, L. Nana, and L. Marcé, "Cheddar: A flexible real time scheduling framework," in *Proceedings of the 2004 Annual ACM SIGAda International Conference on Ada: The Engineering of Correct and Reliable Software for Real-time &Amp; Distributed Systems Using Ada and Related Technologies*, ser. SIGAda '04. New York, NY, USA: ACM, 2004, pp. 1–8. [Online]. Available: http://doi.acm.org/10.1145/1032297.1032298

[9] T. Hollstein, S. Payandeh Azad, T. Kogge, and B. Niazmand, "Mixed-criticality noc partitioning based on the nocdepend dependability technique," in *Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2015 10th International Symposium on*, June 2015, pp. 1–8.

[10] T. Hollstein, S. Payandeh Azad, T. Kogge, H. Ying, and K. Hofmann, "Nocdepend: A flexible and scalable dependability technique for 3d networks-on-chip," in *Design and Diagnostics of Electronic Circuits Systems (DDECS), 2015 IEEE 18th International Symposium on*, April 2015, pp. 75–78.

[11] 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

[12] T. Kogge, "Graph-based methods for property evaluation of on-chip routing algorithms and implementation of a scalable network-on-chip dependability layer," Bachelor Thesis, Technischen Universitt Darmstadt, October 2015.

[13] G.-M. Chiu, "The odd-even turn model for adaptive routing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 11, no. 7, pp. 729–738, Jul. 2000. [Online]. Available: http://dx.doi.org/10.1109/71.877831

[14] S. P. Azad, B. Niazmand, N. George, and R. pihlak, "$SoCDep^2$," https://github.com/siavooshpayandehazad/SoCDep2, 2015.

[15] "Mapping exp. script," https://github.com/siavooshpayandehazad/SoCDep2/blob/master/src/main/scripts/ReCoSoC_2016.

Appendix D: Publication C

S. P. Azad et al., ”From online fault detection to fault management in Network-on-Chips: A ground-up approach,” 2017 IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), Dresden, Germany, 2017.

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

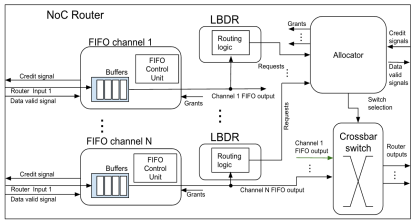


Fig. 1. Overview of the architecture of baseline credit-based flow control NoC router used in Bonfire network

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

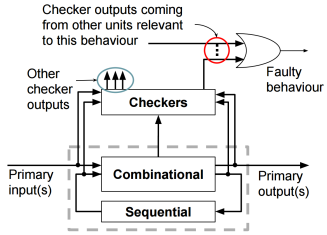


Fig. 2. General structure of the fault detection, grouping and classification mechanisms

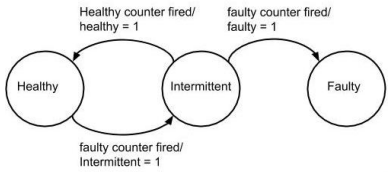


Fig. 3. Finite State Machine (FSM) for the fault classifier unit

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

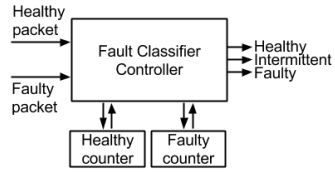


Fig. 4. Fault classifier block diagram

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

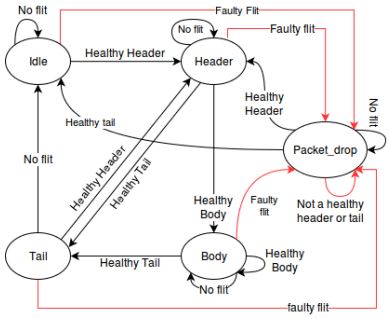


Fig. 5. Finite State Machine (FSM) diagram for the packet dropping mechanism

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

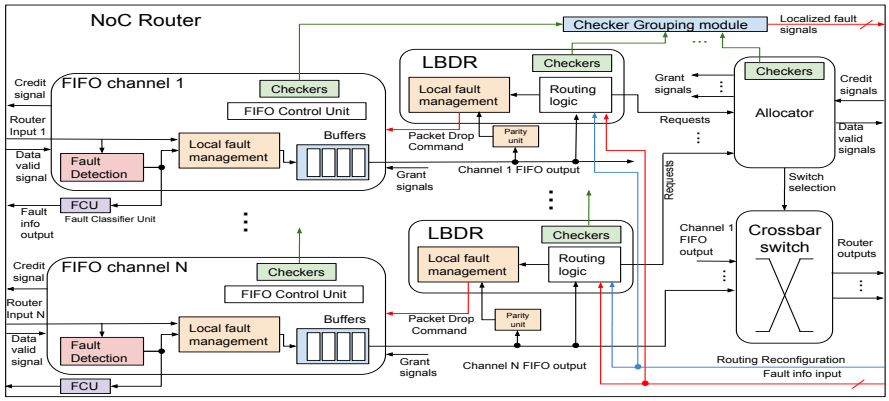


Fig. 6. Overview of the architecture of baseline credit-based equipped with fault tolerance mechanisms

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.

Appendix E: Publication 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, Cyprus, 2017.

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.

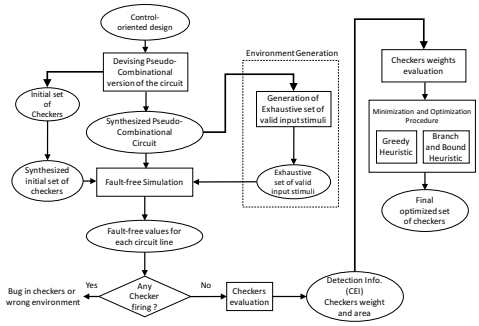


Fig. 1: Checkers Evaluation and Minimization Framework Flowchart

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] A. I. Jutman, 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 F: Publication E

S. P. Azad, B. Niazmand, K. Janson, T. Kogge, J. Raik, G. Jervan, T. Hollstein, ”Comprehensive Performance and Robustness Analysis of 2D Turn Models for Network-on-Chips,” 2017 IEEE International Symposium on Circuit and Systems (ISCAS), Baltimore, USA, 2017.

Comprehensive Performance and Robustness Analysis of 2D Turn Models for Network-on-Chips

Siavoosh Payandeh Azad *, Behrad Niazmand*, Karl Janson*, Thilo Kogge‡,
Jaan Raik*, Gert Jervan*, Thomas Hollstein*†
Department of Computer Engineering
*Tallinn University of Technology, † Frankfurt University of Applied Sciences, ‡Darmstadt University of Technology
email: {siavoosh, bniazmand, karl.janson , thilo, jaan.raik, gert.jervan, thomas}@ati.ttu.ee

***Abstract*—Routing algorithms play an important role in Network-on-Chip (NoC) based System-on-Chips. Turn model based routing disallows some of the turns in order to avoid deadlock, while providing partial adaptivity. In this paper, all 2D uniform turn models are examined for deadlock freeness and connectivity; 50 deadlock free turn models are extracted that provide full connectivity in the network. An extended adaptivity metric is introduced to classify the turn models; all extracted turn models are compared in terms of adaptivity, robustness and latency. Experimental results identify the most robust turn models and the most efficient ones in terms of latency.**

***Keywords*—Turn Model, Routing Algorithm, Robustness, Minimal Path, Network-on-Chip.**

I. Introduction

Network-on-Chip (NoC) has emerged as a paradigm to overcome some of the limitations existing in the conventional shared medium bus-based architectures [1], such as performance and scalability issues. In a NoC, the communication between cores is administered by on-chip routers based on a routing algorithm. Routing algorithms can be either classified as deterministic or adaptive [2]. Deterministic routing algorithms, use a single path for each source-destination pair, whereas adaptive routing provides more path diversity, taking into account criteria such as traffic load on links, etc. One of the important factors when choosing a routing algorithm is deadlock freeness. Deadlock occurs when a cyclic dependency is created between the packets in a NoC, waiting for resources held by other packets in the cycle [2].

Two main approaches exist for addressing deadlocks: deadlock avoidance where deadlocks are completely avoided and deadlock recovery where deadlock can happen, but is handled using a deadlock recovery mechanism. The focus of this paper is on deadlock avoidance. *Turn model* approach was first introduced in [3] for deadlock avoidance in 2D Mesh Network-on-Chips. A turn is defined as a change of direction in a packet's path. Directions are named based on cardinal directions: North (N), East (E), West (W) and South (S). In a 2D Mesh network, maximum of 8 *turns* exist: N2E, N2W, E2N, E2S, W2N, W2S, S2E and S2W. For instance S2E indicates a ↱ turn that, if allowed, it enables a packet coming from the South input port of the router be forwarded to the East output port. A total number of $2^8 = 256$ uniform 2D turn models can be derived from eight possible turns. In a uniform turn model, all network nodes have the same disallowed turns.

In [3], three deadlock free turn models are introduced, *i.e.* West-First, North-Last and Negative-First. Furthermore, [4] introduces the North-South First (NSF) turn model, by combining North-First and South-First. Moreover, the East-First turn model is addressed in [5].

Even though, some of the previous works such as [6] have covered performance comparison of some of the well-known turn models, to the best of our knowledge, exploration of the entire search space of all possible turn models for a 2D mesh NoC have not been thoroughly performed. In this paper, all 256 uniform turn models for routing in 2D Mesh NoCs are enumerated and the characteristics of the deadlock-free turn models are evaluated. The proposed approach uses metrics for connectivity and adaptivity, based on which turn models are classified. In addition, latency and robustness of all turn model groups are assessed.

The rest of this paper is organized as following: in section II a methodology for evaluation of deadlock freeness of turn models is discussed. In section III metrics for network connectivity and routing adaptiveness are introduced. In section IV robustness and latency of the chosen routing algorithms have been evaluated in minimal and non-minimal routing. Finally, section V concludes the paper.

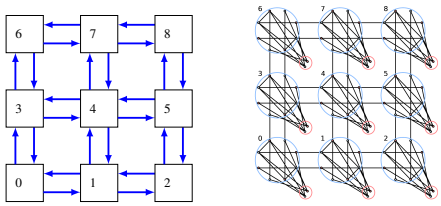
II. Evaluating Deadlock Freeness

In [7], the concept of Cyclic Dependency Graph (CDG) is used for deadlock detection, where nodes represent the network channels and edges denote the channels dependency. It is proven that for guaranteeing deadlock-freeness, CDG must be acyclic. In this work similar approach has been used, but instead of CDG, we use the concept of Routing Graphs (RGs) which was introduced in [8]. The rest of this section overviews the construction of routing graphs and describes the process of evaluating deadlock freeness.

A. Routing Graph

A Routing Graph, RG(V, E), is a directed graph, in which the set of vertices (V) denotes the set of all the input/output ports in the network (two nodes per port) and the set of edges (E) represent the set of (v_i, v_j) where v_i is a vertex (input or output port) that is depending on v_j port. For the sake of simplicity, a vertex v in RG is denoted as $node_{i,p,dir}$, which describes direction $dir \in \{in, out\}$ of port $p \in \{N, E, W, S, L\}$ of node i in the network. There are two different types of links represented as edges in the routing graph:

- *Inter-router edges*, representing connections between routers (from an output port of a router to an input port of an adjacent router).



(a) (b)

Fig. 1. a) Example of a 3 × 3 mesh network and b) the resulting routing graph for XY routing algorithm

- *Intra-router links*, representing allowed connections inside the router (from an input port of a router to an output port in the same router). An intra-router link can be: 1) From or to local port: representing dependency between the router's north, east, west and south ports and the local port connected to the processing element (PE). 2) Straight connections: describing dependency between ports involved in maintaining straight connections inside the router (e.g from west input to east output port of a router). 3) Turns: dependency of the ports in perpendicular direction (e.g. from east input to south output port).

As an example, Fig. 1b shows the RG for XY routing for a 3x3 2D Mesh network, corresponding to Fig. 1a. In order for a routing algorithm to be deadlock free, its corresponding RG must also be acyclic.

B. Proof of Deadlock Freeness

Theorem 1. *A deadlock in a turn model results in a cycle in the RG derived from the turn model.*

Proof. Let us assume that a deadlock in the turn model results in a RG with no cycles. Since the RG represents a sequence of all dependencies between the inputs and outputs of a routers under the applied turn model and there are no cycles in RG, there cannot be cyclic dependencies between the inputs and outputs of routers in the network. Hence, no deadlock can be formed. This is in contradiction with our initial assumption of having a deadlock for the turn model. This means that a deadlock in a turn model results in a cycle in the RG derived from the turn model. □

Using this method, it is possible to discard 35 turn models that have deadlocks, which leaves 221 deadlock free turn models.

III. Metrics for Connectivity and Adaptivity

Out of 221 deadlock free turn models, some provide partial connectivity. Examples of this case are: one turn model with zero turns and 8 turn models with one turn. To evaluate the connectivity of turn models, a simple metric has been used:

$$Connectivity_{RG} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} C_{i,j,RG} \tag{1}$$

where N is number of nodes in the network and:

$$C_{i,j,RG} = \begin{cases} 1 & \text{if exists a path from } node_{i,L,out} \text{ to } node_{j,L,in} \\ & \text{in } RG \text{ where } i \neq j \\ 0 & \text{otherwise} \end{cases}$$

TABLE I
List of previously named turn models

#	Allowed turns	Conventional Name
0	E2N, E2S, W2N, W2S	XY [9]
13	S2W, S2E, N2W, N2E	YX [10]
33	E2S, S2W, S2E, N2W, N2E	Restricted North First [4]
39	E2N, E2S, W2N, W2S, S2W, N2W	East-First [5]
40	E2N, E2S, W2N, W2S, S2E, N2E	West-First [3]
41	E2N, E2S, W2N, W2S, N2W, N2E	North-Last [3]
42	E2N, E2S, W2N, S2E, N2W, N2E	Negative-First [3]
46	E2N, W2N, S2W, S2E, N2W, N2E	South-First [4]
48	E2S, W2S, S2W, S2E, N2W, N2E	North-First [4]

TABLE II
DoA and DoA_{Ex} for all 2D routing algorithms of Fig. 2

	4 turns			5 turns		6 turns	
Turn Model Num	0, 13	3, 5, 8, 10	1, 2, 4, 6, 7, 9, 11, 12	14, 15, 16, 17, 28, 33, 36, 37	18-27, 29-32, 34, 35	42, 43, 45, 47	38, 39, 40, 41, 44, 46, 48, 49
DoA	1			1.23		1.43	
DoA_{Ex}	1	1.41	1.63	2.11	2.41	3.83	4.33

and RG is representing the routing graph. In this case, the assumption is that *RG* represents a full mesh under a turn model based routing algorithm. Path search in RG can be either minimal or non-minimal. For a 3 × 3 mesh, the maximum connectivity is 72 which means that each node can communicate with eight other nodes. Using this simple metric, all the deadlock free turn models were evaluated and turn models that do not provide full connectivity have been excluded. There are only 50 uniform turn models that are deadlock free and also provide full connectivity. These turn models are visualized in Fig. 2. There are 14 four-turn turn models, 24 five-turn turn models and 12 six-turn turn models. Table I lists the turn models which are previously named and addressed in the literature.

In order to classify turn models, we can use the Degree of Adaptiveness (DoA) introduced in [3] which only considers the shortest paths from a source node to destination node. A general form of DoA metric can be formulated as:

$$DoA = \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} NoSP_{i,j,rg}}{\text{number of pairs of nodes}} \tag{2}$$

Where N is the number of nodes in the network and $NoSP_{i,j}$ is defined as:

$$NoSP_{i,j,RG} = \begin{cases} \text{number of shortest paths in } RG \text{ from} & i \neq j \\ node_{i,L,out} \text{ to } node_{j,L,in} & \\ 0 & \text{otherwise} \end{cases}$$

The resulting DoA for the turn models are presented in Table II. However, it is no surprise that turn models with higher number of turns, also have higher DoA. Extending this metric (DoA_{Ex}) to include all the simple paths in the network (paths that do not have repeating nodes in them) provides a slightly different picture than the original DoA. DoA_{Ex} makes it possible to classify the turn models even further. The DoA_{Ex} metric can be described as follows:

$$DoA_{Ex} = \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} NoSP'_{i,j,RG}}{\text{number of pairs of nodes}} \tag{3}$$

Where N is the number of nodes in the network and $NoSP'_{i,j}$ is defined as:

$$NoSP'_{i,j,RG} = \begin{cases} \text{number of simple paths in } RG \text{ from} & i \neq j \\ node_{i,L,out} \text{ to } node_{j,L,in} & \\ 0 & \text{otherwise} \end{cases}$$

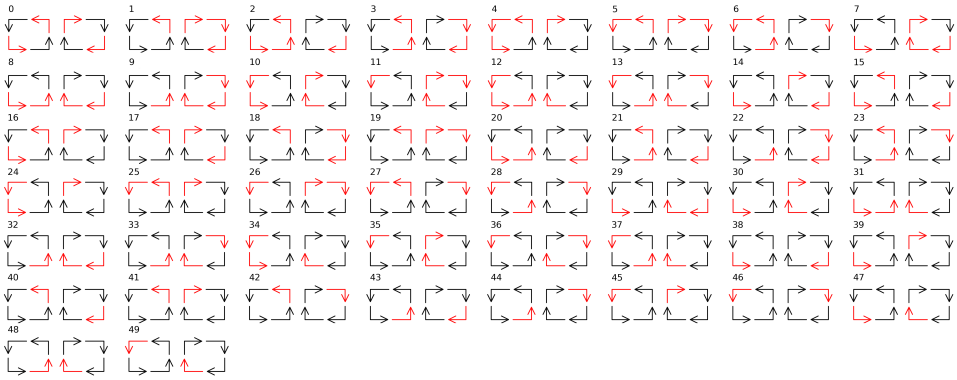


Fig. 2. Visualization of all deadlock free 2D turn models with full connectivity. The forbidden turns are drawn in red.

Table II presents DoA and DoA_{Ex} metrics for the turn models in Fig. 2. This table shows that inside each class of turn models (four, five, and six-turn turn models), there are sub-classes that have different characteristics. As an example, turn model no. 3 shares two turns with XY and two turns with YX which allows it to have non-minimal de-routes. Similarly, under non-minimal routing, turn model no. 1 and 2 have even further advantage in providing path diversity.

Similar investigation has been conducted for 3D routing algorithms, where there are 24 turns available. Out of over 16 million possible 3D turn models, over 95 thousand deadlock free turn models which provide full connectivity in the network are extracted. The full list of these turn models and their visualizations are accessible at [11].

IV. Experimental Results

In this section, all the 50 turn models extracted in the previous section, are compared in terms of robustness and latency.

A. Robustness Evaluation

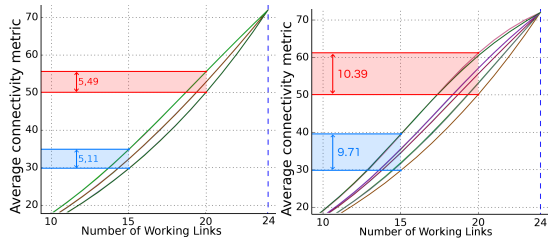
Robustness of a routing algorithm is defined here as the average connectivity metric of routing algorithm running on a 2D mesh NoC with specific number of faulty links. Experiments were conducted to evaluate the robustness of each routing algorithm. Using the connectivity metric introduced in section III, average connectivity of the network with n_{broken} permanently broken links (out of total number of n_{total} inter-router links), can be calculated using Algorithm 1. Where avg_{con} is a list of average connectivities and $avg_{con}[k]$ is average connectivity for a 3×3 network with k broken links.

For each turn model and each possible amount of broken links (n_{broken}) the connectivity metric is calculated and averaged over all possible configurations with n_{broken} links.

Fig. 3 illustrates the difference between average connectivity metrics for the turn models listed in Table II. Fig 3a shows the average connectivity metric of the turn models under minimal path routing for different number of working links. The three lines in the figure correspond to three classes of DoA in Table II, where turn models with higher DoA provide better connectivity. However, the gap between the curves is not substantial. Fig 3b depicts the average connectivity metric of turn models under non-minimal routing for different numbers of working links. In this case the curves diverge more, which corresponds to the seven classes, when considering DoA_{Ex}, and the gap between the curves is rather substantial where turn models with higher DoA_{Ex} provide better connectivity (Red and Blue marked regions in Fig 3).

Algorithm 1: average connectivity calculation algorithm

for $n_{broken} \in [0, n_{total}]$ **do**
  $list\_of\_configurations$ = list of all 3×3 2D mesh NoCs with n_{broken} broken links
  $sum_{con} = 0$
  forall $conf_{broken} \in list\_of\_configurations$ **do**
    Generate RG based on $conf_{broken}$
    sum_{con} += $Connectivity_{rg}$
  end
  $avg_{con}[n_{broken}] = sum_{con}$/len(list_of_configurations)
end
return avg_{con}



(a) (b)

Fig. 3. Comparison of avg. connectivity metric of turn models under a) minimal, b) non-minimal routing by number of available links.

B. Latency Evaluation

In this section, all the 50 deadlock free turn models shown in Fig. 2 are evaluated under synthetic traffic patterns using Noxim [12] NoC simulator. The experimentation setup parameters are as follows: A 4x4 2D mesh was considered. The system clock frequency is set to 1 GHz for all routers. As a synthetic traffic pattern, random uniform is considered in the simulations. Packets are generated using Poisson distribution.

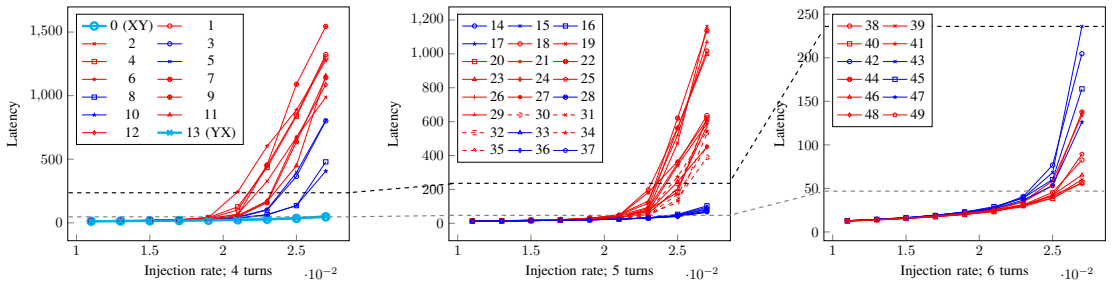


Fig. 4. Latency results under random uniform traffic for a) four, b) five, c) six-turn turn models.

The length of the packets is fixed and set to 8 flits, and FIFO depth of routers is 4 flits. For each simulation, the warm-up time is considered 1000 cycles in order to allow the transient effects to stabilize and subsequently, the simulation has been run up to 20000 cycles.

The average latency results are grouped based on the number of turns allowed, *i.e.* 4-turn, 5-turn and 6-turn turn models. Fig. 4a-c shows the average latency results for these turn models under random uniform traffic pattern (with packet injection rate ranging from 0.001 to 0.025). The curves are color coded in each figure to distinguish different classes of DoA_{Ex} (see Table II). The dotted lines in Fig. 4a-c indicate the corresponding highest value of the Fig. 4c and lowest value of Fig. 4a, since the range of axes is different between Fig. 4a-c. As it can be observed in Fig. 4a, two of the turn models (0 and 13 which are highlighted with thick cyan color), which correspond to XY and YX routing, outperform the other turn models in terms of average latency which conforms to the observations made in [13]. After those two turn models, both classes of 6-turn turn models and 5-turn turn models with lower DoA_{Ex} perform better than others. The performance (average latency and average throughput) and communication energy of all the 50 deadlock free turn models under different synthetic (Random uniform, Bit-reversal, Shuffle, Transpose and Butterfly) traffic patterns are available at [11].

V. Conclusion

In this paper, an in depth comparison of all 2D uniform turn models in terms of connectivity, adaptivity and robustness has been made. All possible 50 deadlock-free turn models that provide full connectivity have been extracted. An extended adaptivity metric have been introduced to classify the turn models even further. It became clear that one class of turn models with highest degree of adaptivity is much more robust under non-minimal path routing. The latency results shows that aside from XY and YX routing algorithms which outperform all other turn models under random uniform traffic pattern, 2 more classes fall very close to this class which are 5-turn turn models with lower adaptivity metric and 6-turn turn models with higher adaptivity metric.

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] S. Kumar, A. Jantsch, J. P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani, "A network on chip architecture and design methodology," in *VLSI, 2002. Proceedings. IEEE Computer Society Annual Symposium on*, 2002, pp. 105–112.

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

[3] C. J. Glass and L. M. Ni, "The turn model for adaptive routing," in *Computer Architecture, 1992. Proceedings., The 19th Annual International Symposium on*, 1992, pp. 278–287.

[4] Y. Miura, K. Shimozono, S. Watanabe, and K. Matoyama, "An Adaptive Routing of the 2-D Torus Network Based on Turn Model," in *Computing and Networking (CANDAR), 2013 First International Symposium on*, Dec 2013, pp. 587–591.

[5] S. Mubeen and S. Kumar, "Designing efficient source routing for mesh topology network on chip platforms," in *Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on*, Sept 2010, pp. 181–188.

[6] A. Patooghy and H. Sarbazi-Azad, "Performance comparison of partially adaptive routing algorithms," in *20th International Conference on Advanced Information Networking and Applications - Volume 1 (AINA'06)*, vol. 2, April 2006, pp. 5 pp.–.

[7] J. Duato, "A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 4, no. 12, pp. 1320–1331, Dec. 1993. [Online]. Available: http://dx.doi.org/10.1109/71.250114

[8] 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.

[9] J. Duato, S. Yalamanchili, and N. Lionel, *Interconnection Networks: An Engineering Approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002.

[10] A. M. Shafiee, M. Montazeri, and M. Nikdast, "An innovational intermittent algorithm in networks-on-chip (noc)," *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 2, no. 9, pp. 2907 – 2909, 2008. [Online]. Available: http://waset.org/Publications?p=21

[11] " Turn Model website ," http://turnmodel.pld.ttu.ee/, 2016.

[12] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti, "Noxim: An open, extensible and cycle-accurate network on chip simulator," in *2015 IEEE 26th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, July 2015, pp. 162–163.

[13] 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.

Appendix G: Publication F

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, Spain, 2017.

Fault-Resilient NoC Router with Transparent Resource Allocation

Tsotne Putkaradze, Siavoosh Payandeh Azad, Behrad Niazmand, Jaan Raik, Gert Jervan
Department of Computer Engineering
Tallinn University of Technology
{tsotne, siavoosh, bniazmand, jaan.raik, gert.jervan}@ati.ttu.ee

***Abstract*—The current trend of aggressive technology scaling results in a decrease in system's reliability. This motivates investigation of fault-resilient architectures which provide graceful degradation of system's functionality. In this paper, three novel fault-resilient Network-on-Chip (NoC) router architectures are proposed. These 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. In contrast to Silicone Protection Factor (SPF) metric, the proposed metric also takes into account the areas of different units. Area overhead and reliability of proposed architectures are compared with Triple Modular Redundancy (TMR) and Unit-Duplication mechanisms. All proposed architectures showed remarkable reliability improvement compared to original, TMR and Unit Duplication architectures; while at the same time, their area overhead is less than or equal to unit-duplication mechanisms.**

I. Introduction

Network-on-Chip (NoC) has emerged as an interconnection network infrastructure to overcome the performance and scalability issues of the previously existing shared medium bus-based architectures [1]–[3]. Even though NoCs provide many advantages over bus-based architectures, the extreme down-scaling of digital circuits jeopardizes the reliability of the system and makes it prone to wear-out and increases the probability of permanent faults.

This paper introduces three architectures and presents the comparison with Unit-Duplication and TMR architectures, in terms of area and reliability improvements. The novelty of developed architectures is the transparent replacement of any faulty router unit with either a spare unit or a unit with less priority or with a healthy unit that is located in a faulty turn. Architectures are exploiting the regularity of the router to obtain this ability. This paper also introduces a new metric for reliability comparisons, based on the concept of reliability block diagrams [4] and provides an analytical assessment of the dependability of the proposed architectures.

The rest of this paper is organized as follows, section II is the literature review, section III talks about NoC router that was used during the work. In section IV proposed architectures are described. In section V the details of synthesis and reliability improvements are presented and discussed.

978-1-5386-3344-1/17/$31.00 ©2017 IEEE

II. Related Work

Fault tolerance in Network-on-Chips (NoCs) is a broad topic. The approaches in this research area can be classified based on (1) hardware redundancy, (2) information redundancy and coding techniques, and (3) resource sharing (which, in turn, is classified as internal and external resource sharing).

Two common techniques usually used for fault tolerance which are categorized as hardware redundancy techniques are Dual Modular Redundancy (DMR) [5]–[8] and Triple Modular Redundancy (TMR) [9]–[14]. Constantinides et al. have introduced BulletProof router [11] and a method that automatically injects sparing logic to the net-list of a router and is able to replace faulty circuitry. However, it imposes area overhead of up to 3.4× with respect to the baseline non-fault tolerant router. In [5] Shield NoC router is introduced, in which DMR is used for making parts of the NoC router tolerant to hard errors. However, considering Virtual Channel (VC) and VC allocator and hardening these modules against hard faults incurs additional area overhead. Moreover, Shield imposes a 31% power overhead at the price of providing fault tolerance. [9] utilizes redundant multiplexers (MUXes) in the input port controller of the router. In [6] all the physical links between the routers are doubled in order to improve the reliability of the NoC, by maintaining its connectivity. Limitations of this work are that it is not considering the faults in internal circuit of the router and the additional overhead is incurred due to replicating the links between routers.

In [7] Crosspoint redundancy and redundant links are used to improve yield and reliability of on-chip communication fabrics, including the crossbar switch. In [8], redundant routing computation unit is used. In the case of a fault, the circuit is replaced by a light-weight routing unit, which is static but provides reliable routing services. [12] and [13] use TMR in order to protect the control signals of a NoC router. [9] uses TMR for control part of FIFO. In [10], TMR is used to harden the multiplexers of the crossbar switch. In [15], the network is divided into 2x2 blocks, with a spare router in each block. The main drawback of DMR and TMR-based approaches is the imposed area overhead, especially in the case of TMR, whose additional area is incurred due to using a voter to vote among the outputs of the three redundant replicas. Moreover, the voter can become a single point of failure if it becomes defective. Moreover, in methods such as [15] in which spares

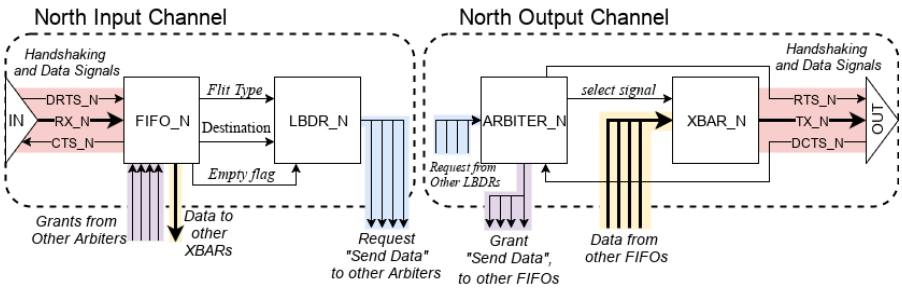


Fig. 1. Block diagram of original version of Bonfire router. The figure shows only a North channel. Exact same architecture is for other channels.

are applied at the granularity level of routers rather than router components, significant area overhead is incurred.

Another group of works address fault tolerance via information redundancy and error detection/correction coding techniques [9], [10], [12]. In [9] Hamming coding is used to make data links of the NoC routers fault-tolerant. One of the drawbacks of Hamming coding is the overhead of added bits for error detection and correction, as it is dependent on the original data's number of bits. Authors of [10] utilize row and column parity bits in order to protect routing tables against Single Event Upsets (SEUs). However, all the above mentioned methodologies apply to the data-path only and do not detect and correct the faults in the functionality of the architecture itself, hence, do not provide any improvement in fault tolerance of the control circuits.

The third approach is exploitation of regularity of the network by resource sharing. There are two main approaches in this category; internal resource sharing and external resource sharing. In internal resource sharing, functionality is usually borrowed from a module inside the same router, if a component gets faulty, whereas, in external resource sharing, the functionality is borrowed from the component of a neighbor or adjacent router. In the literature, the following works [9], [14], [16], [17] have utilized internal resource sharing. In [16] the NoC router is able to tolerate faults in the FIFO module by replacing the defective module with a healthy one from another input port (input port swapping). One of the drawbacks of this work is its high hardware overhead (as stated in [17]). In [9], fault tolerance for data-path of FIFO (the memory elements) is provided via reconfiguration. This is performed by borrowing from adjacent FIFOs of the same router. The approach in [17] uses decoupled resource sharing approach in input ports, in which a faulty input port can use its adjacent ports for data transmission. Also, a fault tolerant crossbar switch is utilized. However, this approach cannot handle the case in which both adjacent input ports of a port are faulty. Moreover, fault tolerance in the arbitration logic of the router is not addressed. [14] uses slice sharing (three identical router slices) for fault-tolerance. In case of occurrence of a fault in any of the slices, resource sharing is enabled in order to improve fault-tolerance. However, all the fault-tolerance advantages come at the price of performance degradation.

RoCo [18] uses external resource sharing to provide fault tolerance, in which, functionality is borrowed from neighbor switches in order to recover from the faults occurring in the current switch's components. One of the limitations of this work is the complexity of its virtual channel allocation logic. Similar to [5], this approach also provides a bypass path for the crossbar switch. In [19], switches are equipped with internal Default Backup Paths (DBPs), which are connected to form a ring topology if the regular topology has faulty resources. However, this approach degrades performance (latency) in order to provide connectivity between all Processing Elements (PEs) even when all the routers are faulty.

Existing architectures provide reliability improvements with the cost of performance degradation or they only address the reliability of data path. Also, the above mentioned approaches use the Silicon Protection Factor (SPF [11]) metric for reliability comparison which has many shortcomings (see section V). This paper focuses mainly on addressing transparent resource sharing and re-using of components to improve NoC routers reliability. Three architectures are presented and compared against Unit-Duplication and TMR approaches in terms of area and reliability. The comparison of routers is done through a new metric developed and described in this paper which covers the shortcomings of SPF. The main contributions of this work are the new metric for reliability comparison and developed architectures that can replace any defective router unit: (1) with a spare unit or (2) with a unit with less priority or (3) with a healthy unit that is located in a faulty turn.

III. Bonfire Network-on-Chip Router

Bonfire [20] router, an open-source wormhole switching NoC router, is used in this work. It is chosen due to its simplicity and regularity, which makes it easier to implement the ideas of architectures proposed in this paper.

The Bonfire router consists of four parts: 1) FIFO Buffer that is used to store the flits of a packet in the router until it is being sent. The current design, used in synthesis and simulations, has the size of four flits. 2) Routing logic: which is implemented using Logic-Based Distributed Routing (LBDR) [21], to decide where the packet will be transmitted. Currently,

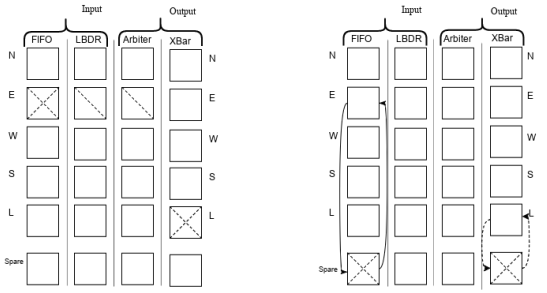


Fig. 2. Possible Fault scenario in router and the router after reconfiguration. Single Spare version of router. ⊠ - represents faulty unit. ▨ - unit that is healthy but is connected to faulty unit.

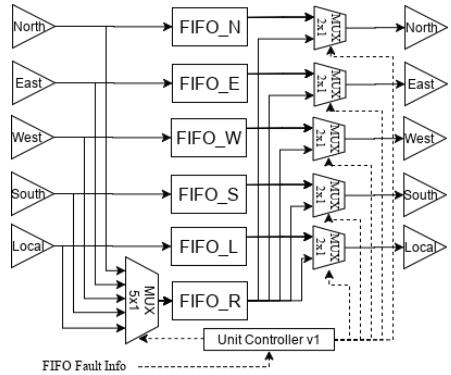


Fig. 3. Single Spare architecture. FIFO units, multiplexers and Unit Controller. suffixes _N, _E, _W, _S, _L denote the channel names: North, East, West, South, Local. Turn Priority version of router.

XY routing algorithm is used. However, it can be reconfigured to any deadlock-free turn model based routing algorithm. LBDR acquires destination information from header flit that is provided by the FIFO. 3) Arbiter - which is responsible for communicating with neighboring routers that provide information on empty spaces in their FIFOs. It also decides which input request on specific output, will be served first. Current Arbiter implements Round-Robin algorithm, that prioritizes neighbors dynamically. 4) XBAR - the crossbar switch which is used to direct the data into neighbor routers. It is controlled by the arbiter.

Figure 1 shows the architecture of router. In total, there are five input and five output channels in the router, North, East, West, South and Local channels. An input channel consists of FIFO and LBDR modules. An Output channel consists of Arbiter and XBAR. A turn consists of a pair of input and output channels where those channels do not belong to the same direction. In total there are 20 turns (5×4) in each router.

IV. Proposed Architectures

In this work three fault tolerant architectures are proposed, which operate by using resource managing and sharing. The resource sharing granularity is at the unit level (FIFO, LBDR, Arbiter, XBAR) and will be performed either on the system's functional units or on spare units. In all three architectures, a unit controller is in charge of selecting the module that provides the output when necessary. The fault information provided to the unit controller is fed from a set of fault monitors, called concurrent online checkers [22], [23], which perform the fault detection in each of the control part modules (control part of FIFO, LBDR and arbiter). For the data-path of FIFO and crossbar switch (XBAR), a single parity checker detects the fault. The following subsections provide a detailed description of all three architectures.

A. Architecture 1 : Single Spare

The single Spare architecture utilizes spare units available to replace faulty units. The architecture includes a Unit Controller, that selects the spare units when necessary. As mentioned before, the router has four different units, and correspondingly, the current implementation of Single Spare architecture has a single spare unit of each type.

Fig 2 depicts a possible fault scenario in the router. On the left side of the picture, it is assumed there is a fault in FIFO of East input channel. In this scenario then LBDR of the east channel is healthy but it gets incorrect data from FIFO and correspondingly data sent to any direction is faulty.

In the original version of the router, if the north input channel's LBDR is faulty corresponding FIFO's correct functionality goes in vain; Because faulty LBDR might transmit data in the wrong direction. On the other hand, in Single Spare architecture, LBDR or any other faulty unit can be replaced with a spare unit.

Figure 3 displays the schematic of Single Spare architecture only for FIFO units in the router. It is visible that only the spare FIFO has five-input-one-output multiplexer and its output can be multiplexed with any of the FIFO output, all multiplexers are controlled from the Unit Controller. The Unit Controller receives information about fault locations and replaces a faulty unit with the spare one, by selecting the right multiplexers. The Unit Controller is also capable of prioritizing the channels. For example in a case that there are faults on North and Local channels, it can select the channel with higher pre-programmed priority. For each type of unit in the router, there is a separate Unit Controller.

B. Architecture 2 : Turn Priority

Turn Priority version of router exploits the regularity of the router. It gathers data on all units and then tries to utilize all available units to assemble as many input and output channels as possible. The unit controller prioritizes the channels in the following order: Local, North, East, West, South for Input channel and East, West, South, North, Local for output channel. This order of prioritization ensures that in the worst case, the current node can still send packets in the network.. This operation can be done without any modification in the

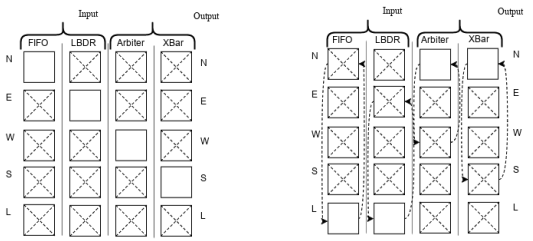


Fig. 4. Possible Fault scenario in router and the router after reconfiguration. Turn Priority version of router. ⊠ - represents faulty unit.

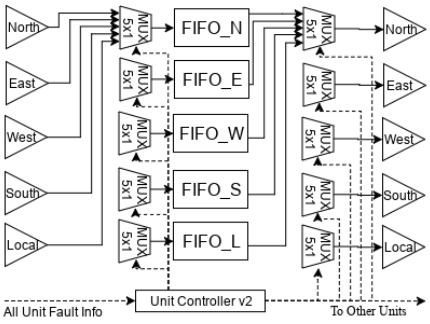


Fig. 5. Turn Priority architecture. FIFO units, multiplexers and Unit Controller.

units since all the units of the same type are identical. There is only one Unit Controller in each router since it should perform router-wide configuration. In comparison with Single Spare, this architecture does not have extra spare units; it only consists of the original circuit with added multiplexers and a Unit Controller. An example of a fault scenario is shown in Fig. 4. If there exists at least one unit of each type, it can be utilized to build a pre-defined turn, for example, North to East.

Fig. 5 depicts that the multiplexers and Unit Controller used in the design are different from Single Spare version. Multiplexers are placed in front of every FIFO so that any inputs going through any FIFO can be redirected to any other FIFO.

C. Architecture 3: Stay Alive

The idea behind the Stay Alive architecture is to combine the features of Turn Priority and Single Spare approaches. The functionality of Unit Controller used in this architecture is the same as in Turn Priority, but it controls an additional spare unit of each type which results in the larger area, Fig. 6. The additional spare unit also leads to an increase in multiplexers number (from 5 to 6) and size(6×1 instead of 5×1 multiplexer in outputs).

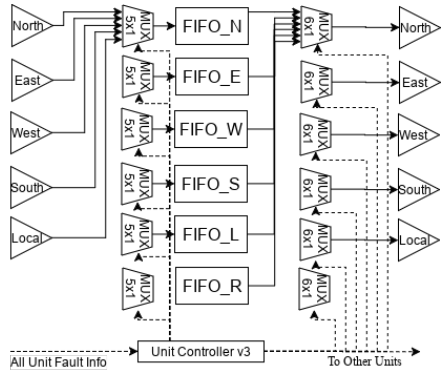


Fig. 6. Stay Alive architecture. FIFO units, multiplexers and Unit Controller.

V. Reliability Analysis

This section introduces a framework for reliability comparison and provides a comparison of developed architectures with TMR and Unit Duplication architectures. For comparison with other routers SPF [11] was considered at first. SPF is a silicone protection factor used for comparing architecture reliability. It equals to:

$$SPF = \frac{Mean number of faults for failure}{\frac{\text{area of fault-tolerant architecture}}{\text{area of baseline architecture}}} \quad (1)$$

In [5], [11], [16], [17] SPF is used for assessing the reliability of the architecture. However, this methodology has following aspects missing:

- SPF does not consider the effect of area of units on the probability of faults. While units with the smaller area have less probability to be faulty. For example, in this work FIFO has the area of 14315 μm^2 and LBDR has an area of 375μm^2, then the probability of FIFO failing is extremely larger than LBDR. Thus probing that simple best case, worst case scenarios are not sufficient.
- SPF over-emphasizes the number of units during best case(max number of faults for failure) scenario calculation. For example, if we make a unit, with the smallest area, n-modular ($n \gg 3$) that will improve best-case scenario radically without a significant increase in area. Thus, improving SPF radically without significant or no improvement in reliability since the bottleneck for faults most probably is a unit with the largest area.
- SPF uses a ratio of the area of the original and modified router. This ratio will result in an unfair comparison of fault-tolerant architectures if they are applied on different original routers. For example, if we apply same fault tolerant architecture on two different original routers, one of them having initial area two times larger than other(e.g. caused by virtual channels), then the area contribution of introduced fault-tolerant architecture to the routers, are different; resulting in an unfair comparison.

Because of above-mentioned reasons analytical analysis of reliability is conducted using reliability block diagrams [4]. In this method of calculation areas of units are taken into consideration. The probability of a single turn (single input channel and single output channel) being alive is calculated. The method takes one unit as a reference point and applies multiple values of reliability, in the range [0:1]. Probabilities of rest of the units are normalized against this unit.

The reliability of a router having all channels faulty except one input channel and one output channel will be used as a comparison point of architectures. The reliability calculations will include all router units and added multiplexers. Interconnection area and Unit controller are not considered in reliability calculations. It is assumed that Unit controller is fault-free by using n-modular redundancy or other fault tolerant techniques.

A. The Original Router

In the original version of the router, one permanent fault in a unit involved in a channel used by a turn is enough to stop its correct functionality. Regardless of the location of the fault. To have at least one functional turn, at least one input channel and one output channel should be operational. The Reliability of input and output channels are shown on equations 2, 3.

Where, R_{FIFO} and R_{LBDR} refer to the reliability of failure in FIFO and LBDR. Similar definitions for rest of the units.

The reliability formula of single turn available in the original router is shown equation 4.

$$R_{IC} = R_{\text{FIFO}} \times R_{\text{LBDR}} \tag{2}$$

$$R_{OC} = R_{\text{ARBI}} \times R_{\text{XBAR}} \tag{3}$$

$$R_{\text{Original}} = R_{IC} \times R_{OC} \tag{4}$$

Fig. 7a illustrates the reliability diagram for the baseline router.

B. Single Spare version

Single Spare architecture has a higher level of dependability due to the existence of spare units. Reliability of one input and output channels are shown on equations: 5, 6, where $R_{\text{FIFO-MUX-IN}}$ stands for the reliability of multiplexer that is located on the input port of FIFO, and $R_{\text{FIFO-MUX-OUT}}$ is located on the output port of FIFO. The probability of at least single turn staying alive is shown on equation 7.

$$\begin{aligned} R_{IC} = (1-(1-R_{\text{FIFO}} \times R_{\text{FIFO-MUX-IN}}) \times (1-R_{\text{FIFO}})) \times R_{\text{FIFO-MUX-OUT}} \times \\ (1-(1-R_{\text{LBDR}} \times R_{\text{LBDR-MUX-IN}}) \times (1-R_{\text{LBDR}})) \times R_{\text{LBDR-MUX-OUT}} \end{aligned} \tag{5}$$

$$\begin{aligned} R_{OC} = (1-(1-R_{\text{ARBI}} \times R_{\text{ARBI-MUX-IN}}) \times (1-R_{\text{ARBI}})) \times R_{\text{ARBI-MUX-OUT}} \times \\ (1-(1-R_{\text{XBAR}} \times R_{\text{XBAR-MUX-IN}}) \times (1-R_{\text{XBAR}})) \times R_{\text{XBAR-MUX-OUT}} \end{aligned} \tag{6}$$

$$R_{\text{Single-Spare}} = R_{IC} \times R_{OC} \tag{7}$$

Fig. 7b shows the reliability diagram for the Single Spare architecture.

C. Turn Priority and Stay Alive version

To have at least one turn functional in Turn Priority architecture, its needed to have at least one working instance of each unit along with its input and output multiplexers. Accounting multiplexers, the reliability of input and output channels are shown on equations : 8, 9. And reliability of single turn available in the router is shown on equation 10.

$$\begin{aligned} R_{IC} = (1-(1-R_{\text{FIFO-MUX-IN}} \times R_{\text{FIFO}} \times R_{\text{FIFO-MUX-OUT}})^5) \times \\ (1-(1-R_{\text{LBDR-MUX-IN}} \times R_{\text{LBDR}} \times R_{\text{LBDR-MUX-OUT}})^5) \end{aligned} \tag{8}$$

$$\begin{aligned} R_{OC} = (1-(1-R_{\text{ARBI-MUX-IN}} \times R_{\text{ARBI}} \times R_{\text{ARBI-MUX-OUT}})^5) \times \\ (1-(1-R_{\text{XBAR-MUX-IN}} \times R_{\text{XBAR}} \times R_{\text{XBAR-MUX-OUT}})^5) \end{aligned} \tag{9}$$

$$R_{\text{Turn-Priority}} = R_{IC} \times R_{OC} \tag{10}$$

Since the difference between Turn Priority and Stay Alive version is only one extra spare unit of each type; The probability of survival for Stay Alive version can be similarly calculated. Formulas are shown on equations : 11, 12, 13.

$$\begin{aligned} R_{IC} = (1-(1-R_{\text{FIFO-MUX-IN}} \times R_{\text{FIFO}} \times R_{\text{FIFO-MUX-OUT}})^6) \times \\ (1-(1-R_{\text{LBDR-MUX-IN}} \times R_{\text{LBDR}} \times R_{\text{LBDR-MUX-OUT}})^6) \end{aligned} \tag{11}$$

$$\begin{aligned} R_{OC} = (1-(1-R_{\text{ARBI-MUX-IN}} \times R_{\text{ARBI}} \times R_{\text{ARBI-MUX-OUT}})^6) \times \\ (1-(1-R_{\text{XBAR-MUX-IN}} \times R_{\text{XBAR}} \times R_{\text{XBAR-MUX-OUT}})^6) \end{aligned} \tag{12}$$

$$R_{\text{Stay-Alive}} = R_{IC} \times R_{OC} \tag{13}$$

The reliability diagrams for Turn Priority and Stay Alive architectures are demonstrated in Fig. 7c and Fig. 7d, respectively.

D. Unit-Duplication

For a fair comparison, it is necessary to compare design with Unit-Duplication. Fig. 8 depicts the block diagram of Unit-Duplication architecture used for this comparison. It is assumed that fault information is provided for each unit and it is used for healthy unit selection. The architecture applies this method to every unit in the router. Reliability of input and output channels are shown on equations : 14, 15. Hence, the probability of Survival of at least one turn in the original router architecture is shown on equation 16.

$$\begin{aligned} R_{IC} = ((1-(1-R_{\text{FIFO}}) \times (1-R_{\text{FIFO}})) \times R_{\text{FIFO-MUX}}) \\ ((1-(1-R_{\text{LBDR}}) \times (1-R_{\text{LBDR}})) \times R_{\text{LBDR-MUX}}) \end{aligned} \tag{14}$$

$$\begin{aligned} R_{OC} = ((1-(1-R_{\text{ARBI}}) \times (1-R_{\text{ARBI}})) \times R_{\text{ARBI-MUX}}) \\ ((1-(1-R_{\text{XBAR}}) \times (1-R_{\text{XBAR}})) \times R_{\text{XBAR-MUX}}) \end{aligned} \tag{15}$$

$$R_{\text{Unit-Duplication}} = R_{IC} \times R_{OC} \tag{16}$$

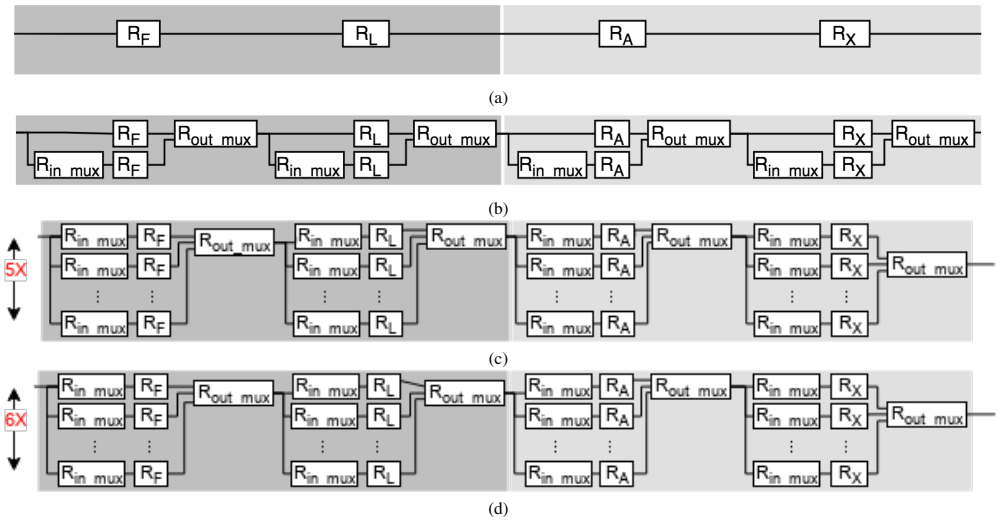


Fig. 7. Reliability diagram of proposed architectures: a)Baseline router, b)Single Spare, c)Turn Priority d)Stay Alive.

E. TMR

Two TMR architectures are implemented with regular voter (see Fig. 9). Unit TMR, that triplicates each unit in the router; and system TMR architecture, that triplicates the whole router. It is worth noting that TMR is inherently different compared to the proposed architectures due to its ability to detect faults. However, it is provided as reference.

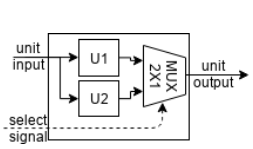


Fig. 8. Unit Duplication

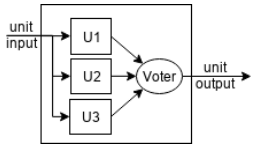


Fig. 9. Triple Modular Redundancy

In unit TMR, similar to Unit Duplication, the reliability of input and output channels are shown on equation 17, 18. And accordingly, the probability of the whole router, having single turn available is shown on equation 19.

$$R_{IC} = ((3 \times R^2_{\text{FIFO}} - 2 \times R^3_{\text{FIFO}}) \times R_{\text{FIFO-MUX}}) \\ ((3 \times R^2_{\text{LBDR}} - 2 \times R^3_{\text{LBDR}}) \times R_{\text{LBDR-MUX}}) \tag{17}$$

$$R_{OC} = ((3 \times R^2_{\text{ARBI}} - 2 \times R^3_{\text{ARBI}}) \times R_{\text{ARBI-MUX}}) \\ ((3 \times R^2_{\text{XBAR}} - 2 \times R^3_{\text{XBAR}}) \times R_{\text{XBAR-MUX}}) \tag{18}$$

$$R_{\text{Unit-TMR}} = R_{IC} \times R_{OC} \tag{19}$$

On the other hand reliability of system TMR is using original router for its modules, thus having same formula for router, equations : 20, 21, 22. And additionally, equation of reliability after applying TMR is shown on equation: 23

$$R_{IC} = R_{\text{FIFO}} \times R_{\text{LBDR}} \tag{20}$$

$$R_{OC} = R_{\text{ARBI}} \times R_{\text{XBAR}} \tag{21}$$

$$R_{\text{single-router}} = R_{IC} \times R_{OC} \tag{22}$$

$$R_{\text{System-TMR}} = (3 \times R^2_{\text{single-router}} - 2 \times R^3_{\text{single-router}}) \times R_{voter} \tag{23}$$

F. Reliability Discussion

In this work, it is assumed that probability of failure of units is proportional to unit's cell area and the probability of failure of all units are normalized against FIFO. For example, in the case of the original version of the router if the probability of failure in FIFO is 0.5 while its area is 14315, then, the probability of failure in Arbiter with an area of 860, would be: 0.5 * (860/14315) = 0.03 .

In Fig. 10, X axis represents the probability of FIFO failing; The probabilities of other units are derived according to FIFO reliability. And Y axis represents the probability of survival of the router.

The reliabilities can be applied on different time frames(hour, day, month, etc.).

During this architecture space exploration, all developed architectures improved the reliability of original router. Turn Priority and Stay Alive architectures outperformed all others. Followed by Unit Duplication and then Single Spare architecture. It should be noted that Unit Duplication outperformed system TMR and unit TMR architectures. System and unit TMR architectures become worse than original router after reliability goes down to 0.7 and 0.49, respectively. Results can be seen on fig. 10.

To measure overall reliability improvement in numbers, for each value of FIFO reliability, router reliability improvement

Router Version (μm^2)	ARBITER (μm^2)	FIFO (μm^2)	LBDR (μm^2)	XBAR (μm^2)	Unit Controller(s) (μm^2)	Spare Units (μm^2)	Interconnection, % (μm^2)	Total (μm^2)	Overhead (%)
Original	860	14315	375	2334	0	0	44 803 (33.37 %)	134244	–
Single Spare	1533	15 542	913	3515	1420	21 772	67 525 (35.32 %)	191165	42
Turn Priority	2070	19 206	1351	4969	5975	0	93 694 (39.36 %)	238068	77
StayAlive	2253	19 736	1438	5472	7702	21 772	113 776 (39.48 %)	288160	115
Unit-Duplication	2040	29547	945	5526	0	0	96 871 (33.73 %)	287176	114
Unit TMR	2966	44108	1320	8086	0	0	143 797 (33.74 %)	426220	217
System TMR	860	14315	375	2334	0	0	143 574 (34.86 %)	411897	207

TABLE I

AREA RESULTS FOR INTERCONNECTION AND ITS AREA PERCENTAGE OF CORRESPONDING ROUTER, AREA OF ROUTER UNITS INCLUDING INTEGRATED VOTERS OR MULTIPLEXERS, AND AREA OVERHEAD PERCENTAGE OF ALL ROUTERS - COMPARED TO BASELINE ROUTER(ORIGINAL)

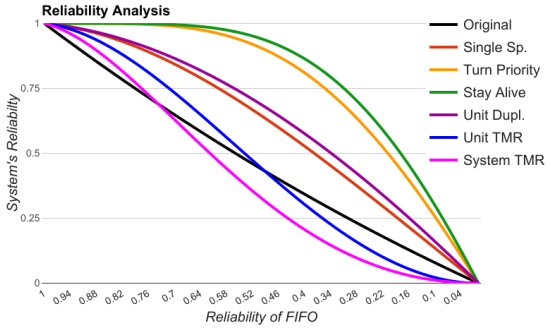


Fig. 10. Reliability of routers. Probability that at least single turn will be functional.

factor is calculated, then summed up and averaged to the number of samples, this equation is shown below.

$$R_{overall} = \frac{1}{n} * \sum_{i=1}^{n} R_{developed}[i]/R_{original}[i] \quad (24)$$

The results of this calculations are displayed on table II. The numbers are averaged on the whole range: [1:0] since it is preferable to generalize the results.

Architecture Version	Reliability in range [1:0]
Original	1
Single Spare	1.432
Turn Priority	2.118
Stay Alive	2.298
Unit Duplication	1.562
Unit TMR	0.842
System TMR	0.672

TABLE II

RELIABILITY IMPROVEMENTS OF ARCHITECTURES. DEVELOPED ARCHITECTURES ARE MARKED WITH YELLOW

VI. AREA COMPARISON

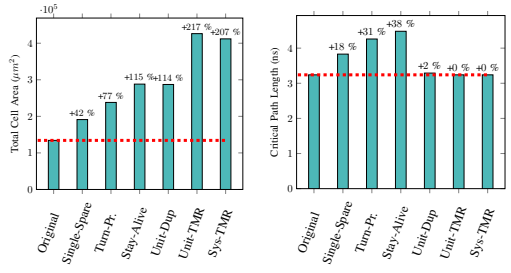
Different architectures are synthesized using Synopsys Design Vision 2015 [24] along with AMS 180 nm technology library [25]. In figure 11 the total cell area and critical path delay of all versions are compared to Unit-Duplication (with an assumption of an existence of fault detection mechanism) and TMR. The router is synthesized using 10ns clock period. The increase in the critical path, which was due to insertion of multiplexers between units, does not violate the original timing constraints. The critical path of the original router which was 3.24ns, was increased in the Single Spare router to 3.83ns, in Turn Priority to 4.26ns and in Stay Alive to 4.48ns. Since the clock period is not changed and no extra pipeline stage is added to the router, the latency and throughput of the network do not change. This was verified by simulations using Questasim 2015 simulator [26].

Detailed synthesis results are presented in table : I. In the table, areas of different units are listed for all architectures, note that areas of multiplexers are included in the units. For Unit-Duplication and unit TMR, the voter and multiplexer area is included in the unit area. The last column lists the total area overhead compared to the original router. The Single Spare version has four small Unit Control units, each one of them having an area of 355. In other two versions, there were single Unit Control units having significantly larger areas, while providing more logic and control over units.

According to table I, multiplexers contribute significantly to the area overhead. For example in Stay Alive version, the area of Arbiter is 3.45 times larger than the original arbiter because of multiplexers.

VII. CONCLUSION

In this paper a new metric for comparing the reliability of NoC router architectures is introduced. This metric derives reliability value of router based on the areas of the units in contrast to SPF which does not take into account the areas for assessing the reliability. Also, this work proposes three novel architectures which use transparent resource allocation to improve the reliability of the router. Also, a comparison of the



(a) Total area (b) Critical path in nanoseconds

Fig. 11. Total area and critical path comparison for all versions of router.

proposed architectures in terms of reliability improvement and area overhead with respect to the baseline, Unit-Duplication and TMR architectures is provided. Developed architectures improved the reliability of the router 1.4-2.9 times with area overhead 42%-115%. As for future work, it is planned to develop an architecture, implementing channel multiplexing instead of unit multiplexing and applying multiple spares.

REFERENCES

[1] 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.

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

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

[4] E. Dubrova, *Fault-Tolerant Design*, 1st ed. 978-1-4614-2113-9: Springer-Verlag New York, 2013.

[5] P. Poluri and A. Louri, "Shield: A reliable network-on-chip router architecture for chip multiprocessors," *IEEE Transactions on Parallel and Distributed Systems*, vol. PP, no. 99, pp. 1–1, 2016.

[6] M. R. Kakoee, V. Bertacco, and L. Benini, "Relinoc: A reliable network for priority-based on-chip communication," in *2011 Design, Automation Test in Europe*, March 2011, pp. 1–6.

[7] C. Grecu, A. Ivanov, R. Saleh, and P. P. Pande, "Noc interconnect yield improvement using crosspoint redundancy," in *2006 21st IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, Oct 2006, pp. 457–465.

[8] X. Zhang, M. Ebrahimi, L. Huang, and G. Li, "Fault-resilient routing unit in nocs," in *2015 28th IEEE International System-on-Chip Conference (SOCC)*, Sept 2015, pp. 164–169.

[9] C. Concatto, D. Matos, L. Carro, F. Kastensmidt, A. Susin, E. Cota, and M. Kreutz, "Fault tolerant mechanism to improve yield in nocs using a reconfigurable router," in *Proceedings of the 22Nd Annual Symposium on Integrated Circuits and System Design: Chip on the Dunes*, ser. SBCCI '09. New York, NY, USA: ACM, 2009, pp. 26:1–26:6. [Online]. Available: http://doi.acm.org/10.1145/1601896.1601929

[10] A. Eghbal, P. M. Yaghini, H. Pedram, and H. R. Zarandi, "Designing fault-tolerant network-on-chip router architecture," *International Journal of Electronics*, vol. 97, no. 10, pp. 1181–1192, 2010.

[11] K. Constantinides, S. Plaza, J. Blome, B. Zhang, V. Bertacco, S. Mahlke, T. Austin, and M. Orshansky, "Bulletproof: a defect-tolerant cmp switch architecture," in *The Twelfth International Symposium on High-Performance Computer Architecture, 2006.*, Feb 2006, pp. 5–16.

[12] T. Lehtonen, P. Liljeberg, and J. Plosila, "Online reconfigurable self-timed links for fault tolerant noc," *VLSI design*, vol. 2007, 2007.

[13] D. Park, C. Nicopoulos, J. Kim, N. Vijaykrishnan, and C. R. Das, "Exploring fault-tolerant network-on-chip architectures," in *International Conference on Dependable Systems and Networks (DSN'06)*, June 2006, pp. 93–104.

[14] C. Li, M. Yang, and P. Ampadu, "An energy-efficient noc router with adaptive fault-tolerance using channel slicing and on-demand tmr," *IEEE Transactions on Emerging Topics in Computing*, vol. PP, no. 99, pp. 1–1, 2016.

[15] Y. Ren, L. Liu, S. Yin, Q. Wu, S. Wei, and J. Han, "A vlsi architecture for enhancing the fault tolerance of noc using quad-spare mesh topology and dynamic reconfiguration," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, May 2013, pp. 1793–1796.

[16] D. Fick, A. DeOrio, J. Hu, V. Bertacco, D. Blaauw, and D. Sylvester, "Vicis: A reliable network for unreliable silicon," in *2009 46th ACM/IEEE Design Automation Conference*, July 2009, pp. 812–817.

[17] M. Shahiri, M. Valinataj, and A. M. Rahmani, "A reliable and high-performance network-on-chip router through decoupled resource sharing," in *2016 International Conference on High Performance Computing Simulation (HPCS)*, July 2016, pp. 88–95.

[18] J. Kim, C. Nicopoulos, and D. Park, "A gracefully degrading and energy-efficient modular router architecture for on-chip networks," in *33rd International Symposium on Computer Architecture (ISCA'06)*, 2006, pp. 4–15.

[19] M. Koibuchi, H. Matsutani, H. Amano, and T. M. Pinkston, "A lightweight fault-tolerant mechanism for network-on-chip," in *Second ACM/IEEE International Symposium on Networks-on-Chip (nocs 2008)*, April 2008, pp. 13–22.

[20] Siavoosh Payandeh Azad,Behrad Niazmand, Karl Janson, Nevin George, Adeboye Stephen Oyeniran, Tsotne Putkaradze, Apneet Kaur, Jaan Raik, Gert Jervan, Raimund Ubar, Thomas Hollstein, "From Online Fault Detection to Fault Management in Network-on-Chips: A Ground-Up Approach." [Online]. Available: https://github.com/Project-Bonfire/EHA

[21] J. Flich and J. Duato, "Logic-Based Distributed Routing for NoCs." [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4407676

[22] 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.

[23] 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)*, April 2017, pp. 48–53.

[24] (1994) Synopsys design compiler. http://www.synopsys.com.

[25] (2016) AMS 0.18um CMOS process. http://ams.com/eng/Products/Full-Service-Foundry/Process-Technology/CMOS/0.18-m-CMOS-process/.

[26] The questa advanced simulator. https://www.mentor.com/products/fv/questa/.

CURRICULUM VITAE

Personal data

Name:	Siavoosh Payandeh Azad
Date of birth:	June 30 1986

Contact data

Address:	ICT 509, 15A Akadeemia St., 12618 Tallinn, Estonia
Phone:	+372-6202267
E-mail:	siavoosh.azad@ttu.ee

Education

2014 – 2018:	Tallinna Tehnikaülikool	PhD studies
2010 – 2012:	Kungliga Tekniska Högskolan	M.Sc. in System on Chip
2004 – 2009:	Shahid Beheshti University	B.Sc. in Electrical Engineering

Professional employment

2014 – 2018:	Tallinn University of Technology	Early-stage researcher

ELULOOKIRJELDUS

Isikuandmed

Nimi:	Siavoosh Payandeh Azad
Sünniaeg:	30.06.1986

Kontaktandmed

Aadress:	ICT 509, 15A Akadeemia Tee, 12618 Tallinna, Eesti
Telefon:	+372-6202267
E-post:	siavoosh.azad@ttu.ee

Hariduskäik

2014 – 2018:	Tallinna Tehnikaülikool – doktorikraad
2010 – 2012:	Kungliga Tekniska Högskolan – tehnikateaduse magisterl
2004 – 2009:	Shahid Beheshti University – tehnikateaduse bakalaureus

Teenistuskäik

2014 – 2018	nooremteadur, arvutisüsteemide instituut, Tallinna Tehnikaülikool