FAULT TOLERANT AND ADAPTIVE SYSTEMS WITH FOCUS ON NETWORKS-ON-CHIPS

A Dissertation
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Hamed Sajjadi Kia

In Partial Fulfillment
for the Degree of
DOCTOR OF PHILOSOPHY

Major Department:
Electrical and Computer Engineering

February 2014

Fargo, North Dakota

North Dakota State University
Graduate School

Title

Fault tolerant and adaptive systems with focus on networks on chip

By

Hamed Sajjadi Kia

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

DOCTOR OF PHILOSOPHY

SUPERVISORY COMMITTEE:

Dr. Sudarshan Srinivasan
Chair

Dr. Cristinel Ababei

Dr. Samee Khan

Dr. Rui Dai

Approved:

06/03/2014	Dr. Scott C. Smith
Date	Department Chair

ABSTRACT

The first step in design of reliable systems is the ability to evaluate the reliability of the system. This is an important step in the process of designing reliable systems because design techniques that proactively improve the lifetime reliability of systems on chip (SoC) require some form of redundancy. The cost (area, power, and design time overheads) associated with such redundancy makes developing systems with resilience to all types of failure mechanisms impractical. To address this problem we developed an accurate reliability evaluation algorithm that is capable of identifying the vulnerable subblocks of the system. The proposed reliability evaluation methodology can also be utilized to develop a new lifetime aware floorplanning strategy that is capable of identifying the most reliable floorplan for a given design. We consider this an essential step toward a design approach where reliability is a primary objective.

Recent advances in CMOS technology and integration of multiple processing elements in a single chip has also made the on chip communication a challenge in design of multi-processor SoCs (MPSoC). Networks on Chip (NoC) has been introduced as a new communication medium in response to the rising need for the new communication structure for MPSoCs. While NoC is proven to be an efficient communication structure for SoC, the same failure mechanisms and processing faults that have adverse effects on processing elements can also render NoC inoperable. We proposed a new multi layered reliable design methodology for NoCs as a hybrid solution composed of multiple layers of fault tolerant design techniques to address this challenge. The proposed structure for NoCs can address hard failures across three levels of abstraction. In first layer (software layer), we use a reliability aware mapping algorithm to assign application tasks on NoC such that network reliability is improved. In second and third layers (architecture and network-routing layers), we design an NoC architecture that uses self-repairable links and a distributed routing. The combination of these techniques in the proposed layered approach helps to provide a better performance and tradeoff between the improvement in reliability and cost due to the required redundancy and extra logic.

ACKNOWLEDGMENTS

I am grateful to acknowledge and thank all of those who assisted me during my graduate studies at North Dakota State University. First, I would like to thank Dr. Cristinel Ababei and Dr. Sudarshan Srinivasan my academic advisors. Their guidance, support, and patience throughout my years as a graduate student are truly appreciated. Special thanks go to Dr. Samee Khan and Dr. Rajesh Kavasseri for their support and encouragement throughout my studies. Also special thanks are given to my other graduate committee member, Dr. Rui Dai. I would also like to thank Dr. Hyunsook Do for her help during my proposal defense. Finally, I would like to thank my family for their patience and unceasing support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

NoC	Networks on Chip
SoC	Systems on Chip
TDDB	Time Dependent Dielectric Breakdown
NBTI	Negative Bias Temperature Instability
TC	Thermal Cycling
EM	Electromigration
SM	Stress Migration
MC	Monte Carlo
MTTF	Mean Time To Failure
RAMP	Reliability Aware MicroProcessor
SOFR	Sum-Of-Failure-Rates
FaRBS	Failure Rate Based Spice
MaCRO	Maryland Circuit Reliability Oriented
CTG	Communication Task Graph
MPSoC	Multi-Processor Systems on Chip

CHAPTER 1. INTRODUCTION

1.1. Motivation

Continuous downscaling of integrated circuit technology has led to new reliability challenges. The increased effects of aging mechanisms and processing faults have caused performance degradation in sub-micron integrated circuits. Therefore, lifetime reliability has become a fundamental challenge in the design of Systems on Chip (SoCs). To maintain downscaling benefits, SoCs need to be designed with built in resilience techniques. More importantly, reliability must become one of the main objectives across the system design process [1.2]–[1.4]. The first step in doing so is to evaluate system reliability. Evaluation of reliability can be a challenging task because it is affected by numerous factors including aging or wearout mechanisms (e.g., time-dependent dielectric breakdown (TDDB) [1.5], negative bias temperature instability (NBTI) [1.6], [1.7], electromigration (EM) [1.8], [1.9], thermal cycling (TC), and stress migration (SM) [1.10]), process variations, dynamic power and thermal management, workload conditions, and system architecture and configuration. The information acquired by accurately evaluating the reliability of the different components of the system can enable system designers to focus their efforts on failure mechanism specific techniques and on the reliability crucial blocks.

In this dissertation we focus on the reliability challenge that we are facing in design of sub-micron integrated circuits with focus on the networks on chip (NoC) which forms the communication structure for multiprocessor SoC (MPSoCs) [1.11], [1.12]. Fault tolerance is one of the oldest resilience areas [1.13], [1.14] and relies on redundancy as a technique to compensate for the random failure of components to improve reliability [1.15]. However the cost associated with different fault tolerant techniques makes hardening SoC against all failure mechanisms impractical. As a solution to this challenge we propose a new accurate reliability evaluation mechanism that is capable of identifying the reliability crucial blocks and the failure mechanism that has the most adverse effect on lifetime reliability of a given system. The proposed algorithm can also be utilized in developing a lifetime aware floorplanning algorithm that is capable of identifying the most reliable floorplan for the system.

NoC reliability is a growing challenge in the design of MPSoCs [1.16]. Previous work has focused mainly on processing elements or the computation units of MpSoCs. They address reliability by employing fault resilient techniques based on error detection [1.17], failure prediction [1.18], and error masking [1.19]. While in the field of computer networks there has been a lot of work done on reliability, there have been only few recent attempts to estimate or indirectly optimize NoC reliability. The organization in layers of NoCs resembles the open system interconnection (OSI) protocol stack. Therefore, it is convenient to discuss fault tolerant design techniques for NoCs in association with this organization. At the physical layer, wires may be subject to delay variations [1.20], while routers may be impacted by single event upsets (SEUs) [1.21]. The data link layer can provide the functional and procedural means to detect and possibly correct errors that may occur in the physical layer, by employing error correcting codes (ECCs) [1.22]–[1.24], data encoding [1.25], [1.26], and redundancy based reconfiguration [1.27]. At the network layer, reliability of the routing algorithms can be enhanced by routing multiple copies of the same packet via multiple paths [1.28]–[1.31] or by adaptive re-routing [1.32]. The system software provides an abstraction of the underlying hardware platform, which can be leveraged by the application developer to effectively exploit the hardware's capabilities via reconfiguration [1.33].

In this dissertation we propose a new multi layered fault-tolerance oriented design methodology for NoCs as a hybrid solution composed of several reliability and fault tolerance design techniques applied at the CAD tool, NoC architecture, and network-routing levels. It is this layered approach, which combines the benefits of the proposed techniques at different levels that improves NoC resilience significantly. Under our proposed approach it will take a very large number of hard faults to render NoC inoperable. What makes the proposed method even more powerful is that it is very cost effective in terms of area and power overheads.

1.2. Dissertation outline

Chapter 2 demonstrates a new circuit level vulnerability and reliability evaluation methodology and its application in developing a lifetime aware floorplanning strategy. This methodology is based on a divide and conquer approach, and enjoys the benefits of transistor level accuracy and of block level efficiency. At the core of the lifetime estimation engine lies a Monte Carlo algorithm which works with failure times modeled as Weibull and lognormal distributions for several aging mechanisms including time dependent dielectric breakdown, negative bias temperature instability, electromigration, thermal cycling, and stress migration.

Chapter 3 discusses a new approach to partition links in a NoC into multiple segments and use spare wires at the level of each segment to address permanent errors due to manufacturing or wearout defects. Because different segments of the spare wires address different errors from different segments, the proposed reconfigurable link structure can tolerate a larger number of errors with a reduced number of spare wires. Experimental results on area, power consumption, delay, and reliability show that the optimal link is achieved when the link is partitioned into two segments.

Chapter 4 presents a new fault-tolerant and congestion-aware adaptive routing algorithm for NoCs. The proposed algorithm is based on the ball-and-string model and employs a distributed approach based on partitioning of the regular NoC architecture into regions controlled by local monitoring units. Experimental results based on an actual Verilog implementation demonstrate that the proposed adaptive routing algorithm improves significantly the network throughput.

Chapter 5 formulates the problem of energy consumption and reliability oriented application mapping on regular Network-on-Chip topologies and presents a multi layered approach to design fault resilient NoC. Simulation results demonstrate that reliability can be improved significantly without sacrificing much of energy consumption.

Chapter 6 provides a summary, conclusion, and outlook based on this dissertation.

1.3. References

[1.1] S. Borkar, “Designing reliable systems from unreliable components: the challenges of transistor variability and degradation,” IEEE Micro, vol. 25, no. 6, pp. 10-16, Nov. 2005.

[1.2] V. Raghunathan, M.B. Srivastava, and R.K. Gupta, “A survey of techniques for energy efficient on-chip communication,” ACM/IEEE Design Automation Conference (DAC), 2003.

[1.3] A. DeHon, H.M. Quinn, and N.P. Carter, “Vision for cross-layer optimization to address the dual challenges of energy and reliability,” ACM/IEEE Design, Automation and Test in Europe Conference (DATE), 2010.

[1.4] S. Mitra, K. Brelsford, and P.N. Sanda, “Cross-layer resilience challenges: metrics and optimization,” ACM/IEEE Design, Automation and Test in Europe Conference (DATE), 2010.

[1.5] J.H. Stathis, “Reliability limits for the gate insulator in CMOS technology,” IBM Journal of Research and Development, vol. 46, 2002.

[1.6] 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, July 2003.

[1.7] R. Vattikonda, W. Wang, and Y. Cao, "Modeling and minimization of PMOS NBTI effect for robust nanometer design," ACM/IEEE Design Automation Conference (DAC), 2006.

[1.8] S.M. Alam, C.L. Gan, D.E. Troxel, and C.V. Thompson, "Circuit-level reliability analysis of Cu interconnects," International Symposium on Quality Electronics Design (ISQED), 2004.

[1.9] Z. Lu, J. Lach, M.R. Stan, and K. Skadron, "Temperature-aware modeling and banking of IC lifetime reliability," IEEE Micro, vol. 25, no. 6, pp. 40-49, Nov./Dec. 2005.

[1.10] Failure mechanisms and models for semiconductor devices, JEDEC Publication JEP122E, 2009.

[1.11] W.J. Dally, B.P. Towles, "Principles and Practices of Interconnection Networks," Morgan Kaufmann, 2004.

[1.12] G. De Micheli and L. Benini, "Networks on Chip," Morgan Kaufmann, 2006.

[1.13] J. Von Neumann, "Probabilistic logic and the synthesis of reliable organizms from unreliable components," Automata Studies, C.E. Shannon and J. McCarthy Eds., Princeton Univ. Press, 1956.

[1.14] E. Moore and C.E. Shannon, "Reliable circuits using less reliable relays," Journal of the Franklin Institute, 1956.

[1.15] J.P.G. Sterbenz, D. Hutchison, E. Cetinkaya, A. Jabbar, J.P. Rohrer, M. Scholler, and P. Smith, "Resilience and survivability in communication networks: strategies, principles, and survey of disciplines," Computer Networks, 2010.

[1.16] J.D. Owens, W.J. Dally, R. Ho, D.N. Jayasimha, S.W. Keckler, and L.S. Peh, “Research challenges for on-chip interconnection networks,” IEEE Micro, vol. 27, no. 5, Sept.-Oct. 2007.

[1.17] S. Feng, S. Gupta, A. Ansari, and S. Mahlke, “Shoestring: probabilistic soft error reliability on the cheap,” ASPLOS, 2010.

[1.18] Y. Li, Y.M. Kim, E. Mintarno, D.S. Gardner, and S. Mitra, “Overcoming early-life failure and aging for robust systems,” IEEE Design & Test of Computers, 2009.

[1.19] M. Choudhury, V. Chandra, K. Mohanram, R. Aitken, “TIMBER: time borrowing and error relaying for online timing error resilience,” ACM/IEEE Design, Automation and Test in Europe Conference (DATE), 2010.

[1.20] C. Hernandez, F. Silla, J. Duato, “A methodology for the characterization of process variation in NoC links,” ACM/IEEE Design, Automation and Test in Europe Conference (DATE), 2010.

[1.21] A. Ejlali, B.M. Al-Hashimi, P. Rosinger, and S.G. Miremadi, “Joint consideration of fault-tolerance, energy-efficiency and performance in on-chip network,” ACM/IEEE Design, Automation and Test in Europe Conference (DATE), 2007.

[1.22] H. Zimmer and A. Jantsch, “A fault model notation and error control scheme for switch-to-switch buses in a Network-on-Chip,” International Conference on, Hardware/Software Codesign and System Synthesis, 2003.

[1.23] S.R. Sridhara and N.R. Shanbhag, “Coding for system-on-chip networks: a unified framework,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 13, no. 6, June 2005.

[1.24] S. Murali, T. Theocharides, N. Vijaykrishan, M.J. Irwin, L. Benini, and G. De Micheli, “Analysis of error recovery schemes for Networks on Chips,” IEEE Design &Test of Computers, 2005.

[1.25] D. Bertozzi, L. Benini, and G. De Micheli, “Error control schemes for on-chip communication links: the energy-reliability trade-off,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 24, no. 6, June 2005.

[1.26] R. Singhal, G. Choi, and R. Mahapatra, “Information theoretic approach to address delay and reliability in long onchip interconnects,” ACM/IEEE EEE/ACM International Conference on Computer-Aided Design, 2006.

[1.27] T. Lehtonen, D. Wolpert, P. Liljeberg, J. Plosila, and P. Ampadu, “Self-adaptive system for addressing permanent errors in on-chip interconnects,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 18, no. 4, 2010.

[1.28] M. Pirretti, G.M. Link, R.R. Brooks, N. Vijaykrishnan, M.T. Kandemir, and M.J. Irwin, “Fault tolerant algorithms for Network-On-Chip interconnect,” IEEE Computer society Annual Symposium on VLSI, 2004.

[1.29] S. Manolache, P. Eles, and Z. Peng, “Fault and energy-aware communication mapping with guaranteed latency for applications implemented on NoC,” ACM/IEEE Design Automation Conference (DAC), 2005.

[1.30] S. Murali, D. Atienza, L. Benini, and G. De Micheli, “A multi-path routing strategy with guaranteed in-order packet delivery and fault-tolerance for networks on chip,” ACM/IEEE Design, Automation and Test in Europe Conference (DATE), 2006.

[1.31] A. Patooghy and S.G. Miremadi, “Complement routing: a methodology to design reliable routing algorithm for network on chips,” Microprocessors and Microsystems, 2010.

[1.32] A.D. Choudhury, G. Palermo, C. Silvano, and V. Zaccaria, "Yield enhancement by robust application-specific mapping on Network-on-Chips," International Workshop on Network on Chip Architectures, 2009.

[1.33] L. Zhang, Y. Yu, J. Dong, Y. Han, S. Ren, and X. Li, "Performance asymmetry-aware topology virtualization for defect-tolerant NoC-based many-core processors," ACM/IEEE Design, Automation and Test in Europe Conference (DATE), 2010.

CHAPTER 2. A NEW RELIABILITY EVALUATION METHODOLOGY WITH APPLICATION TO LIFETIME ORIENTED CIRCUIT DESIGN

This paper was presented in IEEE transactions on device and materials reliability, vol. 13, no. 1, March 2013. The authors of the paper are Hamed Sajjadi Kia and Cristinel Ababei. The preliminary results of this research were published in a paper titled: A new reliability evaluation methodology and its application to network-on-chip routers, in IEEE/IFIP international conference on VLSI and System-on-Chip (VLSI-SoC), 2012.

2.1. Abstract

We propose a new circuit-level vulnerability and reliability evaluation methodology and utilize it to develop a lifetime aware floorplanning strategy. Our work is motivated by increasingly adverse aging failure mechanisms, which have made reliability a growing fundamental challenge in the design of integrated circuits. Because the proposed methodology is based on a divide-and-conquer approach, it enjoys the benefits of transistor level accuracy and of block-level efficiency. At the core of the lifetime estimation engine lies a Monte Carlo algorithm which works with failure times modeled as Weibull and lognormal distributions for several aging mechanisms including time-dependent dielectric breakdown, negative bias temperature instability, electromigration, thermal cycling, and stress migration. To demonstrate the value of the proposed reliability evaluation methodology and floorplanning strategy, we apply them to a network-on-chip router design example. The new floorplanning approach is able to find floorplans with up to 15% difference in the lifetime of the router design. In addition, the proposed reliability evaluation methodology identifies the routing computation and virtual channel allocation units as the most vulnerable subblocks of the design. Such information can be

very useful to designers to predict circuit and system mean time to failure and to focus on cost-effective design techniques targeted at specific parts of the design to improve its lifetime.

2.2. Introduction

Reliability has become a growing fundamental challenge in the design of integrated circuits due to increasingly adverse aging failure mechanisms that can cause performance degradation and eventual device and system failure [2.1]. To maintain downscaling benefits, increasingly complex integrated circuits must be designed with built-in resilience techniques [2.2]–[2.4]. To achieve that, one of the main difficulties is to evaluate reliability. Evaluation of reliability is a challenging task because reliability is affected by numerous factors including aging or wearout mechanisms [2.5] (e.g., time-dependent dielectric breakdown (TDDB) [2.6], negative bias temperature instability (NBTI) [2.7]–[2.9], electromigration (EM) [2.10], [2.11], thermal cycling (TC), and stress migration (SM) [2.12]), process variations, dynamic power and thermal management, workload conditions, and system architecture and configuration.

2.2.1. Related work

2.2.1.1. Reliability evaluation techniques

While there has been significant work carried out to estimate reliability [2.14]–[2.24], we discuss next two approaches that are related to our work. An extensive review of previous reliability simulation tools can be found in [2.25]. The RAMP approach [2.15] models the mean time to failure (MTTF) of a processor microarchitecture as a function of temperature related failure rates of individual structures on chip. It divides the processor into several discrete structures (e.g., ALU, register files, etc) and applies analytical models to each structure. Then, it combines the structure level MTTFs to compute the overall MTTF of the entire processor assumed as a series failure system. Because the lifetime distributions of failure mechanisms are assumed to be exponential [2.16], the reliability is calculated by applying the sum-of-failure-rates (SOFR) model. This approach is not realistic because failure rates of units increase with time due to aging. To address this limitation of the SOFR model, RAMP 2.0 [2.17], [2.26] uses lognormal distributions, which are harder to deal with analytically. One of the main limitations of the RAMP approach as an architecture level approach is its accuracy. In addition, it may estimate equal MTTFs for blocks of different sizes but with activity factors that cancel out the area proportionality factor. Another more recent class of simulation based reliability evaluation approaches are based on Spice simulations. Failure rate based Spice (FaRBS) [2.27] and Maryland circuit reliability oriented (MaCRO) [2.29], [2.30] are circuit-level reliability simulation methods. Both of these methods utilize degradation models for TDDB, NBTI, and hot carrier degradation (HCD). They are based on a series of accelerated lifetime models and failure equivalent circuit models for these wearout mechanisms [2.25], [2.32]. They employ Spice to calculate electrical parameters of fresh and degraded devices and to predict their degradation or failure from these parameters [2.27]. The main advantage of this class of simulation methods is the device level granularity that enables reliability analysis at transistor level to identify the most vulnerable transistors. There are some issues related to the Spice based reliability simulation. These approaches do not consider the layout of the system and simulations are done under worst case temperature scenarios, which is not realistic. Besides, Spice circuit simulations tend to take long time especially when done for large circuits. In addition, both methods (FaRBS and MaCRO) are developed under the assumption that failure rate is constant. As discussed above this assumption is inaccurate.

2.2.1.2. Floorplanning

Floorplanning is an important step during the design of integrated circuits. Because the relative locations of different subblocks are decided during floorplanning, the overall temperature profile of the chip is directly affected by the quality of the floorplanning step. As such, there has been significant work done on the problem of thermal aware floorplanning [2.33]–[2.39]. Even though reliability is directly related to temperature, it has been significantly less investigated. Nevertheless, a reliability aware voltage island partitioning and floorplanning algorithm for SoC is reported in [2.40]. The algorithm considers the sensitivity of the SoC to soft errors and does not address aging mechanisms. The authors of [2.41] define reliability in terms of supply voltage noise margin and propose a floorplanning algorithm that distributes the thermal profile evenly and reduces the power supply noise. The effect of temperature on the probability of errors in SRAM memories is discussed in [2.42], where a leakage aware floorplanner is introduced. Currently, there is no aging failure mechanisms aware floorplanning method reported in the literature.

2.2.2. Contribution

To address the limitations of previous reliability evaluation methods, we propose a new circuit-level reliability evaluation methodology. To this end, our main contribution is as follows: 1) We propose and implement a new divide-and-conquer-based reliability evaluation methodology. Its core engine employs a Monte Carlo algorithm, which works with failure times modeled realistically as Weibull and lognormal distributions for five different aging failure mechanisms: TDDB, NBTI, EM, TC, and SM. Hence, our results are more accurate and realistic compared to previous works that are based on the assumption that lifetime distributions are exponential. 2) We utilize the proposed reliability evaluation methodology to develop a new lifetime aware floorplanning strategy that is capable of identifying the most reliable floorplan for a given design. We consider this an essential step toward a design approach where reliability is also a primary objective. To demonstrate the usefulness of the proposed algorithms, we apply them to an NoC router as a design example. We analyze its reliability, identify its most vulnerable subblocks, and generate the most reliable floorplan for it.

2.3. Lifetime failure models

2.3.1. Importance of lifetime distribution of failure mechanisms

Many proposed lifetime reliability models assume a uniform device density on the chip and an identical vulnerability of devices to failure mechanisms [2.14]. The lifetime distributions of failure mechanisms are usually assumed to be exponential [2.15], [2.16], [2.18], [2.26], [2.43]. As discussed in the previous section, this allows system-level reliability to be calculated by applying the sum-of-failure-rates (SOFR) model. However, this approach is not realistic because failure rates of units increase with time due to aging. To address this issue and to develop an accurate reliability model, more general lifetime distributions (e.g., Weibull and lognormal) must be utilized. On the other hand, when Weibull or lognormal distributions are utilized the prediction of reliability becomes more difficult and therefore Monte Carlo simulations must be employed [2.16], [2.26], [2.43]. In this paper, we adopt Weibull distribution modeling for TDDB, NBTI, TC, and SM and lognormal distribution modeling for EM because these distributions have been found to best fit the corresponding wearout mechanisms [2.12].

2.3.2. Time dependent dielectric breakdown (TDDB)

Time-dependent dielectric breakdown is caused by formation of a conducting path through the gate oxide to substrate due to electron tunneling current. TDDB has become increasingly severe as the thickness of the gate oxide decreased due to continuous technology downscaling. Under the same stress conditions, devices can feature directly hard breakdown or several soft breakdown events before the final hard breakdown [2.31]. While in this paper we utilize a recently proposed model [2.32], the proposed reliability evaluation methodology is flexible and can be changed by replacing (1) with different models as they are discovered.

2.3.2.1. TDDB lifetime model

The model for $\mathrm{MTTF}_{\mathrm{TDDB}}$ is described by the following expression [2.32]:

$$MTTF_{TDDB} \propto \left(\frac{1}{A}\right)^{\frac{1}{\beta}}(F)^{\frac{1}{\beta}}V_{gs}^{a+bT}e^{\left(\frac{c}{T}+\frac{d}{T^2}\right)} \tag{2.1}$$

where A is the transistor's gate oxide area, β is the Weibull slope parameter, F is cumulative failure percentile, T is temperature, and V_{gs} is gate source voltage of the MOSFET. Model fitting parameters a, b, c, d, β, and F are determined from experimental data. In this paper, we utilize typical values of these parameters [2.32]: $\beta = 1.2$, $F = 0.01\%$, $a = -78$, $b = 0.081$, $c = 8.81\times10^{3}$, and $d = -7.75\times10^{5}$.

2.3.3. Negative bias temperature instability (NBTI)

Negative bias temperature instability mainly affects PFETs, when they are stressed at large negative gate voltages and high temperatures. NBTI manifests as a gradual increase in the threshold voltage and consequent decrease in drain current and transconductance. The degradation exhibits logarithmic dependence on time. This effect has become more severe with technology downscaling, with the increase of the electric field applied to the gate oxide, and with the decrease of operating voltages.

2.3.3.1. NBTI lifetime model

The model for $\text{MTTF}_{\text{NBTI}}$ is described by the following expression [2.28], [2.29]:

$$MTTF_{NBTI} \propto V_{gs}^{-\frac{1}{\beta}}[\frac{1}{1+2e^{\left(-\frac{E_1}{kT}\right)}}+\frac{1}{1+2e^{\left(-\frac{E_2}{kT}\right)}}]^{-\frac{1}{\beta}} \tag{2.2}$$

where k is Boltzmann's constant, and E_1, E_2 are material and oxide electric field dependent parameters. In addition, E_2 is a voltage dependent parameter and therefore it depends on the operation of circuit. Values of E_1 and E_2 are given by:

$$E_1 = E_{it} - E_g + E_F \tag{2.3}$$

$$E_2 = E_{fx} - E_F + \gamma E_{ox}^{\frac{2}{3}} \tag{2.4}$$

where E_{it} and E_{fx} are the trap energy level at the oxide/Si interface and the trap energy in the oxide, respectively. E_F is Fermi energy, is a constant, E_{ox} is the applied electric field across the gate and can be computed as follows [2.32]:

$$E_{ox} \approx \frac{V_{gs} - 0.2V}{t_{ox}} \tag{2.5}$$

2.3.4. Electromigration (EM)

Electromigration is generally considered to be the result of momentum transfer from the electrons, which move in the applied electric field, to the ions which make up the lattice of the interconnect material. As a result, ions get dislocated from their original positions and migrate along the interconnect. Over time this phenomenon knocks a significant number of atoms far from their original positions. Failure results either from voids growing over the entire line width that cause breaking of the line or extrusions or hillocks that cause short circuits to neighboring lines.

2.3.4.1. EM lifetime model

EM has an exponential dependence on temperature. The model for $\mathrm{MTTF_{EM}}$ is based on Black's equation [2.5], [2.12] and is described by the expression below. This model is widely adopted and studied for a long time [2.13]. Its limitations depend on the probability distributions that one assumes for this failure mechanism; it is widely accepted that a lognormal distribution is more realistic [2.12]:

$$MTTF_{EM} \propto (J - J_{crit})^{-n} e^{\frac{E_{aEM}}{kT}} \tag{2.6}$$

where J is the current density in the wire, J_{crit} is the critical current density required for electromigration, E_{aEM} is the activation energy for electromigration, k is the Boltzmann's constant, and T is the absolute temperature in Kelvin. n and E_{aEM} are constants. We use 1.1 for n and 0.9 for E_{aEM} as modeled in RAMP. Notice that J is usually 2 orders of magnitude higher than J_{crit} in interconnects; hence, we approximate ($J - J_{crit} \approx J$) [2.12], [2.15].

2.3.5. Thermal cycling (TC)

Degradation due to each temperature cycle accumulates in time and can potentially lead to permanent damage. The effect is mostly seen in the package and die interface. The package is affected with two types of thermal cycles: (1) Large thermal cycles that occur a few times a day like powering up and down or going into stand-by mode. (2) Small cycles that occur a few times a second. These are due to changes in workload behavior and context switching. The effect of small thermal cycles at high frequencies has not been well studied by the packaging community, and valid models are not available. Hence, we do not consider models for the reliability impact of small thermal cycles, which is a limitation of the model that we adopt below.

2.3.5.1. TC lifetime model

The model for $MTTF_{TC}$ is described by the following expression [2.15]:

$$MTTF_{TC} \propto \left(\frac{1}{T - T_{ambient}}\right)^{q} \tag{2.7}$$

where T is the average temperature of the structure, and $T_{ambient}$ is the ambient temperature. Notice that ($T - T_{ambient}$) models the thermal cycle. q is Coffin-Manson exponent, and for the package it is equal to 2.35 [2.15].

2.3.6. Stress migration (SM)

Mechanical stress because of different thermal expansion rates of different materials in devices and circuits can lead to stress migration. This mechanical stress is proportional to the change in the temperature which is measured with respect to the stress free temperature of the metal. In general, SM is a phenomenon where the metal atoms in the interconnects migrate. It can lead to open circuit, or increased resistance.

2.3.6.1. SM lifetime model

The model for $MTTF_{SM}$ is described by the following expression [2.15]:

$$MTTF_{SM} \propto |T_0 - T|^{-n} e^{\frac{E_{aSM}}{kT}} \tag{2.8}$$

where T is the operating temperature, T_0 is the stress free temperature, n and E_{aSM} are material dependent constants. We utilize a value of 2 for n, 0.9 for E_{aSM}, and 500K for T_0 as advised in [2.5], [2.29].

2.4. Proposed reliability evaluation methodology

The block diagram with the flow chart of the proposed reliability evaluation methodology is shown in Fig. 2.1 while the corresponding pseudocode is shown in Fig. 2.2. The salient features of our methodology are as follows. First, in order to deal with complexity due to circuit size we adopt a divide and conquer approach. The hierarchy of the structure of a design is partitioned to zoom-in to lower levels where the analysis is tractable within reasonable computational time. Second, similar to MaCRO method [2.29], [2.30], we employ subblock level Spice simulations to derive transistor operating parameters. However, we conduct Spice simulations at realistic temperatures (different subblocks have different temperatures) rather than at a single worst-case temperature for the entire system as it is done pessimistically in [2.29], [2.30]. Third, we model failure times using Weibull and lognormal distributions that have been found to better fit experimental data [2.12]. Fourth, the block level reliability (as MTTF) is estimated via Monte Carlo simulations, which capture the combined effects of all the aging mechanisms considered. This process is implemented such that the design hierarchy is zoomed-out back to upper levels. Finally, as it will be discussed in the next section the proposed method has the ability to identify the most vulnerable sub-blocks from a reliability point of view.

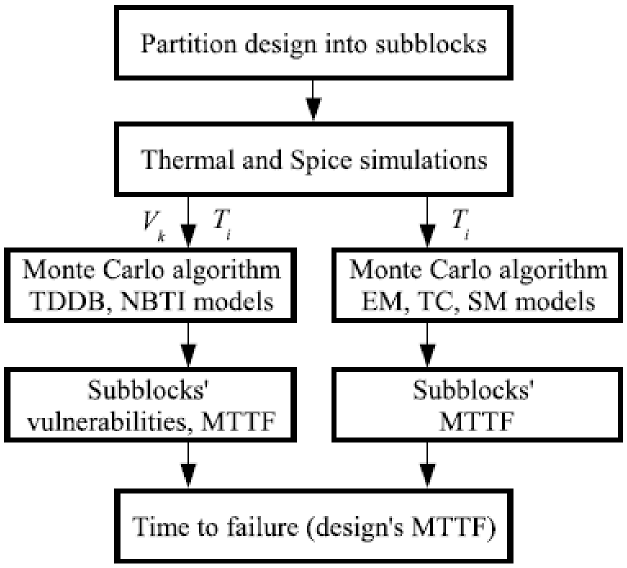
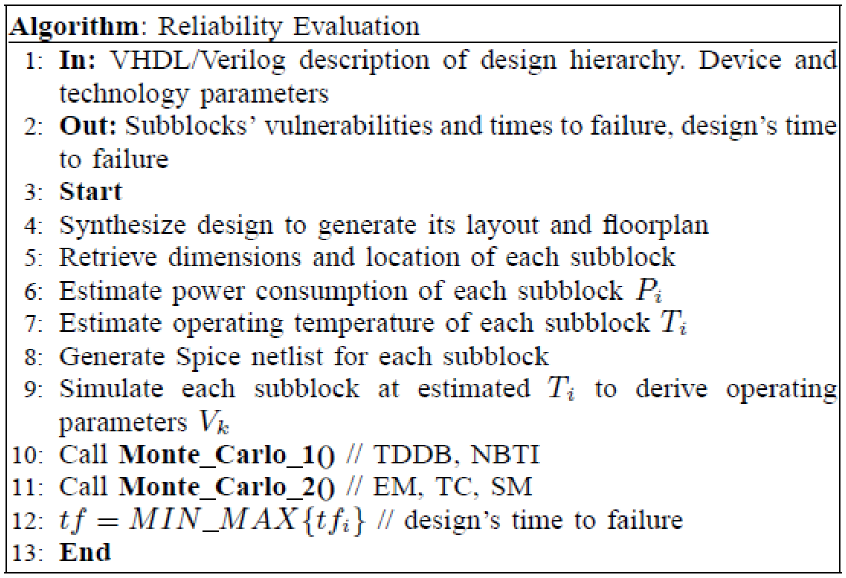

Partition design into subblocks
Thermal and Spice simulations
V_k
T_i
T_i
Monte Carlo algorithm TDDB, NBTI models
Monte Carlo algorithm EM, TC, SM models
Subblocks' vulnerabilities, MTTF
Subblocks' MTTF
Time to failure (design's MTTF)

Figure 2.1. Top level block diagram of the proposed reliability evaluation methodology.



```
Algorithm: Reliability Evaluation
 1: In: VHDL/Verilog description of design hierarchy. Device and
    technology parameters
 2: Out: Subblocks' vulnerabilities and times to failure, design's time
    to failure
 3: Start
 4: Synthesize design to generate its layout and floorplan
 5: Retrieve dimensions and location of each subblock
 6: Estimate power consumption of each subblock $P_i$
 7: Estimate operating temperature of each subblock $T_i$
 8: Generate Spice netlist for each subblock
 9: Simulate each subblock at estimated $T_i$ to derive operating
    parameters $V_k$
10: Call Monte_Carlo_1() // TDDB, NBTI
11: Call Monte_Carlo_2() // EM, TC, SM
12: $tf = MIN\_MAX\{tf_i\}$ // design's time to failure
13: End
```

Figure 2.2. Pseudocode description of the proposed reliability evaluation methodology.

The output of the proposed reliability evaluation methodology consists of the actual estimate of the time to failure1 or MTTF of the design (line number 12 in Fig. 2.2) and vulnerabilities of each individual sub-block as percentage of transistors with average failure time shorter than the selected threshold (discussed in the next subsection). MTTF is estimated using a MIN MAX type of analysis similar to [2.16] in order to be able to handle redundant sub-blocks that may be introduced for improving reliability via, for example, redundancy based fault tolerance techniques.

Because of the hierarchical approach and of the Spice level simulations, the proposed reliability evaluation methodology enjoys the benefits of both RAMP like and Spice simulation based reliability evaluation approaches discussed in the first section. In the next subsections, we describe the two Monte Carlo (MC) algorithms from Fig. 2.2. In the case of TDDB and NBTI failure mechanisms, the first MC algorithm works at the device level where operating temperatures and voltages are utilized. The remaining failure mechanisms, EM, TC, and SM, are modeled at the sub-block level in the second MC algorithm where only operating temperatures are utilized.

2.4.1. Reliability evaluation: TDDB and NBTI failure mechanisms

The block diagram that illustrates the main steps of the proposed reliability evaluation methodology to address TDDB and NBTI failure mechanisms is shown in Fig. 2.3. Additional details are provided by the pseudocode description from Fig. 2.4. Following the flow chart from Fig. 2.3, the main steps of the proposed reliability evaluation methodology are as follows:

Step 1: We start from a given hierarchical description of the design under consideration. This description can be in any hardware description language such as VHDL or Verilog. In addition, transistor and technology parameters are assumed to be given based on the technology node in which the design is to be fabricated.

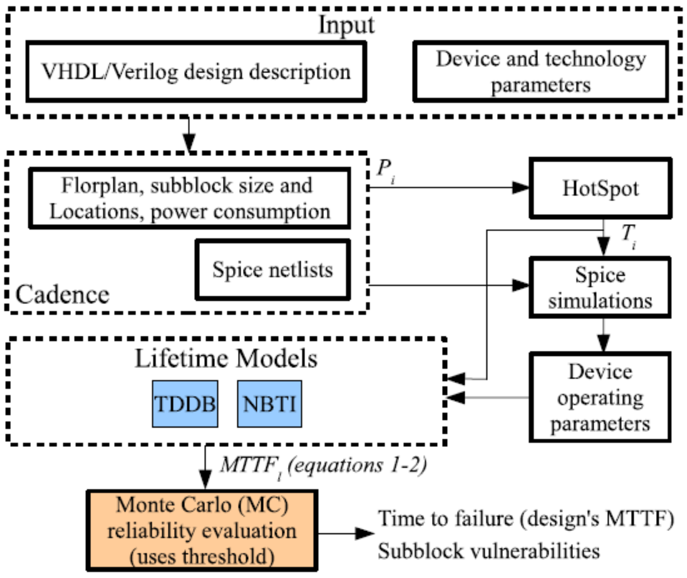

Input
VHDL/Verilog design description
Device and technology parameters
Florplan, subblock size and Locations, power consumption
Spice netlists
Cadence
P_i
HotSpot
T_i
Spice simulations
Device operating parameters
Lifetime Models
TDDB
NBTI
$MTTF_i$ (equations 1-2)
Monte Carlo (MC) reliability evaluation (uses threshold)
Time to failure (design's MTTF)
Subblock vulnerabilities

Figure 2.3. Flow chart of the proposed reliability evaluation methodology for TDDB and NBTI failure mechanisms.

```
Algorithm: Monte_Carlo_1()
In: Subblock level temperatures and device level operating voltages
Out: Subblocks' vulnerabilities and times to failure due to TDDB and NBTI
Estimate times to failure $tf_i$ and percentage of vulnerable transistors:
for $i \leftarrow 1$ to $S$ do // S: number of subblocks
    for $l \leftarrow 1$ to $F$ do // F: number of failure types
        Calculate $MTTF_l$ using equations 1,2 from Section II
        for $j \leftarrow 1$ to $N$ do // $N = 10^7$ Monte Carlo iterations
            $tf_{min}^j \leftarrow INF$ // Initialize
            $Counter_j \leftarrow 0$ // Initialize
            for $k \leftarrow 1$ to $D$ do // D: number of devices
                $tf_k \leftarrow generate\_sample(MTTF_l)$
                if $tf_k < tf_{min}^j$ then
                    $tf_{min}^j = tf_k$
                end if
                if $tf_k < threshold$ then
                    $Counter_j = Counter_j + 1$
                end if
            end for
            $Frac\_below\_threshold_j = 100 \cdot \frac{Counter_j}{D}$
        end for
        $tf_l = \frac{\sum_j^N tf_{min}^j}{N}$
        $Frac\_below\_threshold = \frac{\sum_j^N Frac\_below\_threshold_j}{N}$
    end for
    $tf_i = MIN\{tf_l\}$
end for
```

Figure 2.4. Pseudocode of the device level Monte Carlo algorithm, Monte_Carlo_1() from Fig. 2.3.

Step 2: The design is synthesized, placed, and routed using Cadence tools [2.44], but any other CAD tool can be utilized. The resulting layout represents the block level floorplan, which is divided into individual structures or sub-blocks based on the initial structural description of the design. In this way, we basically obtain for each sub-block its layout, location, and aspect ratio. In addition, power consumption estimates are also generated using Cadence tools.

Step 3: The floorplan and power estimates are then fed into HotSpot [2.45]. HotSpot is an accurate and fast thermal model based on an equivalent circuit of thermal resistances and capacitances that correspond to microarchitecture blocks. The output of the HotSpot simulation is a list with temperatures of each sub-block. Our approach addresses one of the limitations of MaCRO like methods [2.29], [2.30]. As mentioned earlier, instead of doing worst-case temperature simulations we work with the actual operating temperature for each sub-block. In addition, we utilize Weibull and lognormal rather than exponential distributions. Therefore, reliability of each sub-block can be evaluated more accurately.

Step 4: These temperatures are utilized together with circuit netlists generated from within Cadence tools to perform sub-block level Spice simulations. These simulations provide us with the transistor operating parameters necessary to be plugged into the equations modeling the wearout mechanisms described in Section 2.3. It is important to note that the level of design hierarchy at which this is done directly impacts the computational runtime, which increases with sub-block-circuit size.

Step 5: At this stage we have everything that is needed by the lifetime failure models described by equations 1 and 2 (or equations 2.6, 2.7, and 2.8 utilized by the algorithm described in the next subsection). At the core of the proposed methodology we employ a Monte Carlo simulation algorithm (see Figure 2.4) implemented and run in Matlab [2.46]. Our technique is inspired from the RAMP method [2.15], [2.17], [2.26] but executed at the sub-block level where the elementary unit is the device or transistor.

The MC algorithm proceeds with the following main steps (1) For each failure mechanism run N = 10^7 simulations: (a) for each transistor, generate failure time samples from the corresponding distribution and (b) use MIN analysis of these times by assuming the sub-block as a series system to calculate the time to failure tf_{min}^{j} of simulation j = 1, ...,N. (2) Calculate the overall sub-block time to failure for the current failure mechanism as tf_{l}= $\frac{(\sum_{j=1}^{N} tf_{min}^{j})}{N}$. (3) Calculate the value of the overall sub-block's time to failure as the minimum among the failure times due to each failure mechanism.

In our experiments, we found that in order to better differentiate between sub-blocks one only needs to focus on the most vulnerable transistors in a given sub-block. Hence, we introduce a threshold that helps to identify transistors whose lifetime samples are smaller than this threshold. As an indicator of how vulnerable a sub-block is, we calculate the percentage of transistors whose lifetime sample is smaller than the selected threshold. This is illustrated in the pseudocode description of the algorithm presented in Fig. 2.4. The threshold value is selected during the reliability qualification process as a function of the desired expected lifetime. An example is provided in the simulation results section. Computational runtime of the methodology described in Fig. 2.3 is in the order of hours for the design example studied later in the simulation results section. This computational runtime is mainly due to the Spice simulations and does not include the time spent on coding in Verilog the structural description of the design or the synthesis step with Cadence tools.

2.4.2. Reliability evaluation: EM, SM, and TC failure mechanisms

The block diagram that illustrates the main steps of the proposed reliability evaluation methodology to address EM, SM, and TC failure mechanisms is shown in Fig. 2.5 and is similar to that in Fig. 3. The main difference is that here the Monte Carlo analysis is done at the sub-block level as in the RAMP approach [2.16]. Therefore, only the HotSpot thermal simulator is utilized to estimate the operating temperature of each sub-block. Details of the MC simulation, which bears similarities that from Fig. 2.4, are provided by the pseudocode description from Fig. 2.6. Because in this case we work at sub-block level and do not perform Spice simulations, the computational runtime of the methodology described in Fig. 2.5 is in the order of minutes for the design example studied later in the simulation results section.

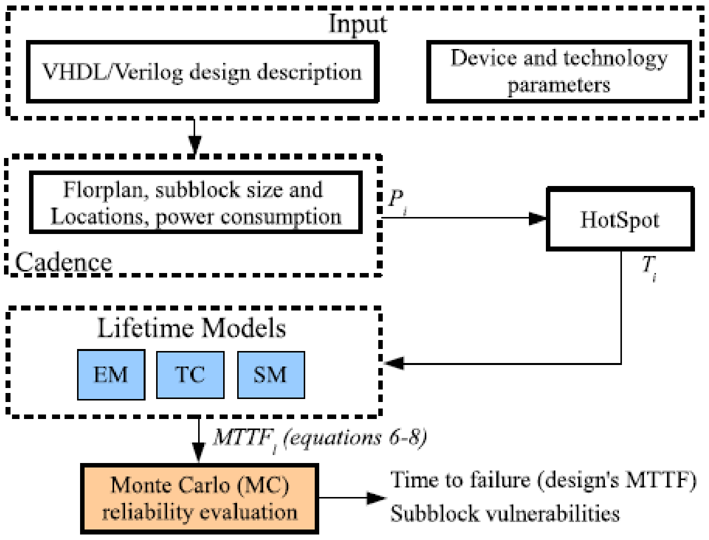
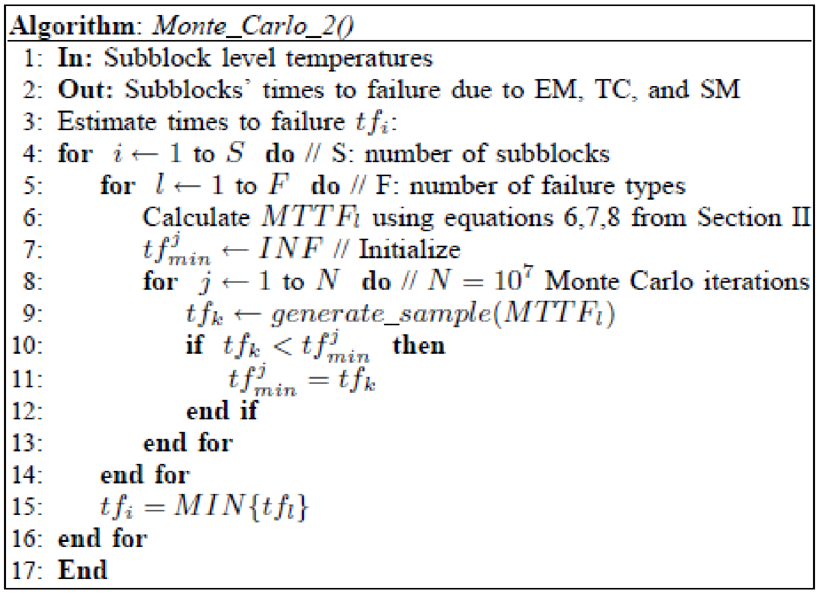

Input
VHDL/Verilog design description
Device and technology parameters
Florplan, subblock size and Locations, power consumption
Cadence
P_i
HotSpot
T_i
Lifetime Models
EM
TC
SM
$MTTF_i$ (equations 6-8)
Monte Carlo (MC) reliability evaluation
Time to failure (design's MTTF)
Subblock vulnerabilities

Figure 2.5. Flow chart of the proposed reliability evaluation methodology for EM, TC, and SM failure mechanisms.

2.4.3. Discussion

The information acquired from the proposed reliability evaluation methodology described in Fig. 2.2 can be useful to circuit and system designers to develop fault tolerant or robust circuits and systems. Armed with information about what are the reliability critical sub-blocks and transistors, designers can concentrate their design efforts [2.47], [2.48] with wearout mechanism specific techniques only on those, thereby saving area and power. In the next sections we provide two examples of scenarios where the proposed reliability evaluation methodology is utilized to search for lifetime aware floorplans and to investigate NoC routers.



```
Algorithm: Monte_Carlo_2()
In: Subblock level temperatures
Out: Subblocks' times to failure due to EM, TC, and SM
Estimate times to failure $tf_i$:
for $i \leftarrow 1$ to $S$ do // S: number of subblocks
    for $l \leftarrow 1$ to $F$ do // F: number of failure types
        Calculate $MTTF_l$ using equations 6,7,8 from Section II
        $tf^j_{min} \leftarrow INF$ // Initialize
        for $j \leftarrow 1$ to $N$ do // $N = 10^7$ Monte Carlo iterations
            $tf_k \leftarrow generate\_sample(MTTF_l)$
            if $tf_k < tf^j_{min}$ then
                $tf^j_{min} = tf_k$
            end if
        end for
    end for
    $tf_i = MIN\{tf_l\}$
end for
End
```

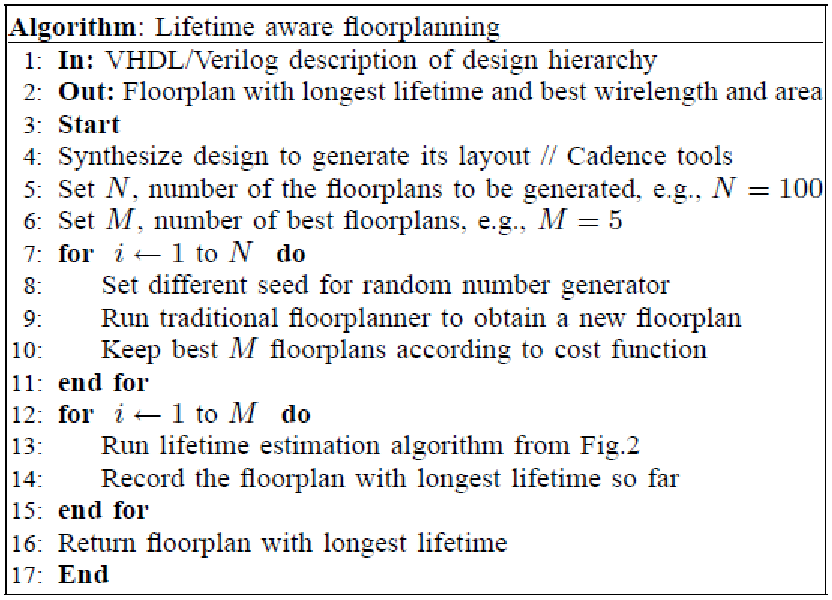
Figure 2.6. Pseudocode of the sub-block level Monte Carlo algorithm, Monte_Carlo_2() from Fig. 2.5.

2.5. Lifetime aware floorplanning

As an example on how the proposed reliability evaluation methodology can be utilized, we propose a lifetime aware floorplanning strategy. The objective of the proposed floorplanning strategy is to seek a floorplan that offers the longest lifetime for the design it represents, aside from optimizing traditional objectives such as total wire length or area. Because the lifetime estimation procedure (described by the algorithm from Fig. 2.2) has a computational runtime that makes it impractical to be included within the inner loop of the simulated annealing (SA) optimization engine (which may have hundreds or thousands of iterations), we adopt a heuristic approach described in the pseudocode from Fig. 2.7.

The idea is to utilize an existing floorplanning algorithm and run it multiple times starting from different initial conditions and then retain for lifetime evaluation only a smaller number of final floorplans. The following steps describe the proposed lifetime aware floorplanning strategy:

Step 1: Start from a given HDL description of the target design and utilize Cadence tools (though any other available tool can be utilized) to generate an initial layout.



```
Algorithm: Lifetime aware floorplanning
In: VHDL/Verilog description of design hierarchy
Out: Floorplan with longest lifetime and best wirelength and area
Start
Synthesize design to generate its layout // Cadence tools
Set $N$, number of the floorplans to be generated, e.g., $N = 100$
Set $M$, number of best floorplans, e.g., $M = 5$
for $i \leftarrow 1$ to $N$ do
    Set different seed for random number generator
    Run traditional floorplanner to obtain a new floorplan
    Keep best $M$ floorplans according to cost function
end for
for $i \leftarrow 1$ to $M$ do
    Run lifetime estimation algorithm from Fig.2
    Record the floorplan with longest lifetime so far
end for
Return floorplan with longest lifetime
End
```

Figure 2.7. Pseudocode of the proposed lifetime aware floorplanning strategy.

Step 2: Run traditional floorplanner a large number of times, say N = 100. We utilize an existing simulated annealing floorplanning algorithm, which works with a B*Tree representation of the design and with a traditional cost function: $\alpha.WireLength + (1 - \alpha).Area$ [2.49]. Initial conditions are set by resetting with a different seed the internal random number generator utilized to generate random sub-block swaps during the annealing process. In this way, during each run, the floorplanning algorithm arrives to a different final floorplan whose quality thus depends on the initial seed and the selected weight. During this step, the best − according to the traditional cost function − say M = 5 floorplans are recorded for processing in the next step. Each of these recorded M floorplans have already satisfactory wirelength and area.

Step 3: Estimate lifetime of each of the best M floorplans recorded in the previous step using the proposed reliability evaluation methodology from Section 2.4. Record and finally return the floorplan with the longest lifetime.

Given that the B*Tree floorplanner is very efficient and by keeping M reasonable small, the proposed lifetime aware floorplanning strategy is an effective approach to generate a floorplan solution that is a good tradeoff between wirelength, area, and reliability. The proposed floorplanning strategy is utilized in the simulation results presented in the next section. Finally, we note that one may want for the floorplanning process to be done such that certain constraints including fixed location or relative position among sub-blocks are satisfied. In such cases, one only needs to replace in the proposed strategy the floorplanning algorithm with another that is capable of handling such constraints.

2.6. Simulation results

In this section, we demonstrate the use of the proposed reliability evaluation methodology and the lifetime aware floorplanning strategy on a Network-on-Chip (NoC) router as a design example. We select the router as our target design because it is the key component of an NoC, which has become the dominant communication paradigm in today's SoCs to cope with the ever increasing complexity of integrated circuits. In addition, the reliability of NoCs has been studied significantly less compared to that of cores. Thus, our objective is to analyze the microarchitecture of a typical NoC router to identify its most vulnerable components and to generate its most reliable floorplan. While our discussion focuses on an NoC router, the entire analysis is applicable to any other block.

2.6.1. Router architecture

We focus our attention on the popular pipelined router architecture [2.50] whose block diagram is shown in Fig. 2.8. The main components of this architecture include: routing computation (RC), virtual channel allocation (VA), switch allocation (SA), crossbar switch, input ports, and output ports. We first code the router's structural description in Verilog.

Specifics of this description include: 5 input and 5 output ports, 2 virtual channels per port, 4 sets of registers for each virtual channel of each port, and 16 bites wide links. The Verilog description is utilized as input to the proposed reliability evaluation methodology described in Fig. 2.2 as well as the proposed lifetime aware floorplanning strategy from Fig. 2.7.

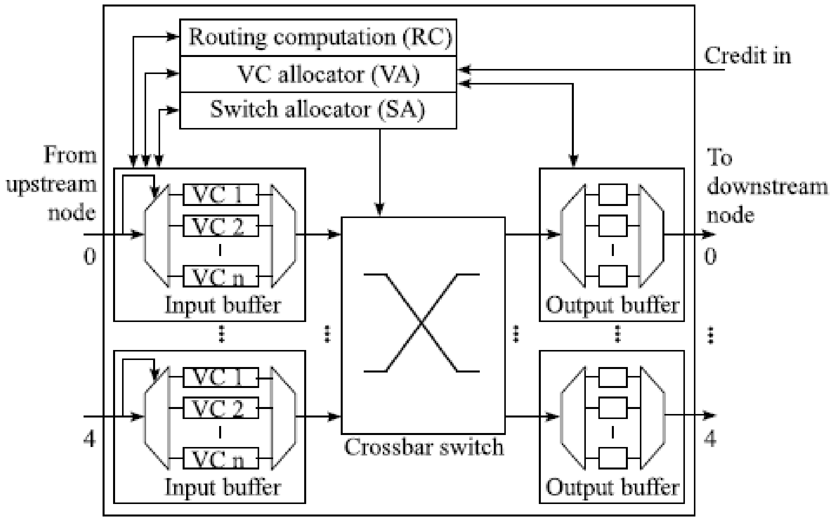


Figure 2.8. NoC router architecture.

2.6.2. Technology node and set-up parameters

We utilize Nangate 45nm Open Cell Library [2.51] within Cadence tools to synthesize and generate the layout of the router. In addition, Cadence tools generate Spice netlists and a list of power consumptions for each sub-block of the router. The traditional floorplanner utilized in the floorplanning strategy from Fig. 2.7 is set to be run for N = 100 times and M = 5 best floorplans are recorded. The power consumption values and the floorplans are utilized by HotSpot to estimate temperatures of the all sub-blocks. As mentioned earlier we partition the router into the following sub-blocks: RC, VA, SA, crossbar, and input and output ports. Spice simulations of all sub-block netlists are done at-temperature (as found by HotSpot) to estimate device operating parameters that are utilized inside the algorithm from Fig. 2.4.

The Monte Carlo algorithms introduced in Section 2.4 require the generation of lifetime samples (i.e., MTTFs) for devices (Fig. 2.4) or sub-blocks (Fig. 2.6) from corresponding Weibull or lognormal distributions as modeled in Section 2.3. To do that we utilize Matlab built-in functions. Because in the case of the Weibull distribution, we utilize a value for the shape parameter α = 1.2 [2.18] and have available the mean value MTTF as computed by the equations from Section 2.3, we need first to compute the scale parameter using the equation below to be able to use Matlab built-in functions.

$$\alpha = \frac{MTTF}{\Gamma\left(1 + \frac{1}{\beta}\right)} \tag{2.9}$$

where $\Gamma(\cdot)$ is the Gamma function and MTTF is the mean time to failure of the device/sub-block computed by equations 2.1, 2.2, 2.7, and 2.8. Because the router architecture has no redundancy (no fault tolerance techniques built-in) the overall lifetime is estimated like for a series system. In other words, the MIN MAX analysis from line number 12 of the algorithm from Fig. 2.2 needs only to take the minimum among all sub-blocks' MTTFs.

2.6.3. Results

2.6.3.1. Lifetime aware floorplanning and reliability evaluation

Once the layout of the router is generated with the Cadence tools we run the lifetime aware floorplanning algorithm described in Fig. 2.7. The best M = 5 floorplans (the one which turns out with the best MTTF is shown in Fig. 2.9) are recorded for further reliability evaluation, which requires also thermal simulation with HotSpot. As we plan to make publicly available the proposed algorithms, the whole methodology is automated and can be run with a simple Perl script.

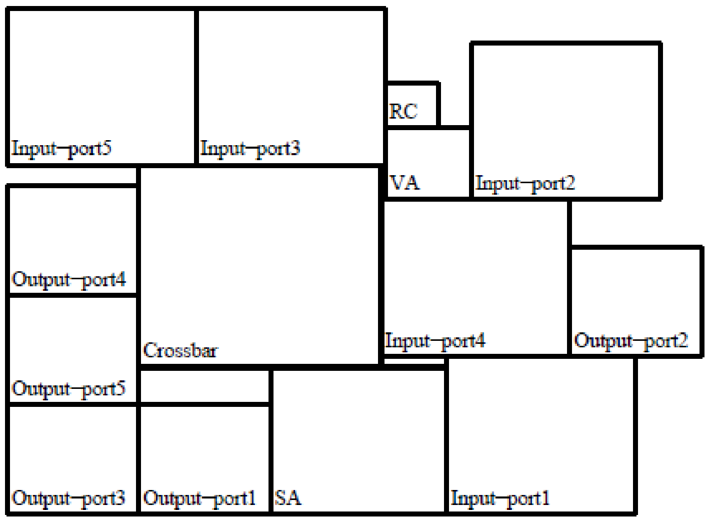
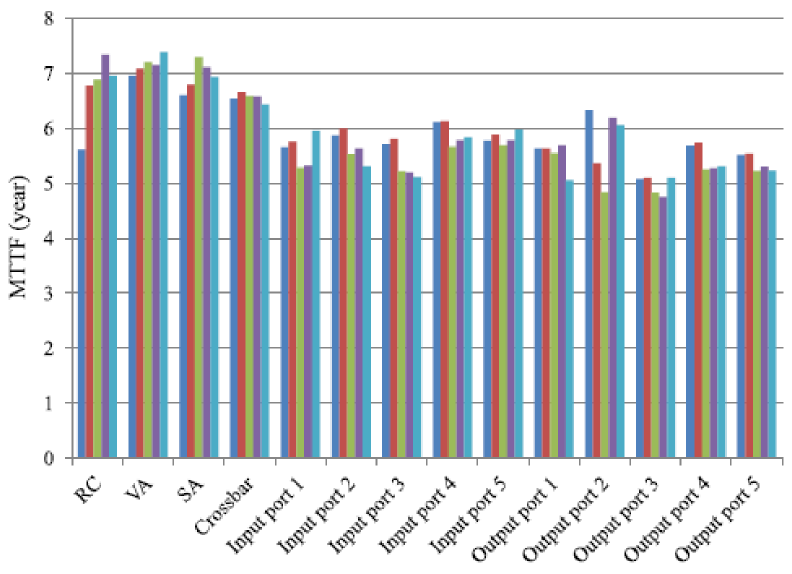


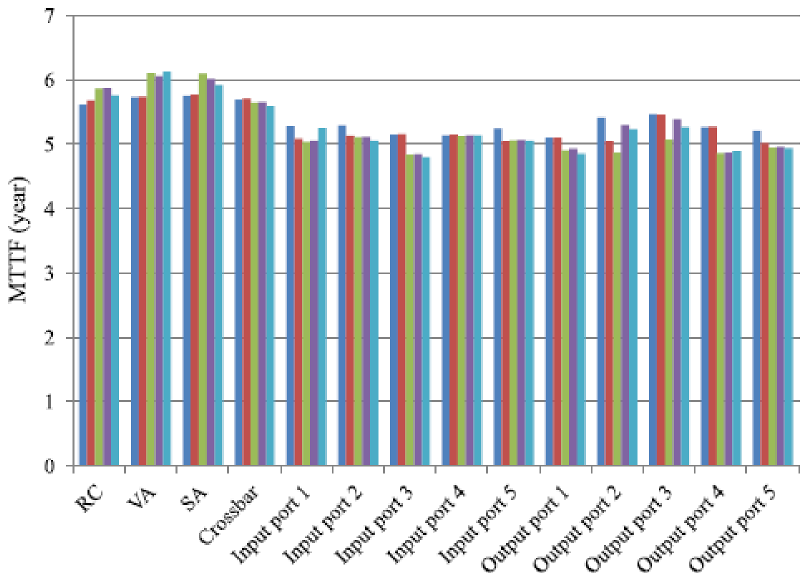
Figure 2.9. The best floorplan of the NoC router found by the lifetime aware floorplanning strategy described in Fig. 2.7.

Once the M = 5 best floorplans are found out, we then evaluate each of them to estimate their time to failure. To do that, we utilize the reliability evaluation methodology described in Fig. 2.2. The Monte Carlo algorithms on lines 10 and 11 of Fig. 2.2 and detailed in Fig. 2.4 and Fig. 2.6 estimate the mean times to failure of all sub-blocks for each of the five best floorplans. These MTTFs are reported in Fig. 2.10. We observe that while the MTTF of each block exhibits a sizable variation among all five floorplans, the relative comparison of MTTFs of different sub-blocks of a given floorplan stays relatively the same. Overall MTTFs of all five floorplans are plotted in Fig. 2.10(f). Note that the first floorplan has the longest lifetime and therefore it is identified as the most reliable floorplan for the studied NoC router. Fig. 2.11 shows with how much the first floorplan is better from an expected lifetime perspective compared to the other four floorplans. This figure demonstrates the value of the proposed lifetime aware floorplanning strategy.

In this example, the expected lifetime of the first floorplan is with 15% longer than the expected lifetime of the fifth floorplan.

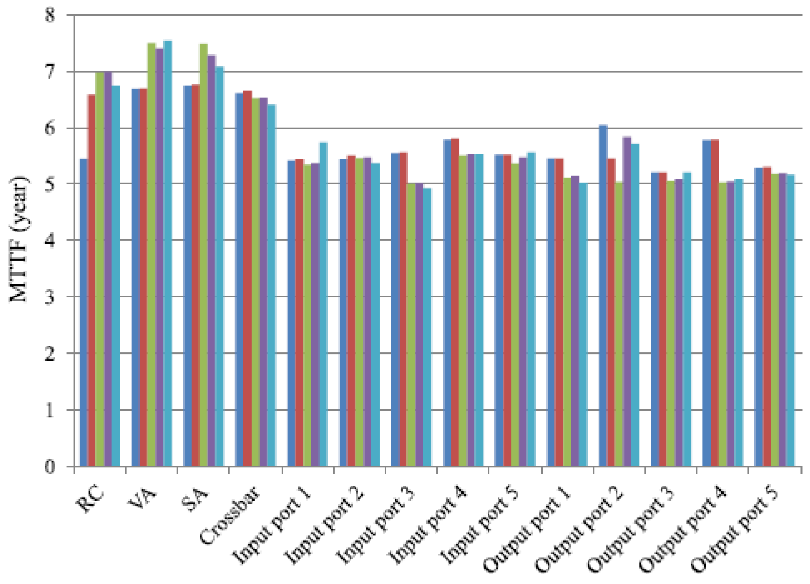


(a)

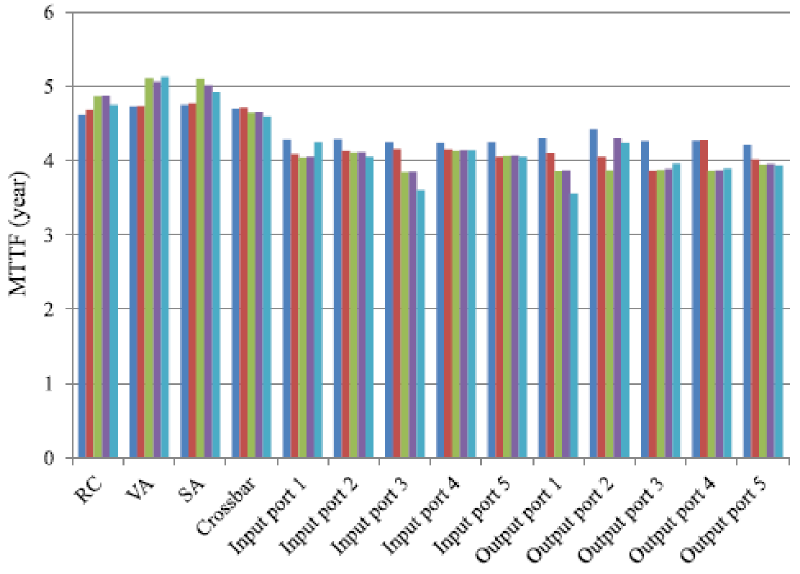


(b)

Figure 2.10. Mean time to failure of individual sub-blocks for a) TDDB, b) NBTI, c) EM, d) TC, and e) SM cases. Each of the five bars in each cluster corresponds to each of the five best floorplans. f) Overall MTTF of each of the five best floorplans.

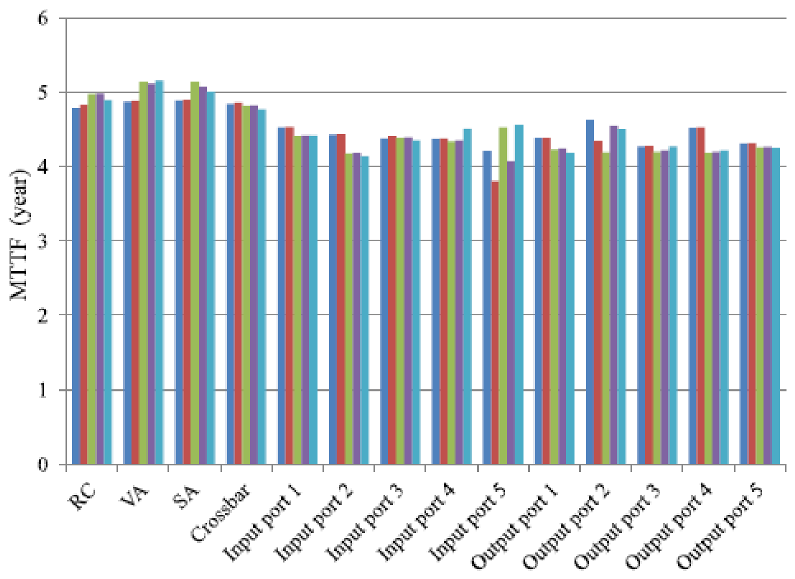


(c)

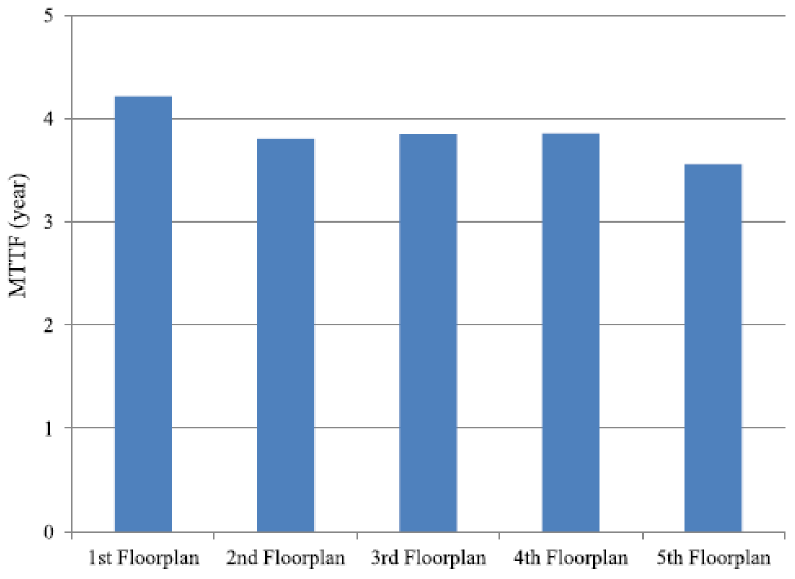


(d)

Figure 2.10. Mean time to failure of individual sub-blocks for a) TDDB, b) NBTI, c) EM, d) TC, and e) SM cases (continued). Each of the five bars in each cluster corresponds to each of the five best floorplans. f) Overall MTTF of each of the five best floorplans.



(e)



(f)

Figure 2.10. Mean time to failure of individual sub-blocks for a) TDDB, b) NBTI, c) EM, d) TC, and e) SM cases (continued). Each of the five bars in each cluster corresponds to each of the five best floorplans. f) Overall MTTF of each of the five best floorplans.

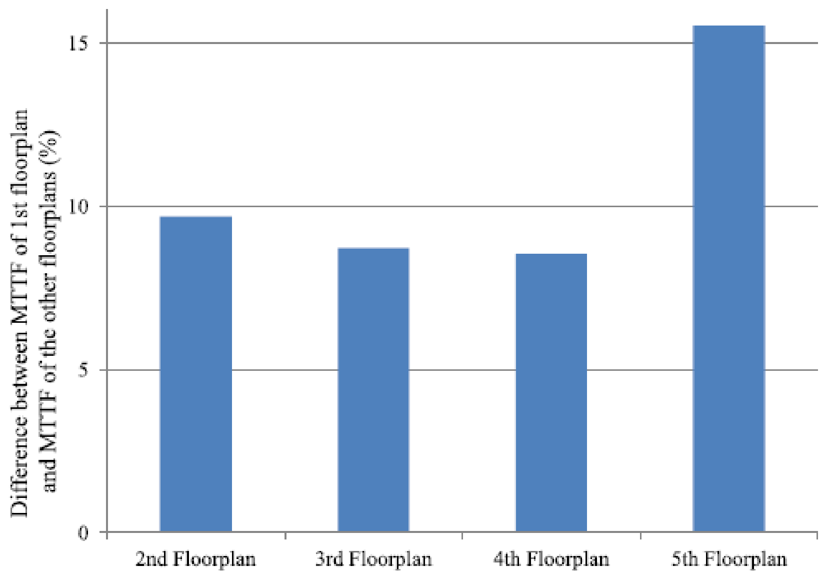


Figure 2.11. Illustration of the amount of the improvement in the expected lifetime of the first floorplan compared to the other 4 floorplans.

2.6.3.2. Vulnerability analysis

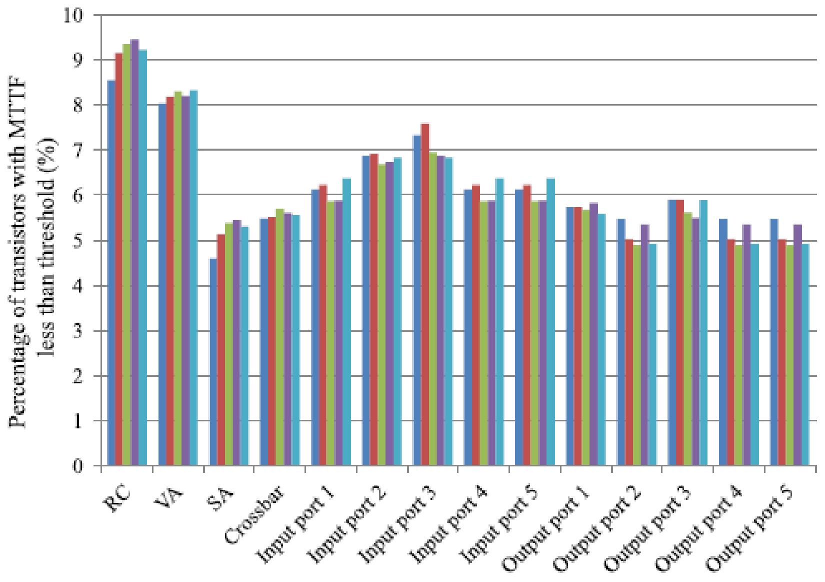
Note that the reliability evaluation methodology described in Fig. 2.3 provides us with sub-block vulnerabilities (computed as percentages of transistors with lifetime shorter than the selected threshold) to TDDB and NBTI failure mechanisms. This information basically helps us identify the most vulnerable sub-blocks in each floorplan. Although such information is not utilized by the lifetime aware floorplanning algorithm, it can prove very useful to system designers who want to develop effective (targeted) resilience techniques. This is the subject of our discussion in this section.

The proposed reliability evaluation methodology provides two types of vulnerability analysis. The first method operates at sub-block level and takes into account all failure mechanisms described in Section 2.3. It checks estimated MTTFs of all sub-blocks for different failure mechanisms and identifies the sub-block with the smallest MTTF and its corresponding failure mechanism. For example, applying this method to the case of the first floorplan from Fig.

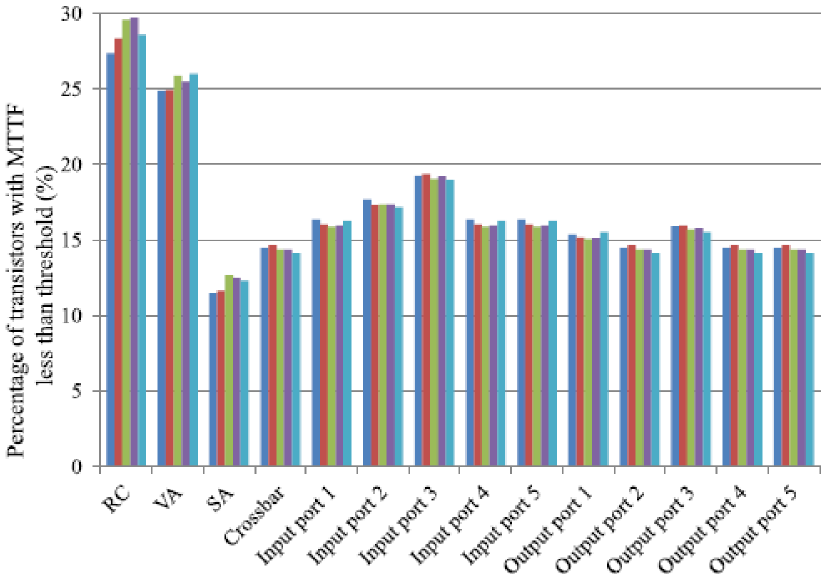
2.9, the most vulnerable sub-block is output port 5 and the corresponding failure mechanism is thermal cycling (TC).

The second method of vulnerability analysis operates at device level and only considers TDDB and NBTI failure mechanisms. This method is described in detail in Fig. 2.4. It requires first a threshold value, which must be defined by the system designer. This threshold reflects the time until when the system designer expects/hopes that the system will operate correctly without any failure. The main idea of the second vulnerability analysis is to identify and report the sub-block that has the highest percentage of transistors with MTTF less than the threshold value. In our example of the NoC router, we select the threshold value to be 8 years. The percentage of vulnerable transistors to TDDB and NBTI failure mechanisms in each sub-block is shown in Fig. 2.12. We observe that RC and VA sub-blocks contain the highest percentages of transistors with lifetime shorter than the selected threshold despite the fact that their area is smaller compared to for example the area of input registers. This can be explained by the fact that RC and VA components experience higher switching activities compared to the other router components, which in turn leads to higher temperatures. Note that this information could not be obtained with RAMP like reliability evaluation approaches.

It is well known that typically, resilience techniques to harden a system against different failure mechanisms require some form of redundancy. Such redundancy consumes valuable area and power resources, especially for designs with tight area and power budgets. As such, it may not be practical and desirable to develop systems with resilience technique to all types of failure mechanisms. Both vulnerability analysis methods discussed above provide system designers with valuable information about the reliability critical sub-blocks and transistors. It can help them to concentrate their design efforts on the critical sub-blocks, thereby saving area and power resources.



(a)



(b)

Figure 2.12. Percentage of transistors with MTTF value lower than selected threshold for a) TDDB and b) NBTI cases.

2.7. Conclusion

We proposed and implemented a new circuit level divide and conquer based reliability evaluation methodology, which enjoys the benefits of transistor level accuracy and of block level efficiency. At the core of the lifetime estimation engine lies a Monte Carlo algorithm which works with failure times modeled as Weibull and lognormal distributions. Using the proposed reliability evaluation methodology we developed a lifetime aware floorplanning strategy. We consider the proposed strategy an important step towards reliability oriented design in general with the potential of improvement via floorplanning. The new floorplanning approach was able to find floorplans with up to 15% difference in the lifetime of a Network-on-Chip router design example. In addition, we applied the proposed reliability evaluation methodology to the router design and identified the routing computation and virtual channel allocation units as the most vulnerable sub-blocks.

2.8. References

[2.1] S. Borkar, “Designing reliable systems from unreliable components: The challenges of transistor variability and degradation,” IEEE Micro, vol. 25, no. 6, pp. 10–16, Nov./Dec. 2005.

[2.2] V. Raghunathan, M. B. Srivastava, and R. K. Gupta, “A survey of techniques for energy efficient on-chip communication,” in Proc. ACM/IEEE DAC, 2003, pp. 900–905.

[2.3] A. DeHon, H. M. Quinn, and N. P. Carter, “Vision for cross-layer optimization to address the dual challenges of energy and reliability,” inProc. ACM/IEEE DATE, 2010, pp. 1017–1022.

[2.4] S. Mitra, K. Brelsford, and P. N. Sanda, “Cross-layer resilience challenges: Metrics and optimization,” in Proc. ACM/IEEE DATE, 2010, pp. 1029–1034.

[2.5] M. White and J. B. Bernstein, “Microelectronics reliability: Physics-of failure based modeling and lifetime evaluation,” Jet Propulsion Lab., California Inst. Technol., Pasadena, CA, Feb. 2008.

[2.6] J. H. Stathis, “Reliability limits for the gate insulator in CMOS technology,” IBM J. Res. Develop., vol. 46, no. 2/3, pp. 265–286, Mar. 2002.

[2.7] D. K. Schroder and J. A. Babcock, “Negative bias temperature instability: Road to cross in deep submicron silicon semiconductor manufacturing,” J. Appl. Phys., vol. 94, no. 1, pp. 1–18, Jul. 2003.

[2.8] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, “A finite oxide thickness based analytical model for negative temperature bias instability,” IEEE Trans. Device Mater. Rel., vol. 9, no. 4, pp. 537–556, Dec. 2009.

[2.9] W. Wang, S. Yang, S. Bhardwaj, R. Vattikonda, S. Vrudhula, F. Liu, and Y. Cao, “The impact of NBTI effect on combinational circuit: Modeling, simulation, and analysis,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 18, no. 2, pp. 173–183, Feb. 2010.

[2.10] S. M. Alam, C. L. Gan, D. E. Troxel, and C. V. Thompson, “Circuitlevel reliability analysis of Cu interconnects,” in Proc. ISQED, 2004, pp. 238–243.

[2.11] Z. Lu, J. Lach, M. R. Stan, and K. Skadron, “Temperature-aware modeling and banking of IC lifetime reliability,” IEEE Micro, vol. 25, no. 6, pp. 40–49, Nov./Dec. 2005.

[2.12] JEDEC, “Failure mechanisms and models for semiconductor devices,” JEDEC Publication JEP122E, 2009.

[2.13] P. S. Ho and T. Kwok, “Electromigration in metals,” Rep. Prog. Phys., vol. 52, no. 3, pp. 301–348, Mar. 1989.

[2.14] J. Srinivasan, S. V. Adve, P. Bose, J. A. Rivers, and C. K. Hu, “RAMP: A model for reliability aware microprocessor design,” IBM, Armonk, NY, IBM Res. Rep. RC23048, Dec. 2003.

[2.15] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, “The case for lifetime reliability-aware microprocessors,” in Proc. IEEE Int. Symp. Comput. Architecture, Jun. 2004, pp. 276–287.

[2.16] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, “Lifetime reliability: Toward an architectural solution,” IEEE Micro, vol. 25, no. 3, pp. 70–80, May 2005.

[2.17] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, “Exploiting structural duplication for lifetime reliability enhancement,” in Proc. IEEE ISCA, 2005, pp. 520–531.

[2.18] A. K. Coskun, T. S. Rosing, K. Mihic, G. D. Micheli, and Y. Leblebici, “Analysis and optimization of MPSoC reliability,” J. Low Power Electron., vol. 2, no. 1, pp. 56–69, Apr. 2006.

[2.19] Z. Gu, C. Zhu, L. Shang, and R. P. Dick, “Application-specific MPSoC reliability optimization,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 16, no. 5, pp. 603–608, May 2008.

[2.20] J. Fang and S. S. Sapatnekar, “Scalable methods for analyzing the circuit failure probability due to gate oxide breakdown,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 20, no. 11, pp. 1960–1973, Nov. 2012.

[2.21] M. R. Choudhury, V. Chandra, K. Mohanram, and R. Aitken, “Analytical model for TDDB-based performance degradation in combinational logic,” in Proc. ACM/IEEE DATE, 2010, pp. 423–428.

[2.22] K. Kang, K. Kim, A. E. Islam, M. A. Alam, and K. Roy, "Characterization and estimation of circuit reliability degradation under NBTI using on-line IDDQ measurement," in Proc. ACM/IEEE DAC, 2007, pp. 358–363.

[2.23] E. Maricau and G. Gielen, "Efficient reliability simulation of analog ICs including variability and time-varying stress," inProc. ACM/IEEE DATE, 2009, pp. 1238–1241.

[2.24] M. Bashir and L. Milor, "Towards a chip level reliability simulator for copper/low-k backend processes," in Proc. ACM/IEEE DATE, 2010, pp. 279–282.

[2.25] J. B. Bernstein, M. Gurfinkel, X. Li, J. Walters, Y. Shapira, and M. Talmor, "Electronic circuit reliability modeling," Microelectron. Rel., vol. 46, no. 12, pp. 1957–1979, Feb. 2006.

[2.26] P. Ramachandran, S. V. Adve, P. Bose, J. A. Rivers, and J. Srinivasan, "Metrics for architecture-level lifetime reliability analysis," inProc. IEEE ISPASS, 2008, pp. 202–212.

[2.27] X. Li, B. Huang, J. Qin, X. Zhang, M. Talmor, Z. Gur, and J. B. Bernstein, "Deep submicron CMOS integrated circuit reliability simulation with SPICE," in Proc. IEEE ISQED, 2005, pp. 382–389.

[2.28] S. Zafar, "Statistical mechanics based model for negative bias temperature instability induced degradation," J. Appl. Phys., vol. 97, no. 10, pp. 103 709-1–103 709-9, May 2005.

[2.29] X. Li, J. Qin, B. Huang, X. Zhang, and J. B. Bernstein, "SRAM circuit failure modeling and reliability simulation with SPICE," IEEE Trans. Device Mater. Rel., vol. 6, no. 2, pp. 235–246, Jun. 2006.

[2.30] X. Li, J. Qin, B. Huang, X. Zhang, and J. B. Bernstein, "A new SPICE reliability simulation method for deep submicrometer CMOS VLSI circuits," IEEE Trans. Device Mater. Rel., vol. 6, no. 2, pp. 247–257, Jun. 2006.

[2.31] A. Ghetti, “Gate oxide reliability: Physical and computational models,” Springer Ser. Mater. Sci., vol. 72, pp. 201–258, 2004.

[2.32] X. Li, J. Qin, and J. B. Bernstein, “Compact modeling of MOSFET wearout mechanisms for circuit-reliability simulation,” IEEE Trans. Device Mater. Rel., vol. 8, no. 1, pp. 98–121, Mar. 2008.

[2.33] Y. Han and I. Koren, “Simulated annealing based temperature aware floorplanning,” J. Low Power Electron., vol. 3, no. 2, pp. 141–155, Aug. 2007.

[2.34] C. C. Ta, X. Zhang, L. He, and T. T. Jing, “Temperature aware microprocessor floorplanning considering application dependent power load,” in Proc. ACM/IEEE ICCAD, 2007, pp. 586–589.

[2.35] W. L. Hung, Y. Xie, N. Vijaykrishnan, C. A. Quaye, T. Theocharides, and M. J. Irwin, “Thermal-aware floorplanning using genetic algorithms,” in Proc. ISQED, 2005, pp. 634–639.

[2.36] K. Sankaranarayanan, S. Velusamy, M. R. Stan, and K. Skadron, “A case for thermal-aware floorplanning at the microarchitectural level,” J. Instruct.-Level Parall., vol. 8, pp. 1–16, Oct. 2005.

[2.37] J. Kung, I. Han, S. Sapatnekar, and Y. Shin, “Thermal signature: A simple yet accurate thermal index for floorplan optimization,” in Proc. ACM/IEEE DAC, 2011, pp. 108–113.

[2.38] V. Nookala, D. J. Lilja, and S. S. Sapatnekar, “Temperature-aware floorplanning of microarchitecture blocks with IPC-power dependence modeling and transient analysis,” in Proc. ISLPED, 2006, pp. 298–303.

[2.39] H. D. Mogal and K. Bazargan, “Thermal-aware floorplanning for task migration enabled active sub-threshold leakage reduction,” in Proc. ACM/IEEE ICCAD, 2008, pp. 302–305.

[2.40] S. Yang, W. Wolf, N. Vijaykrishnan, and Y. Xie, “Reliability-aware SOC voltage islands partition and floorplan,” in Proc. IEEE Symp. Emerging VLSI Technol. Architectures, 2006, pp. 343–348.

[2.41] J. Minz, E. Wong, and S. K. Lim, “Reliability-aware floorplanning for 3D circuits,” in Proc. IEEE Int. SOC Conf., 2005, pp. 81–82.

[2.42] A. Gupta, A. Djahromi, A. Eltawil, N. Dutt, and F. Kurdahi, “Managing leakage power and reliability in hot chips using system floorplanning and SRAM design,” in Proc. IEEE Int. Workshop THERMINIC, 2008, pp. 37–42.

[2.43] J. Shin, V. Zyuban, Z. Hu, J. Rivers, and P. Bose, “A framework for architecture-level lifetime reliability modeling,” in Proc. IEEE/IFIP Int. Conf. DSN, 2007, pp. 534–543.

[2.44] Available: www.cadence.com

[2.45] Available: http://lava.cs.virginia.edu/HotSpot

[2.46] Available: http://www.mathworks.com/products/matlab

[2.47] J. Kim, D. Park, C. Nicopoulos, N. Vijaykrishnan, and C. R. Das, “Design and analysis of an NoC architecture from performance, reliability and energy perspective,” in Proc. ACM Symp. ANCS, 2005, pp. 173–182.

[2.48] A. DeOrio, D. Fick, V. Bertacco, D. Sylvester, D. Blaauw, J. Hu, and G. Chen, “A reliable routing architecture and algorithm for NoCs,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 31, no. 5, pp. 726–739, May 2012.

[2.49] T.-C. Chen and Y.-W. Chang, “Modern floorplanning based on B*-trees and fast simulated annealing,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 25, no. 4, pp. 637–650, Apr. 2006.

[2.50] W. J. Dally and B. Towles, Principles and Practices of Interconnection Networks. San Mateo, CA: Morgan Kaufmann, 2004.

[2.51] Available: http://www.si2.org

CHAPTER 3. IMPROVING FAULT TOLERANCE OF NETWORK-ON-CHIP LINKS VIA MINIMAL REDUNDANCY AND RECONFIGURATION

This paper was presented in International Conference on Reconfigurable Computing and FPGAs, 2011. The authors of the paper are Hamed Sajjadi Kia and Cristinel Ababei.

3.1. Abstract

We propose to partition links in a network-on-chip into multiple segments and use spare wires at the level of each segment to address permanent errors due to manufacturing or wearout defects. Because different segments of the spare wires address different errors from different segments, the proposed reconfigurable link structure can tolerate a larger number of errors with a reduced number of spare wires. The proposed self-repairing segmented link structure is implemented and simulated in Verilog and verified on a Virtex 5 FPGA. Experimental results on area, power consumption, delay, and reliability show that the optimal link is achieved when the link is partitioned into two segments.

3.2. Introduction

Advances in integrated circuit fabrication technology enables the integration of tens and hundreds of cores on the same system-on-chip (SoC). To address the demand for increased communication and concurrency between cores, network-on-chip (NoC) has emerged as a new communication design paradigm [3.1]. An NoC is constructed by connecting a set of routers via bidirectional links. Cores are connected to routers through network interfaces and communicate via messages organized as packets. For example, a typical 2D regular mesh NoC topology is shown in Fig.1.

However, advances in fabrication technology that can make this integration possible may also make the underlying hardware less reliable due to an increasing number of defects and wearout or aging mechanisms. Errors due to these faults may occur in both the computation (cores) and communication (network) components of the SoC. Because it is critical to deal with these faults and address them with cost-effective solutions, one of the major problems facing the design of network-on-chip based systems-on-chip is reliability.

In this paper, we focus on the communication component and address permanent faults which can occur in NoC links during fabrication or due to wearout mechanisms. More specifically, we improve the fault tolerance of NoCs by using redundant or spare wires within the context of segmented links. We do not focus on the computation component, which has been studied more extensively [3.2], [3.3]. The remainder of this paper is organized as follows. In the next section, we discuss previous work and then outline our main contribution. Then, we describe the proposed self-repairing link and introduce a simple technique to detect faults. Later, we report experimental results. Finally, we conclude by summarizing our main contribution.

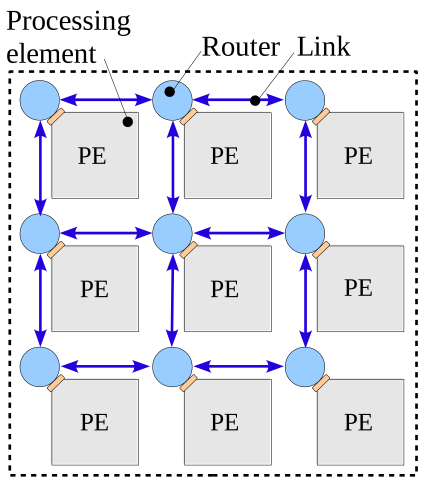


Figure 3.1. Example of typical 2D regular mesh NoC topology.

3.3. Previous work

There are basically two methods to address permanent faults in NoC links. The most popular method is based on the use of adaptive or dynamic routing strategies. The idea is that after faulty links are detected and located, specialized routing algorithms compute new routing paths through the network such that these broken links are avoided. Examples of recently proposed adaptive routing algorithms for NoCs include [3.4]–[3.9]. While this method is straightforward, it has the disadvantage of affecting the network performance (average flit latency) due to typically longer routing paths. Also, it must deal with the issue of deadlock and livelock.

The second method to address permanent faults is based on the concept of adding redundant or spare elements and on the use of reconfiguration [3.10]–[3.14]. In the case of NoC links, several redundant wires are added to each link. When faults occur, reconfiguration is used to replace the faulty wires with spare healthy wires. For example, the authors of [3.15] have used redundant spare wires to repair broken links. The use of redundant wires to replace faulty wires without interruption of the data flow is presented in [3.16]. The authors use a periodic inline test method to detect faulty wires. The authors in [3.17] demonstrate link structures, which can tolerate transient, intermittent, and permanent faults. They use Hamming coding to address transient errors and retransmission as the recovery technique. They have also introduced two techniques to address intermittent and permanent faults. The first technique is based on spare wires and reconfiguration while the second technique uses time redundancy. This method has the disadvantage of addressing a rather small number of faults in a given link, which is limited by the additional area occupied by the multiplexers and demultiplexers utilized to reconfigure the link.

The use of spare wires to repair faulty interconnects has the disadvantage of addressing a rather small number of faults in a given link, which is limited by the additional area occupied by spare wires, and the multiplexers and demultiplexers utilized to reconfigure the link.

3.4. Contributions

In this paper, we focus on permanent faults which can occur in NoC links during fabrication or due to wearout mechanisms. Our goal is to design a self-repairing NoC link, using spare wires and reconfiguration, which offers the best trade off between the overhead in area, power consumption, delay and reliability. Our main contribution lies in: 1) We propose to partition NoC links into multiple segments and use spare wires at the level of each segment. Because different segments of the spare wires address different faults from different segments, the proposed reconfigurable link structure can tolerate a larger number of errors with a reduced number of spare wires. 2) We implement and simulate the proposed self-repairing link structure in Verilog using Cadence tools and verify it on a Virtex 5 FPGA. Experimental results on area, power consumption, delay, and reliability show that the optimal link is achieved when the link is partitioned into two segments.

3.5. Proposed self-repairing link structure

In this section, we introduce the proposed segmented self-repairing link structure. However, first we present the traditional non-segmented link structure, which we will use to build our discussion and for comparison purposes.

3.5.1. Non-segmented link structure

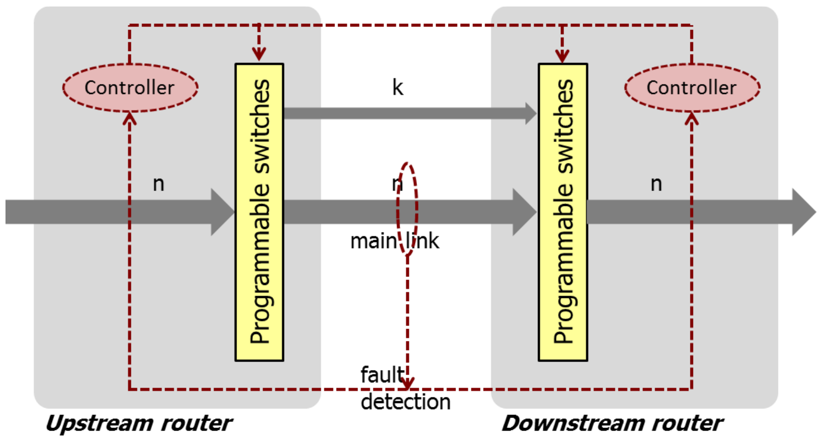
The block diagram of the traditional reconfigurable link is shown in Fig. 2.a. The main link connecting two routers of the NoC is essentially an n-bit bus formed of n primary wires. In addition, the link contains k spare wires. These spare wires are redundant and are normally not used. The error detection logic is responsible with the detection of permanent faults, which initially are assumed to occur only in the n primary wires. This is a reasonable assumption because the spare wires are not normally used and therefore they are not affected by wearout mechanisms. The error detection logic is designed to test all n + k wires because once a spare wire is used to reconfigure the link, it will also be affected by wearout mechanisms. Once a fault is detected by the error detection logic, the link is reconfigured to replace the faulty wire with one of the healthy k redundant wires. The reconfiguration is realized with two sets of configurable switches (multiplexers and demultiplexers) controlled by signals generated by controllers, which reside in the downstream and upstream routers.

While the link structure in Fig. 2.a is simple, it suffers from poor scalability. Obviously, as the number of redundant bits k increases the reliability of the whole link increases too. However, the link structure must use n x 1 - to - (k + 1) demultiplexers in the upstream router and n x (k+1) –to – 1 multiplexers in the downstream router. In addition, the complexity and hence area of the two controllers also increases with the increase of k. Therefore, in order to keep the area overhead within reasonable limits, in practice k must be limited to a small number. The block diagram in Fig. 2.b shows implementation details. In this case the number of redundant bits is k = 4 and therefore the link can only handle up to 4 faults. Once the number of faults increases to more than 4, the controllers enable a failure signal to inform the routers that the link is broken and cannot be used anymore at the initial bandwidth. This information can then be utilized by adaptive routing algorithms, which will find new routing paths such that the broken link is avoided.

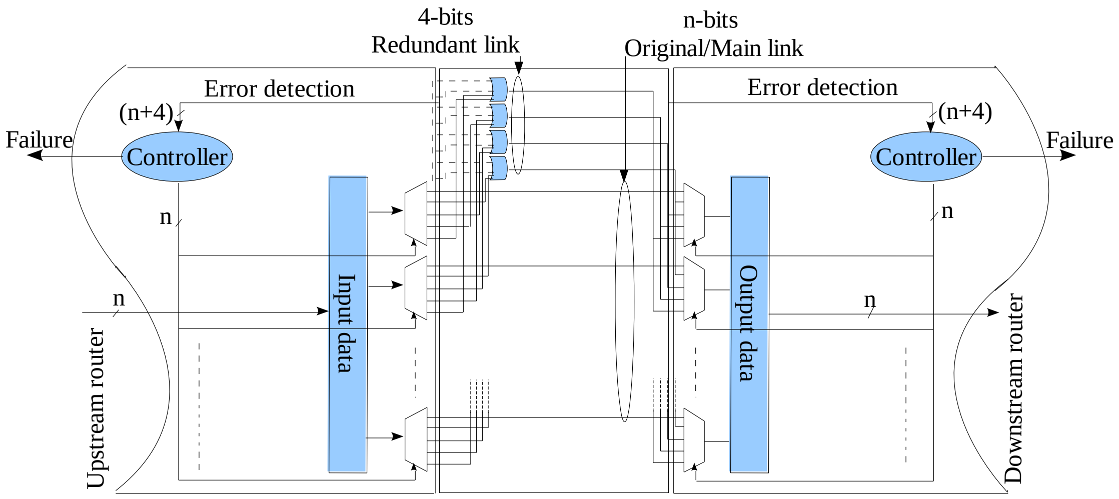
The reliability of the link structure from Fig. 2.a can be computed using an approach similar to the study in [3.18]. For simplicity, we assume that faults can only occur in the main link and that the redundant bits are fault free. The link structure in Fig. 2.a can be seen as a parallel system. Assuming that all n wires of the main link are identical and that the probability of success of any of the n bits is p (that is the probability of the wire to be functional), then the probability of exactly (n - k) bits working correctly out of n bits is given by the binomial distribution:

$$P(n-k,n,p) = \binom{n}{n-k} p^{n-k}(q)^k \tag{3.1}$$

where q = 1 - p represents the probability of the wire to be non-functional.



(a)



(b)

Figure 3.2. (a) Simplified block diagram of an n-bit reconfigurable link structure with k redundant bits. (b) Details of the reconfiguration logic for an example with k = 4.

The link structure is said to work successfully as long as at least (n - k) bits of the main link remain functional. Therefore, the reliability of the link structure can be defined as the probability obtained by summing-up the probabilities of all possible successful configurations:

$$R(n-k,n,p)=\sum_{i=n-k}^{n}\binom{n}{i}p^{i}(q)^{n-i} \tag{3.2}$$

3.5.2. Proposed segmented link structure

To improve its fault tolerance, we propose to partition the link into multiple segments and use spare wires at the level of each segment. Because different segments of the spare wires address different errors from different segments, the proposed reconfigurable link structure can tolerate a larger number of errors with a reduced number of spare wires. Fig. 3 illustrates the main idea behind the proposed segmented link structure. In this figure, the link with only one redundant wire is divided into 3 segments. Faults, represented as ’x’, can occur anywhere along the length of the link. The ’x’ on the first wire in the first segment of the link illustrates the occurrence of a fault. The link structure from Fig. 2 would replace this faulty wire with the only available redundant wire. However, in this example there are two remaining faults, which can not be addressed by the link structure from Fig. 2. In the proposed segmented link structure, we use only the first segment of the redundant wire to replace the faulty wire from the first link segment. The second and third faults denoted with ’x’ in Fig. 3 can be handled using the second and third segments of the redundant wire. Next, we discuss the main limitation of the proposed segmented link structure and present an elegant solution to address it.

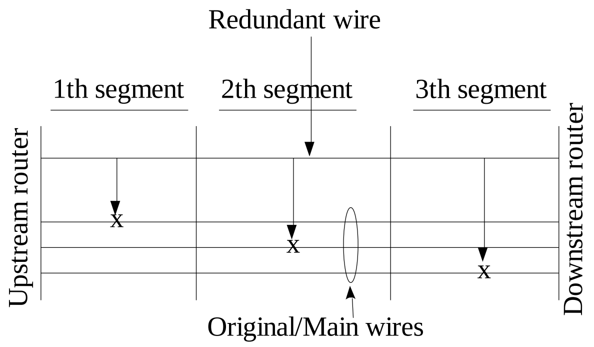


Figure 3.3. Illustration of proposed segmented link structure.

The proposed segmented link has better fault tolerance (hence better reliability) compared to traditional nonsegmented link structure. However, it also suffers from the scalability problem as the link structure from Fig. 2. In fact, the proposed link structure would utilize larger area overheads due to the segmentation, which requires multiple sets of multiplexers and demultiplexers, error detection circuits, and more sophisticated controllers.

To solve the scalability problem and to reduce the hardware overhead, the main and redundant wires are divided into L groups or subsets. Each of the groups of main wires has n_i bits and each of the groups of redundant wires has k_i bits, where i = 1, 2,…, L. Each of the groups of main wires has designated a group of redundant wires. Faulty wires from a given main group can be replaced only by wires from the designated group of redundant wires. This solution effectively reduces the size of the programmable switches (that is multiplexers and demultiplexers) yet enabling the improvement in fault tolerance. In this case, we only need to use $L \times n_i \times 1$ - to - $(k_i + 1)$ demultiplexers and $L \times n_i \times (k_i+1)$-to-1 multiplexers. The proposed solution is illustrated in the block diagram from Fig. 4.

To derive an expression for reliability, we again assume that faults can only occur in the main wires and that the redundant wires are fault free. Our derivation is based on the block diagram shown in Fig. 5. In this figure R_b represents the reliability of a block composed of a group of main wires together with its designated group of redundant wires. Notice that since we divided the link into m segments, the probability of failure of a bit of the main link in any segment is now $(\frac{1}{m})$.q, and hence the probability of success is (p + $(\frac{m-1}{m})$.q). Similar to the discussion from the previous section, we assume that all wires of the main link are identical. Then, the probability of exactly (n_i - k_i) bits working correctly out of n_i bits is given by the binomial distribution:

$$P(n_i - k_i, n_i, p) = \binom{n_i}{n_i - k_i} (p + \left(\frac{m-1}{m}\right) q)^{(n_i - k_i)} (\left(\frac{1}{m}\right) q)^{k_i} \tag{3.3}$$

Therefore the reliability of one block of a segment (formed by a group of main wires and its designated group of redundant wires) can be computed as:

$$R_b(n_i - k_i, n_i, p) = \sum_{j=n_i-k_i}^{n_i} \binom{n_j}{j} (p + \left(\frac{m-1}{m}\right) q)^{j} (\left(\frac{1}{m}\right) q)^{n_i - j} \tag{3.4}$$

Because failure of any block of a segment results in system failure, the segment must be modeled as a system of a series of blocks as illustrated in Fig. 5. Therefore, the overall reliability of a segment can be computed as:

$$R_s = R^L \tag{3.5}$$

Finally, the whole segmented link is modeled as a series system, for which the total reliability of the system can be computed as:

$$R_{link} = R_s^m \tag{3.6}$$

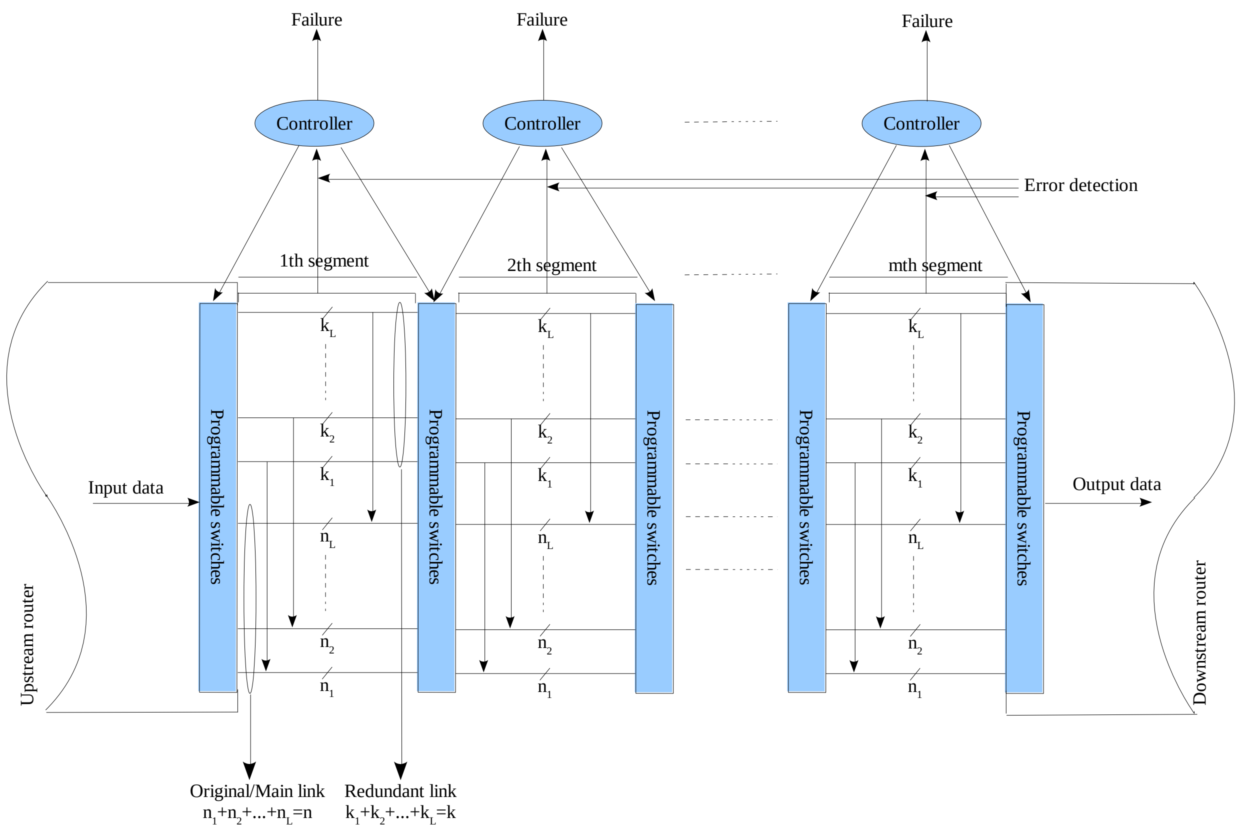


Figure 3.4. Block diagram of the proposed segmented self-repairing link. In each segment, the link is divided into groups of main wires and redundant wires.

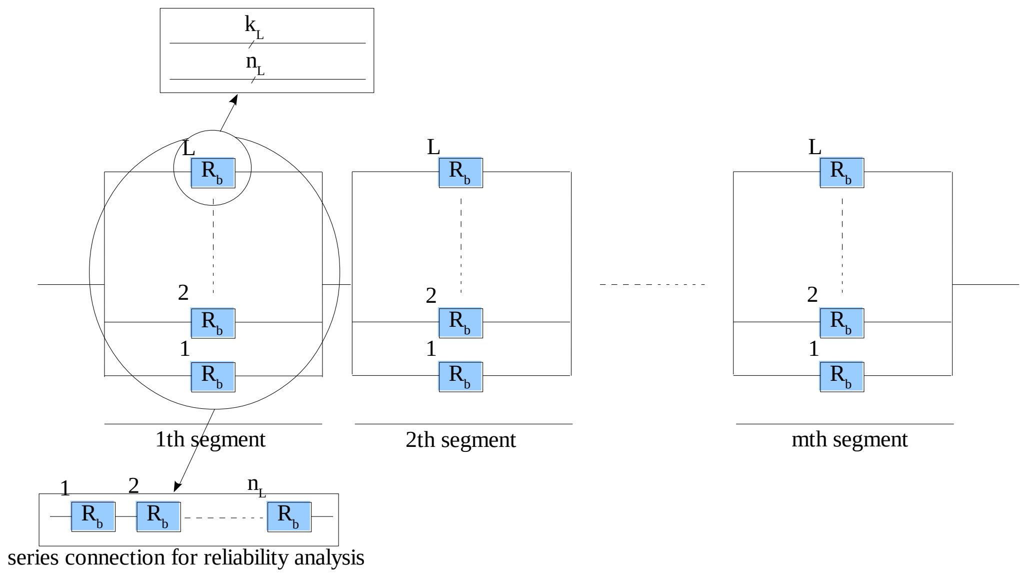


Figure 3.5. System level diagram utilized for reliability computation.

3.6. Fault detection

The operation of both non-segmented and segmented reconfigurable links depends on the ability to detect faulty wires. To detect faulty wires, we propose an effective FSM-based fault detection circuit. We assume that the NoC based system enters a test mode periodically. Fig. 6 shows the design of the fault detection circuit for one bit of the link. In this circuit, during normal operation, the test mode enable signal is 0, hence the link carries regular packets from the upstream router. When the system enters the test mode (test mode enable signal becomes 1) the link carries the test signal. Testing is done by sending over the link a pre-defined pulse test signal, which exercises each wire of the link with both low and high logic levels (to address stuck at low or high faults). At the receiving downstream router side, initially the error signal is set to 0 by the reset signal. As long as the test mode enable signal is 0 the error signal holds its value in state S0. When the system enters the test mode, the receiving downstream router expects to receive a 0 first and then a 1. This causes a transition from S0 to S1 and then a transition back to S0 while the error signal remains 0 indicating that the wire is healthy. However, any different signal received during the test mode, generates an error signal, which is set to 1. Once the system leaves the test mode (test mode enable = 0) the fault detection circuit holds the value of the final error signal. This error signal will be used by controllers during normal operation to decide if a wire needs to be replaced by a healthy redundant wire or not.

3.7. Simulation and experimental results

In this section we compare the proposed segmented link structure against the non-segmented link structure. Both link structures, shown in Fig.2.b and Fig.4, are coded in Verilog-HDL. The Verilog-HDL implementations are synthesized and simulated using Cadence tools [3.19] while the hardware validation is done using Xilinx ISE tools [3.20]. We use an controllers reconfigure the link to replace the faulty wire with the healthy redundant wire, thereby facilitating self-repairing. FPGA development board with a Virtex 5 FPGA. Numerical simulations to estimate reliability is done using Matlab [3.21].

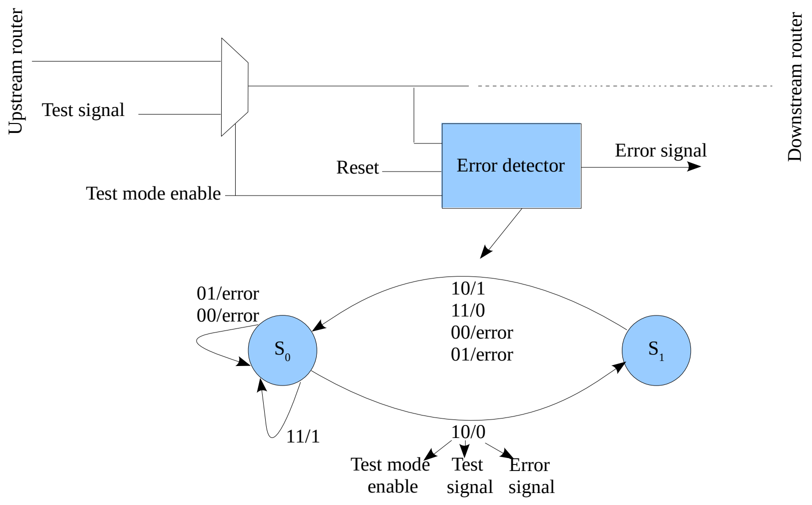


Figure 3.6. Block diagram of the error detection circuit for one bit of the link. Each wire of the link is equipped with an error detection circuit.

Our experiments reveal that in order to keep the area overheads within reasonable limits, the number of redundant wires k should be limited to only a few. For the same reason, the number of segments m for the segmented link should be a small number. In our implementation n = 64 main bits and k = 4 redundant bits. For the segmented link, the main link is divided into L = 4 groups (each group has n_i = 16 bits) with a single redundant bit, k_i = 1, designated to each group. We have implemented three different variants of the segmented link structure with m = 2, 3, 4.

Fig.7 shows a snap-shot of our simulation of the segmented link for m = 3. In this simulation, we inject a fault on the second segment of the last bit (bit index 63) of the main link. As can be seen, once the fault is detected, the error signal is set to logic high permanently. The controllers reconfigure the link to replace the faulty wire with the healthy redundant wire, thereby facilitating self-repairing.

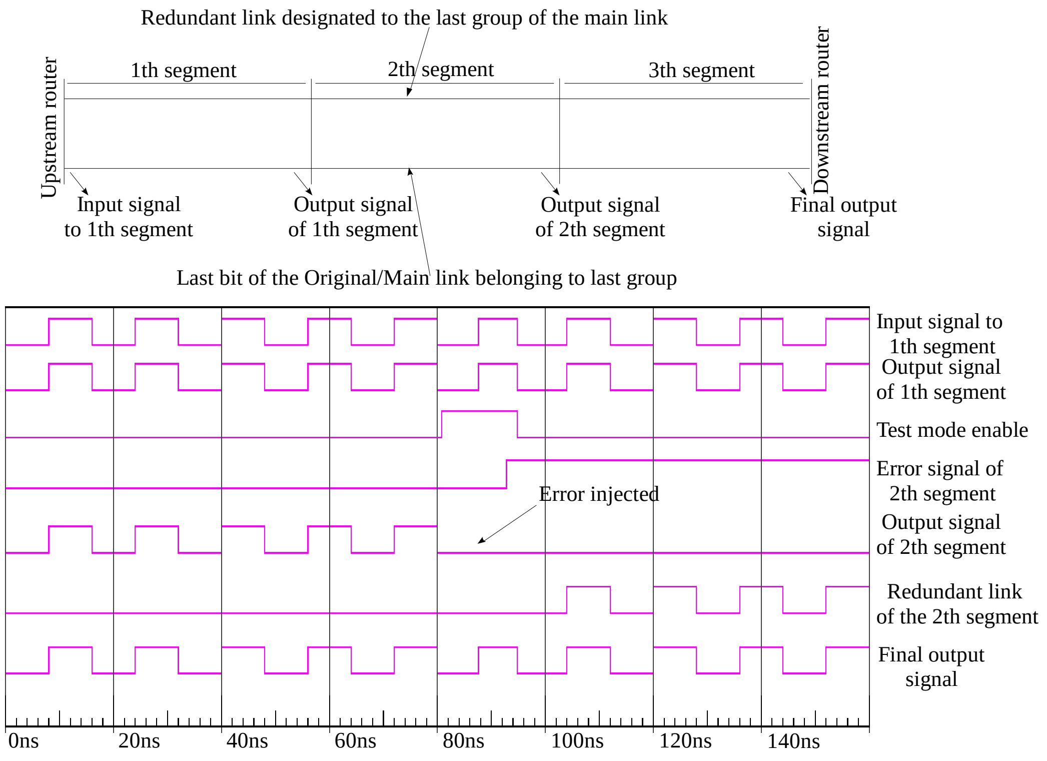


Figure 3.7. Simulation result that illustrates how a segmented link recovers a faulty bit of the link. A fault is injected on 63th bit of the main link.

3.7.1. Reliability

Numerical results based on the expressions of reliability derived in Section 3.5 show that the proposed segmented link non-segmented link. Fig.8 shows the percentage of increase in reliability for different values of the probability of wire failure. This result is intuitive and confirms that the more segments the link has, the higher its reliability (or fault tolerance) will be. However, as we will see shortly, the increase in the number of segments results also into an increase in area overhead, power consumption, and link delay.

3.7.2. Area

Fig.9 shows the percentage of increase in area occupied by the proposed segmented link structure compared to the nonsegmented link. Notably, the area occupied by the 2-segmented link structure is only 3.71% larger than area of the nonsegmented link. This is due to the considerable reduction in size of the programmable switches achieved via the proposed grouping of the main and redundant wires. Nevertheless, it can be seen that as the number of segments increases, the area overhead increases significantly compared to non-segmented link.

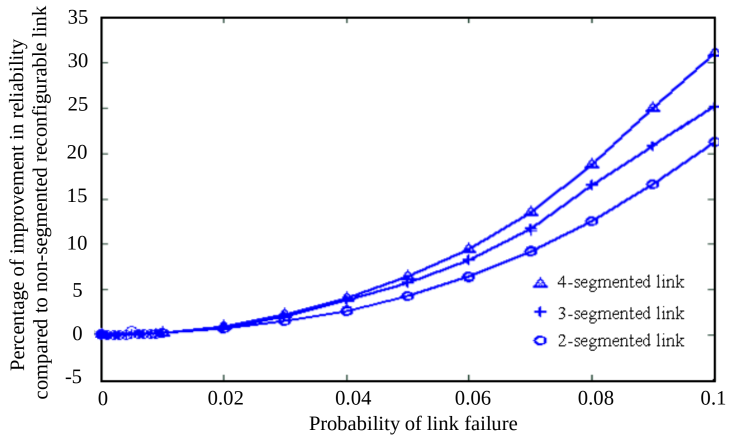


Figure 3.8. Percentage of increase in reliability achieved with the proposed segmented link compared to the non-segmented link.

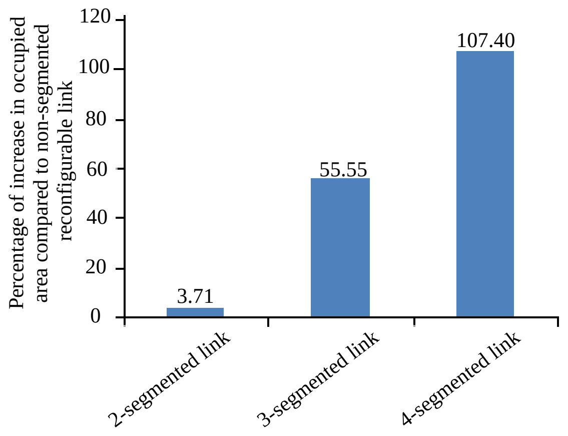


Figure 3.9. Percentage of increase in area occupied by the proposed segmented link compared to non-segmented link.

3.7.3. Power consumption

Fig.10 shows the comparison in terms of power consumption, estimated using Cadence tools. The power consumption of the 2-segmented link is 23.19% higher than that of the non-segmented link. It can be seen that as the number of segments increases, the amount of power consumption increases considerably too.

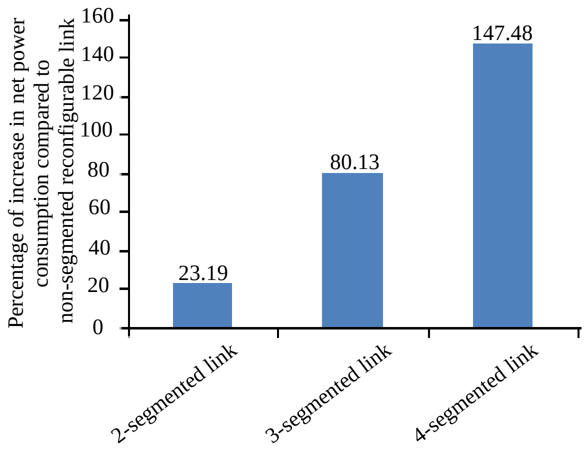


Figure 3.10. Percentage of increase in power consumption of the segmented link compared to non-segmented link.

3.7.4. Delay

Fig.11 shows the comparison in terms of link delay, as reported by Xilinx ISE tools. The increase in delay of the 2-segmented link is 16.41% compared to the non-segmented link. Similar to the cases of area and power comparisons, as the number of segments increases, the performance penalty increase considerably.

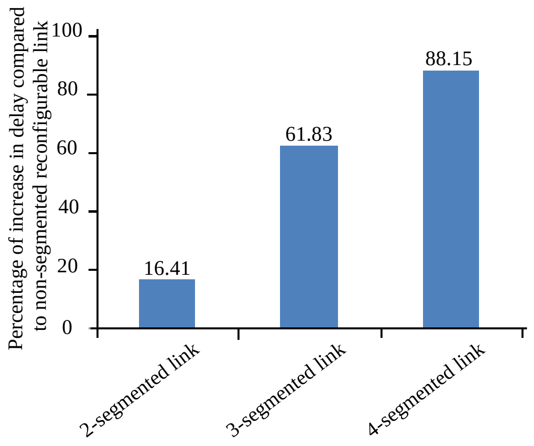


Figure 3.11. Percentage of increase in link delay of the segmented link compared to non-segmented link.

3.7.5. Discussion

It is clear that the best trade off between reliability improvement and penalty in area overhead, power consumption, and link delay is offered by the 2-segmented link structure. It is interesting to note that the concept of link segmentation for further improvement in fault tolerance can also be applied on top of the reconfigurable link designs presented in [3.6]–[3.8].

3.8. Conclusion

We proposed link segmentation and wire grouping as a novel technique to improve the fault tolerance against permanent faults of NoC links. Because different segments of the spare wires address different errors from different segments, the proposed reconfigurable link structure can tolerate a larger number of errors with a reduced number of spare wires. Cost effective fault detection circuits and segment level multiplexers and demultiplexers enable link reconfigurability, thereby self-repairing capability. Experimental results reveal that the 2-segmented link structure offers the best tradeoff between reliability improvement and penalty in area overhead, power consumption, and link delay.

3.9. References

[3.1] L. Benini and G. De Micheli, “Networks on chips: technology and tools,” Morgan Kaufmann, 2006.

[3.2] F.A. Bower, S. Ozev, and D.J. Sorin,“Automatic microprocessor execution via self-repairing arrays,” IEEE Trans. on Dependable and Secure Computing, vol. 2, no. 4, pp. 297-310, 2005.

[3.3] Z. Vasicek, L. Capka, and L. Sekanina, “Analysis of reconfiguration option for a reconfigurable polymorphic circuit,” NASA/ESA Conference on Adaptive Hardware and Systems, 2008.

[3.4] S. D. Mediratta and J. Draper, “Characterization of a fault-tolerant NoC router,” IEEE Int. Symposium on Circuits and Systems (ISCAS), 2007.

[3.5] D. Fick, A. DeOrio, J. Hu, V. Bertacco, D. Blaauw, and D. Sylvester, “Vicis: a reliable network for unreliable silicon,” ACM/IEEE Design Automation Conference (DAC), 2009.

[3.6] D. Fick, A. Deorio, G. Chen, D. Sylvester, and D. Blaauw, “A highly resilient routing algorithm for fault tolerant NoCs,” ACM/IEEE Design Automation and Test in Europe Conference (DATE), 2009.

[3.7] C. Feng, Z. Lu, A. Jantsch, J. Li, and M. Zhang, “A reconfigurable fault tolerant deflection routing algorithm based on reinforcement learning for network-on-chip,” Int. Workshop on Network-on-Chip Architectures (NocArc), 2010.

[3.8] H.S. Kia and C. Ababei, “A new fault-tolerant and congestion aware adaptive routing algorithm for regular Networks-on-Chip,” IEEE Congress on Evolutionary Computation (CEC), 2011.

[3.9] S. Pasricha and Y. Zou, “A low overhead fault tolerant routing scheme for 3D Networks-on-Chip,” IEEE Int. Symposium on Quality Electronic Design (ISQED 2011), 2011.

[3.10] D. Kim, K. Lee, S.J. Lee, and H.J. Yoo, “A reconfigurable crossbar switch with adaptive bandwidth control for Networks-on-Chip,” IEEE Int. Symposium on Circuits and Systems, 2005.

[3.11] B. Ahmad, A. T. Erdogan, and S. Khawam, “Architecture of a dynamically reconfigurable NoC for adaptive reconfigurable MPSoC,” NASA/ESA Conference on Adaptive Hardware and Systems, 2006.

[3.12] M. Palesi, S.i Kumar, R. Holsmark, and V. Catania, “Exploiting communication concurrency for efficient deadlock free routing in reconfigurable NoC platforms,” IEEE Int. Symposium on Parallel and Distributed Processing, 2007.

[3.13] R. Dafali, J. Ph. Diguet, and M. Sevaux, “Key research issues for reconfigurable Network-on-Chip,” Int. Conference on Reconfigurable Computing and FPGAs, 2008.

[3.14] M. B. Stensgaard and J. Sparso, “ReNoC: a Network-on-Chip architecture with reconfigurable topology,” ACM/IEEE Int. Symposium on Networks-on-Chip (NOCS), 2008.

[3.15] Q. Yu and P. Ampadu, “Transient and permanent error co-management method for reliable Networks-on-Chip,” ACM/IEEE Int. Symposium on Networks-on-Chip (NOCS), 2010.

[3.16] T. Lehtonen, D. Wolpert, P. Liljeberg, J. Plosila, and P. Ampadu, “Self-adaptive system for addressing permanent errors in on-chip interconnects,” IEEE Trans. on Very Large Scale Integration (VLSI) Systems, vol. 18, no. 4, 2010.

[3.17] T. Lehtonen, P. Liljeberg, and J. Plosila, “Online reconfigurable self-timed links for fault tolerant NoC,” VLSI Design, 2007.

[3.18] C. Ortega and A. Tyrrell, “Reliability analysis in self-repairing embryonic systems,” NASA/DoD Workshop on Evolvable Hardware, 1999.

[3.19] http://www.cadence.com.

[3.20] Xilinx ISE Tools, http://www.xilinx.com.

[3.21] http://www.mathworks.com/products/matlab.

CHAPTER 4. FAULT-TOLERANT AND CONGESTION-AWARE ADAPTIVE ROUTING ALGORITHM FOR REGULAR NETWORKS-On-Chip

This paper was presented in IEEE Congress on Evolutionary Computation (CEC), 2011. The authors of the paper are Hamed Sajjadi Kia and Cristinel Ababei.

4.1. Abstract

In this paper, we propose a new fault-tolerant and congestion-aware adaptive routing algorithm for Networks-on-Chip (NoCs). The proposed algorithm is based on the ball-and-string model and employs a distributed approach based on partitioning of the regular NoC architecture into regions controlled by local monitoring units. Each local monitoring unit runs a shortest path computation procedure to identify the best routing path so that highly congested routers and faulty links are avoided while latency is improved. To dynamically react to continuously changing traffic conditions, the shortest path computation procedure is invoked periodically. Because this procedure is based on the ball-and-string model, the hardware overhead and computational times are minimal. Experimental results based on an actual Verilog implementation demonstrate that the proposed adaptive routing algorithm improves significantly the network throughput compared to traditional XY routing and DyXY adaptive algorithms.

4.2. Introduction

The Network-on-Chip (NoC) concept replaces design specific global on-chip wires with a generic on-chip interconnection network realized by specialized routers that connect generic processing elements (PE). It represents a paradigm change from computation to communication centric design for Systems-on-Chip (SoCs) [4.1], [4.2]. The benefits of the NoC based SoC design include scalability, predictability, and higher bandwidth with support for concurrent communications.

Data are transferred between PEs organized as packets along paths computed by the routing algorithm. There are two types of routing strategies: deterministic and adaptive routing. In deterministic routing, the routing path is completely determined by the source and destination addresses. Its advantages include the simplicity of the router architecture and the deadlock free property. Due to the simpler hardware logic, deterministic routing offers lower flit latency when the NoC is not congested. However, as the packet injection rate increases and some of the links and routers become congested, deterministic routing is likely to suffer from throughput degradation as it cannot dynamically respond to network congestion [4.3]. In addition, permanent link failures may render the NoC inoperable. In contrast, adaptive routing takes into consideration the traffic variations in the network and computes dynamically alternative paths to route data via less congested regions. Moreover, if the NoC architecture is equipped with link failure detection mechanisms, adaptive routing can address these failures and thereby facilitate fault tolerance [4.4], [4.5]. However, due to the hardware overhead to implement the detection mechanisms and to compute good routing paths, adaptive routing has a higher latency at low levels of network congestion. Also, dynamic routing has to be designed so that it ensures deadlock free property.

4.3. Related work and contribution

Adaptive routing has attracted a lot of attention recently. The DyAD routing algorithm proposed in [4.6] is a hybrid approach, which switches between deterministic routing at low packet injection rates and dynamic routing when the network congestion increases. An adaptive routing architecture based on a dynamic programming (DP) network to provide optimal path planning is proposed in [4.7]. It has introduced a scalable k-step look ahead routing strategy to reduce routing tables storage and to maintain a high quality of adaptation. The routing method in

[4.8] utilizes information from all routers in the source-target path to perform traffic routing. The source units use the information on network conditions to adjust the parameters that configure the path to the target router. A dynamic routing algorithm based on monitoring the congestion status of the neighboring routers is studied in [4.3]. In [4.9] the objective is to route packets to their destination using a path that is as free as possible of congested nodes. The algorithm tries to use the situations of indecision occurring when the routing function returns several admissible output channels. In [4.10] a centralized monitoring system is used to locate congested links and detour them. However the proposed method is not scalable to the hundreds of cores that may soon be integrated on a SoC. Authors in [4.11] have proposed an adaptive routing scheme where intermediate routers make decisions locally depending on the available bandwidth in each direction to the neighboring routers and on the distance between current and the destination routers. A congestion aware routing algorithm, which sends congestion monitoring values in parts of the network beyond adjacent routers, is proposed in [4.12].

Several papers focusing on fault tolerant routing algorithms have recently been published. Reconfigurable architectures have been employed in several papers to address faults. The Vicis NoC architecture proposed in [4.13] can tolerate the loss of routers and links due to wearout induced hard faults. Network level reconfiguration is implemented by rewriting the routing tables based on the information from the built-in-self-test (BIST) units in each router. A reconfigurable fault-tolerant deflection routing algorithm based on reinforcement learning for NoC has been proposed in [4.14]. The algorithm reconfigures the routing tables through reinforcement learning based on 2-hop fault information. In [4.15] a routing algorithm that boosts the robustness of interconnect networks by reconfiguration to avoid faulty components while maintaining connectivity and correct operation has been proposed. A lightweight fault tolerant mechanism based on the notion of default backup paths (DBPs) has been proposed in [4.16]. It uses nominal redundancy to maintain network connectivity of healthy NoC routers and on-chip PEs in the presence of hard failures. Most previous works on adaptive routing report simulation results achieved with simulators developed in a programming language (e.g., C++, SystemC). Therefore, they typically do not report actual area overheads due to the lack of actual hardware implementation details. In addition, they address either congestion issues or errors (e.g., transient, intermittent, permanent failures).

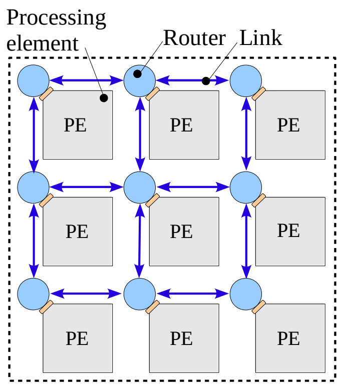
In this paper, we develop a new distributed adaptive algorithm designed to address both congestion and link failures. Our main contribution is as follows: (i) We develop a new NoC architecture which partitions the regular NoC architecture into regions controlled by local monitoring units. Each local monitoring unit runs a shortest path computation procedure to identify the best routing path so that highly congested routers or failed links are avoided, (ii) We propose the use of a ball-and-string model based shortest path computation method, which together with the decentralized region based routing approach leads to minimal hardware overhead, and (iii) The proposed NoC architecture and routing strategy are implemented in Verilog with Virtex 5 as the target FPGA fabric. The experimental results on multimedia benchmarks demonstrate the ability of the proposed routing algorithm to significantly improve the network throughput.

4.4. Proposed adaptive routing

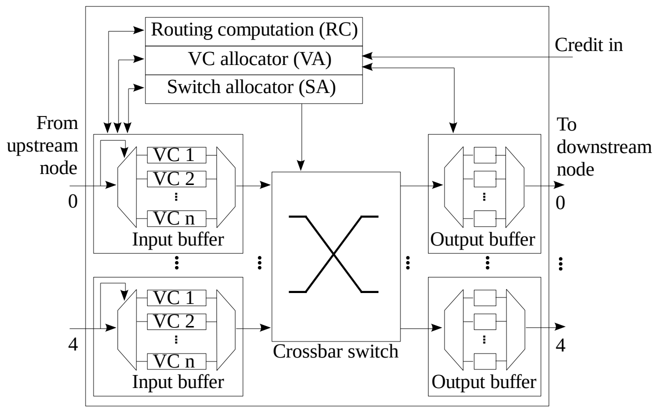
In this section we describe the proposed dynamic routing algorithm based on the ball-and-string model for regular mesh NoC topologies.

4.4.1. General NoC topology description

For simplicity, we assume a regular mesh NoC topology to describe and apply the proposed dynamic routing algorithm. However, the proposed routing algorithm can also be extended to irregular NoCs. Regular mesh NoCs are 2D arrays of routers. Adjacent routers are connected via bi-directional links or channels. An example of a 3x3 mesh NoC is shown in Fig. 4.1.a. The router has a pipelined architecture where routing computation (RC), virtual channel allocation (VA) and switch allocation (SA), and switch traversal (ST) are the main pipeline stages. The block diagram of the router is shown in Fig. 4.1.b. To minimize the required buffering space, in this paper we assume wormhole switching. The router architecture will be modified to add support for the adaptive routing - this will be described in a later section.



(a)



(b)

Figure 4.1. (a) Example of 2D regular mesh. (b) Typical router architecture.

4.4.2. Ball-and-string model based shortest path computation procedure

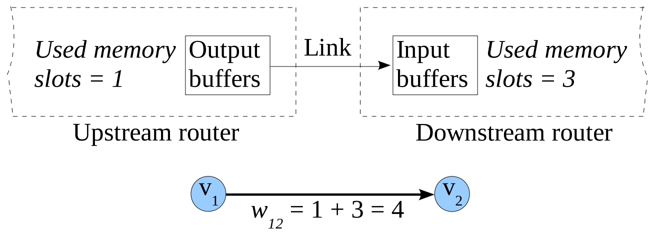
The main idea of shortest paths computation is (1) to associate a directed graph G(V,E) with the NoC topology, (2) to assign edge weights proportionally to congestion, and (3) to develop a procedure to find the shortest paths in this graph for any given source node. To compute edge weights we propose to use buffer occupancies, which are readily available in a typical NoC. More specifically, the weight is computed as the summation of the numbers of memory slots used in the output buffers of the upstream router and of memory slots used in the input buffers of the downstream router. For example, Fig. 4.2.a illustrates the computation of the edge weight of an individual link. Fig. 4.2.b shows the edge weights for a graph associated with a 3x2 NoC. Formally, for the purpose of computing the shortest paths the edge weights w_{ij}, i = 1,…,|V|, j = 1,…,|V| are computed as follows:

$$w_{ij} = \begin{cases} Used\ memory\ slots & if\ (v_i, v_j)\varepsilon E,\ \forall v_i, v_j \varepsilon V \\ \infty & otherwise \end{cases} \tag{4.1}$$

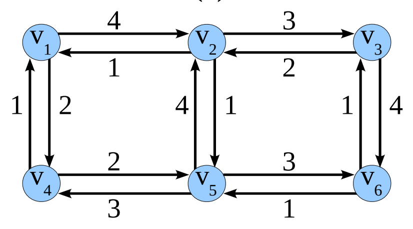
Each time the shortest path procedure is invoked, the shortest path for each source-destination communication pair will be computed so that the path cost is minimized:

$$Min\colon \sum_{i=1}^{|V|}\sum_{j=1}^{|V|} y_{ij}\ x\ w_{ij} \tag{4.2}$$

where y_{ij} is a binary variable, which indicates if the link (v_i, v_j) ε E is used or not as a part of the path. The procedure for the shortest path computation is based on the parallel shortest path searching algorithm proposed in [4.17], which is similar to the ball-and-string model studied in [4.18], [4.19]. The shortest path procedure will identify the best paths for packets to travel to their destinations under the congestion conditions that exist at the time of edge weights computation. To account for the changes in these conditions (due to the changes in network traffic) the procedure will be called periodically multiple times. Therefore, it is important for the implementation of such a procedure to be fast and to require minimal hardware resources. Our custom implementation of this algorithm utilizes the adjacency matrix $[A]_{nxn}$ of the network graph - referred to as the network matrix because each entry stores the corresponding edge weight (i.e., $a_{ij} = w_{ij}$). In addition, it utilizes a specialized array - referred to as the parent-array - which stores the IDs of predecessor node (or the parent node) of each node along the shortest path. For example, the network matrix and the parent array initialized to zero of the network graph from Fig. 4.2.b are shown in Fig. 4.2.c.



(a)



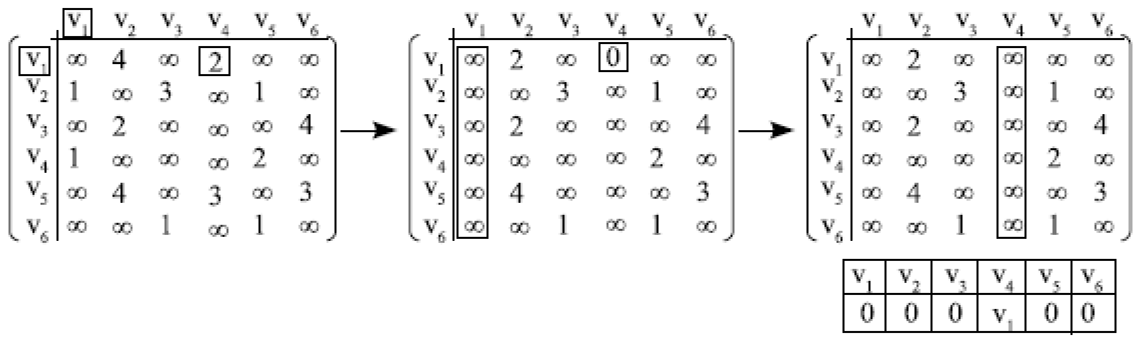
(b)

$$\begin{array}{c|cccccc} & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \\ \hline v_1 & \infty & 4 & \infty & 2 & \infty & \infty \\ v_2 & 1 & \infty & 3 & \infty & 1 & \infty \\ v_3 & \infty & 2 & \infty & \infty & \infty & 4 \\ v_4 & 1 & \infty & \infty & \infty & 2 & \infty \\ v_5 & \infty & 4 & \infty & 3 & \infty & 3 \\ v_6 & \infty & \infty & 1 & \infty & 1 & \infty \end{array}$$

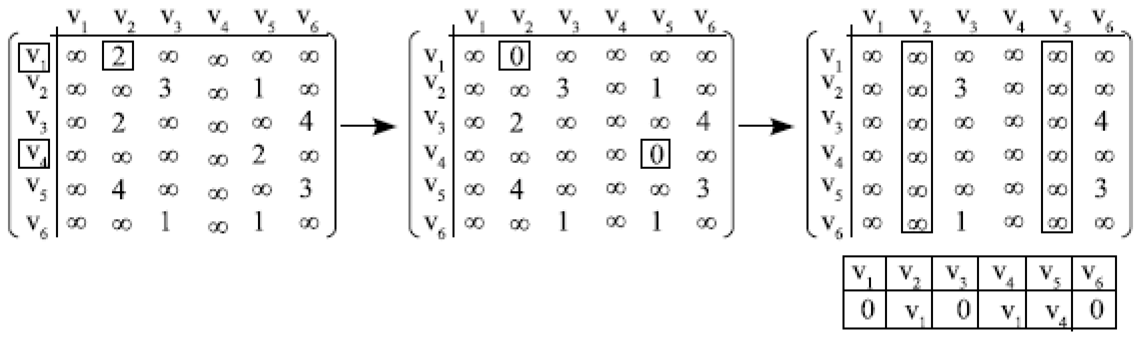
v_1	v_2	v_3	v_4	v_5	v_6
0	0	0	0	0	0

(c)

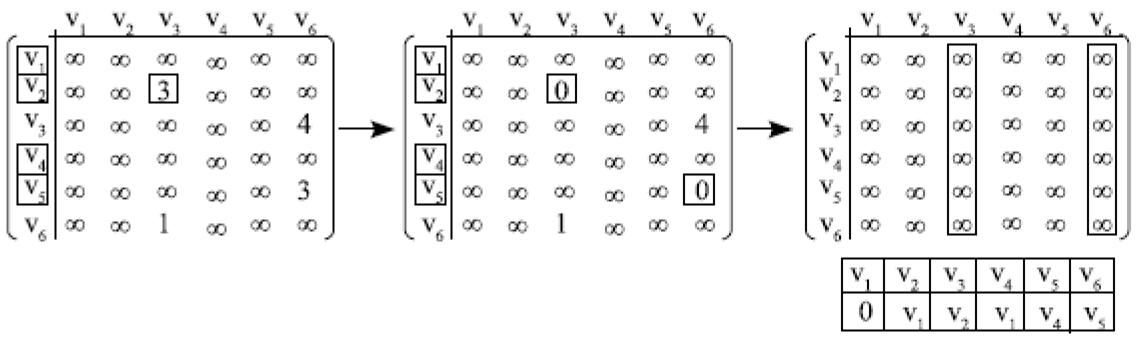
Figure 4.2. (a) Computation of edge weight. (b) Edge weights for a network graph associated with a 3x2 NoC. (c) The network matrix A and the parent-array of the network graph.



(a)



(b)



(c)

Figure 4.3. Applying the algorithm to network matrix.

To illustrate how the algorithm works, we use the example from Fig. 4.2.b. Let us assume node v_1 as the source. In the first step of the algorithm, all entries in the first column of the network matrix are set to infinity. Also, the minimum value in the first row is found and then subtracted from each entry of the first row (see Fig. 4.3.a). Because $a_{14} = 0$, the shortest path to v_4 is already determined and v_1 is recorded in the fourth column of the parent-array. Then, all entries in the fourth column of the network matrix are also set to infinity as shown in Fig. 4.3.a. In the next step of the algorithm, the minimum value among the entries of the first and fourth rows is identified and subtracted from the entries of these rows as shown in Fig. 3.b. Because $a_{12} = 0$ and $a_{45} = 0$, the shortest paths to v_2 and v_5 are also determined at this time and these two nodes are recorded in the second and fifth columns of the parent-array. Also, all entries in the second and fourth columns are set to infinity. This process is repeated until all entries in the network matrix are set to infinity. At this time, all entries in the parent-array store the predecessors of each node, which can be back traced to construct the shortest path from the source v_1 to any node in the graph. Fig. 4.4 illustrates how the network matrix and the parent array are implemented. To minimize the memory usage for storing and manipulating matrices, in our actual hardware implementation (described in detail in a later section) we use registers to store only the entries that are initially non-infinity in the network matrix. In other words, for example in Fig. 4.2 we know that a_{16} and a_{61} will remain equal to infinity throughout the shortest paths computation process - due to the regular mesh NoC topology which tells us that there is no direct connection between nodes v_1,v_6. Therefore, there is no need to allocate and manipulate memory for these entries of the network matrix. However, we need to use two additional registers shown as “Fixed” and “Flag” in Fig. 4.4. Each time when the entries of a given column must be set to infinity only the corresponding entry of the “Fixed” register is set to 1. This eliminates the need for the infinity value, which is difficult to define in a simple hardware implementation. The “Flag” register is used to mark the rows which are processed currently. For example, once it is known that the first and fourth rows are to be processed next, their corresponding entries in the “Flag” register are set to 1. In our hardware implementation, even though the design time was slightly longer this custom implementation of the network matrix reduced significantly the memory usage.

Using the additional registers, the pseudocode of the shortest path algorithm is presented in Algorithm 1.

```
Algorithm 1: Shortest path computation
1: $\forall i, Fixed(i) \leftarrow 1, Flag(i) \leftarrow 1$ if node $i$ is source, 0 otherwise
2: $[Parent]_{1\times n} \leftarrow [0]_{1\times n}$
3: Initialize network matrix $[A]_{n\times n}$ based on network graph
4: while $(Flag \neq [1]_{1\times n})$ do
5:     for $i \leftarrow 1$ to $n$ do
6:         for $j \leftarrow 1$ to $n$ do
7:             $A(i,j)^* = Flag(i) \times Fixed(j)' \times A(i,j)$
8:         end for
9:     end for
10:    Find the non-zero minimum of $A^*(i,j)$
11:    for $i \leftarrow 1$ to $n$ do
12:        for $j \leftarrow 1$ to $n$ do
13:            if $Flag(i) = 1$ then
14:                $A(i,j) = A(i,j) - minimum$
15:            end if
16:            if $A(i,j) = 0$ then
17:                $Flag(i) = 0, Fixed(j) = 0, Parent(j) = i$
18:            end if
19:        end for
20:    end for
21: end while
```

Figure 4.4. The pseudocode of the shortest path computation procedure.

4.4.3. Adaptive routing

To minimize the required extra hardware we propose a distributed (or decentralized) scheme for the implementation of the adaptive routing. The NoC is partitioned into several partitions (or regions) and each partition is managed by a local monitoring unit (LMU). LMUs represent the controllers responsible with routing packets that enter their partitions. For example, the 4x4 NoC from Fig. 4.6 is partitioned into four equal regions controlled by four LMUs. Even though in this example the partitions have equal size, they may have different sizes too.

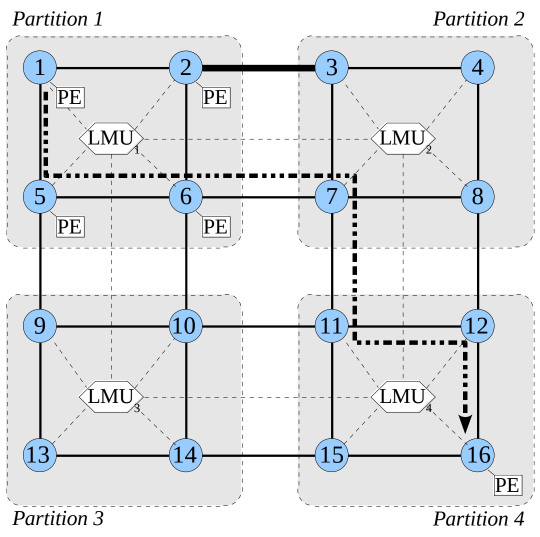


Figure 4.5. NoC partitioned into four partitions controlled by four LMUs.

Each LMU is in charge with routing data to routers within its own partition and to the first-order neighboring routers adjacent to its own partition. To compute edge weights for links that connect routers from different partitions, adjacent LMUs are interconnected to be able to share information. For example in Fig. 4.6, LMU_1 is responsible for routing packets injected within the partition formed by the routers {1, 2, 5, 6} and the packets that arrive from adjacent routers {3, 7, 9, 10}. This LMU will implement the shortest path computation procedure described in the previous section, which will utilize the network matrix of the sub-graph corresponding to routers {1, 2, 3, 5, 6, 7, 9, 10} and all edges between these routers except the two edges corresponding to the links between routers {3, 7} and {9, 10}.

As an example consider a situation when a packet arrives to router 2 in partition 1 via the boundary-crossing link between routers {3, 2} (shown in thicker line in Fig. 4.6). In this case, the source node in the shortest path procedure will correspond to router 2. LMU_1 will extract the destination address from the header flit. If the destination router is located inside partition 1 or is one of the adjacent routers {9, 10}, then LMU_1, which has already computed the shortest paths, will update the header flit (Fig. 4.9) with the shortest path routing information. If the destination is in partition 3, then the header flit will be updated with the routing information toward one of the routers {9, 10} - to the one with the shortest path - and packets will be routed accordingly. Then, LMU_3 will be responsible with routing to the final destination inside partition 3.

As another example, let us consider the source-destination pair v_1, v_{16}. Because the destination is in partition 4, the algorithm will first find the shortest path to either of the routers {3, 7, 9, 10} in partitions 2 and 3. Assuming that the shortest path is to router 7, the packets may be routed as shown in Fig. 4.6. Then, LMU_2 will be responsible with routing packets toward partition 4. This will be done by utilizing the shortest path from the source 7 to either of the routers {11, 12}. If this path is to the router 11 as shown in Fig. 4.6, then LMU_4 will be responsible with routing along the shortest path from the local source 11 to the final destination 16.

4.4.4. Addressing link failures

As CMOS fabrication technologies move to nano-scale feature sizes, integrated circuits become more susceptible to manufacturing faults, transient faults, and aging mechanisms that can lead to permanent faults. In NoC architectures without fault tolerance mechanisms, permanent link failures can render the NoC inoperable. The adaptive algorithm proposed in this paper, can easily address link failures and thereby facilitate fault tolerance. When a link failure is detected inside a given partition, the corresponding LMU can remove that link from the adjacency matrix. Hence, the shortest path computation procedure will compute thereafter paths formed by the remaining healthy links only.

4.4.5. Deadlock

Deadlock occurs when packets are unable to move forward because they are waiting on one another to release resources (i.e., there is a cyclic dependency between packets). This is undesirable because it can paralyze the operation of the network. Therefore, routing algorithms must be designed so that deadlock is avoided [4.1]. While the proposed routing algorithm is not designed to directly guarantee the deadlock free property, it indirectly minimizes the likelihood of deadlock occurrence. If a deadlock situation occurs, the affected links (which do not see activity for long periods of time) can be interpreted as if they were congested or broken. Because, the proposed adaptive algorithm is called periodically, the new computed routing paths will avoid the affected links, thereby most likely eliminating the deadlock situation. In other words, in the event that a packet dependency occurs, it will be eliminated during the next call of the shortest path computation procedure, which will find a different path (using other links) along which packets can move forward.

4.5. Hardware implementation

The implementation of the adaptive routing requires changes in the NoC architecture. First, we added the local monitoring units and their connections as discussed in the previous section and as illustrated in Fig. 4.7. Second, we designed a new router architecture to provide support for the mechanics of the proposed routing algorithm as described below.

4.5.1. Modified router architecture

Because we use input and output buffers occupancies to compute edge weights, we modify the input and output buffers by adding local input and output control units as shown in Fig. 4.8. To minimize the router area, all buffers are implemented using registers instead of SDRAM structures. Input and output ports use 2 virtual channels. Messages are divided into packets, which are further divided into 3 flits (header, body, and tail). The header flit contains the routing information and destination address. Fig. 4.9 shows the format of the header flit. Once a header flit arrives to the input port of a given router, the routing information (i.e., the output port ID to which the flit should be forwarded to) is provided by the local monitoring unit. When the header flit is received the routing bits are shifted as shown in Fig. 4.9. Routers use the routing data bits to determine the output port. In this way the usage of routing tables is avoided. After gaining access to the output port and before the transfer to the output buffer is started, the input control unit (ICU) saves the destination port ID in the port ID control bits (see Fig. 4.8.a) where it will be stored until after the tail flit of the same packet will traverse this router. The ICU also sets the “Port request” bit whenever a flit requests access to any output port.

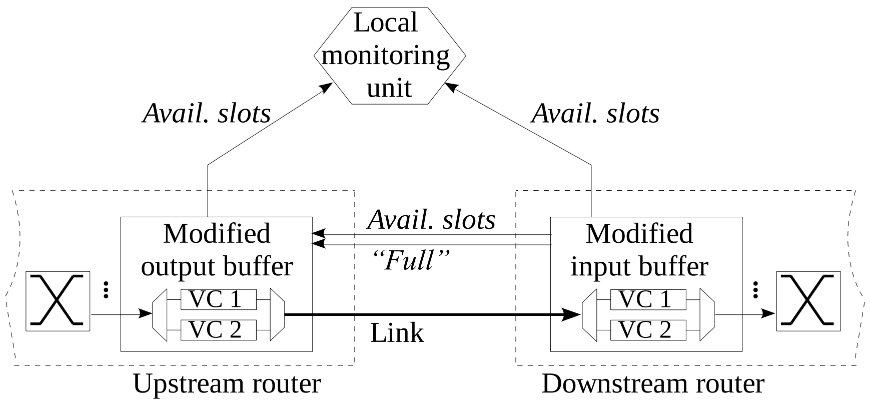
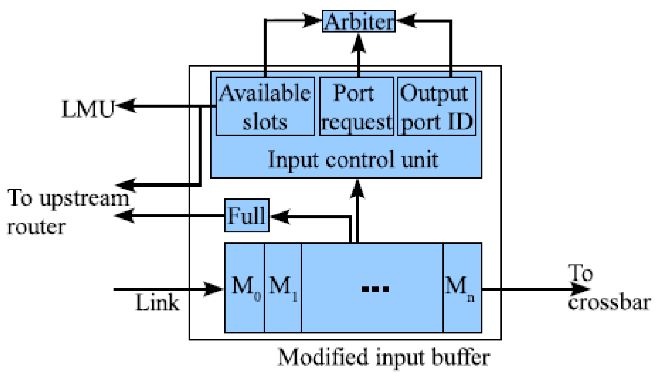
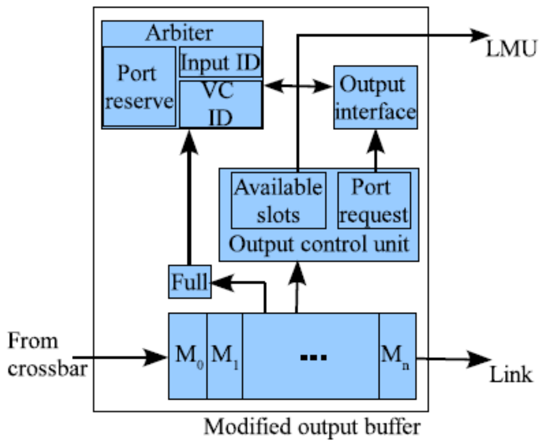


Figure 4.6. Block diagram of the communication between two adjacent routers.



(a)



(b)

Figure 4.7. (a) Block diagram of the input buffer. (b) Block diagram of the output buffer.

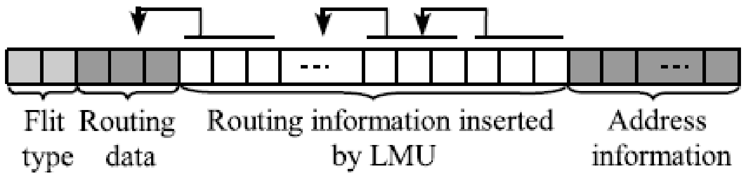


Figure 4.8. Header flit description.

At each positive edge of the clock the ICU computes the number of occupied slots in the input buffer. This information is sent to LMU, upstream router and arbiter. The “Full” bit (see Fig. 4.8.a) is set to 1 when there is at least an empty slot available and this information is communicated to the upstream router. Whenever there is an empty slot in the output buffers, the output control unit (OCU) sets the “Full” bit in Fig. 4.8.b and sends this info to the arbiter, which continuously monitors the output buffers. If an empty slot is available in the output buffer the arbiter will check if it is reserved or not. When both virtual channels of an output port are available the arbiter will select and grant access first the one with more empty slots. When the output buffer receives a header flit the arbiter will set the reserved bit to logic 1, which will be kept until after the tail flit will be received. The OCU also computes the number of used buffer slots (i.e., the output buffer occupancy), which is sent to the local monitoring unit, arbiter, and output interface unit. In addition, OCU also sets the “Request port” bit whenever a flit requests access to physical link. This bit is continuously monitored by the output interface unit. At each positive edge of the clock the ICU computes the number of occupied slots in the input buffer. This information is sent to LMU, upstream router and arbiter. The “Full” bit (see Fig. 4.8.a) is set to 1 when there is at least an empty slot available and this information is communicated to the upstream router. Whenever there is an empty slot in the output buffers, the output control unit (OCU) sets the “Full” bit in Fig. 4.8.b and sends this info to the arbiter, which continuously monitors the output buffers. If an empty slot is available in the output buffer the arbiter will check if it is reserved or not. When both virtual channels of an output port are available the arbiter will select and grant access first the one with more empty slots. When the output buffer receives a header flit the arbiter will set the reserved bit to logic 1, which will be kept until after the tail flit will be received. The OCU also computes the number of used buffer slots (i.e., the output buffer occupancy), which is sent to the local monitoring unit, arbiter, and output interface unit. In addition, OCU also sets the “Request port” bit whenever a flit requests access to physical link. This bit is continuously monitored by the output interface unit.

The output interface unit shown in Fig. 4.8.b functions as an arbiter. A packet in the output buffer will be sent to a virtual channel with more empty slots in the input port of the downstream router. When both output virtual channels compete over the physical link, the output interface unit will select and grant access first to the VC that has the least available memory. When the input buffer of the downstream router receives a header flit it will be marked as being reserved.

The shortest path computation requires eight clock cycles. Therefore, every other eight clock cycles the LMUs update the shortest paths. This period is small enough to ensure rapid response to changes in traffic as observed in our experiments.

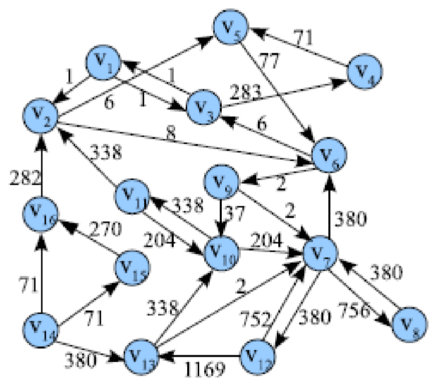
4.6. Experimental results

To validate and test the proposed adaptive algorithm, we have coded in Verilog a 4x4 NoC prototype with an architecture similar to that shown in Fig. 4.6. The NoC design is synthesized using the Xilinx ISE compiler [4.22] and the RTL implementation is verified via dynamic simulation. The target hardware platform is a Virtex 5 FPGA. The ISE tool is utilized to estimate total area. The static timing analysis feature of the ISE tool is used to measure and compute the average flit latency.

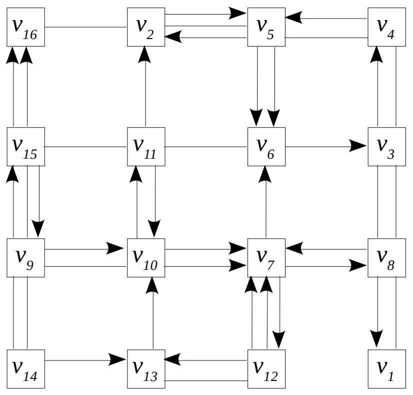
4.6.1. Adaptive routing to address

In the first set of experiments, we compare the proposed adaptive routing algorithm against the traditional XY routing. We also compare the proposed routing algorithm against DyXY adaptive algorithm [4.3] due to its popularity and ease of implementation. Even though DyXY was studied only based on simulations, we have implemented it in Verilog using our own adapted router architecture. The hardware implementation of other previously proposed adaptive routing algorithms is not available. Moreover, because of the complexity of these adaptive algorithms, their Verilog implementation is very challenging. Therefore, we restrict our experiments to comparisons against XY and DyXY algorithms. We report our results for two multimedia benchmarks. The communication task graph (CTG) and optimized mapping of the first benchmark are from [4.23] and are shown in Fig. 4.10. The injected traffic at all sources of the CTG is generated by local generators. The average number of injected packets at each source is proportional to the communication volume of each source-destination pair shown in Fig. 4.10.a. The average latency is computed under the assumption that packets are consumed immediately upon arrival to their destinations. Fig. 4.11 presents the average latencies achieved using the proposed adaptive routing algorithm, the traditional XY routing algorithm, and DyXY algorithm respectively. It can be observed that the proposed adaptive routing improves the network throughput at high packet injection rates. However, at low packet injection rates latency is slightly degraded compared to XY routing due to the delay penalty incurred in the hardware for adaptive routing support.

The communication task graph (CTG) and optimized mapping of the second benchmark are from [4.24] and are shown in Fig. 4.12. Again the proposed adaptive routing improves the network throughput at high packet injection rates (Fig. 4.13). The improvement in throughput and the extra hardware needed to implement the proposed algorithm is shown in Table 4.1. Throughput is defined at the point where the latency is twice as the low packet injection rate latency. The extra hardware to implement the proposed algorithm is around 17%, which is less than 42% of [4.13]. The area penalty is slightly higher than the area penalty for the DyXY routing algorithm. However, note that DyXY routing algorithm is designed to address only congestion, while the proposed routing algorithm addresses both congestion and link failures for fault tolerance.



(a)



(b)

Figure 4.9. CTG and optimized mapping of the first multimedia benchmark.

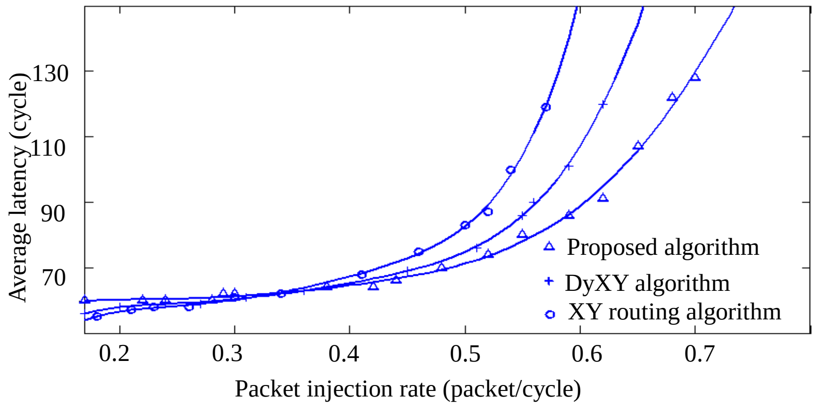
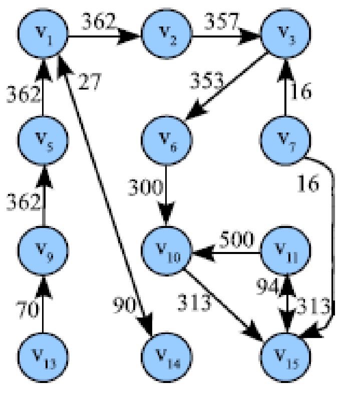
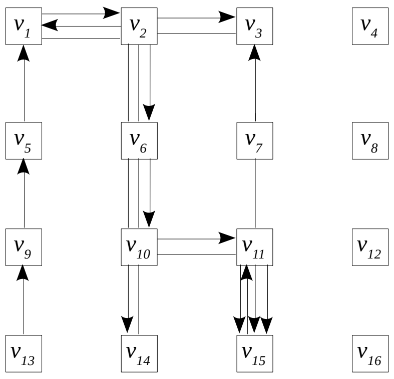


Figure 4.10. Comparison of the average latency achieved by the proposed adaptive routing, the traditional XY routing algorithm and DyXY algorithm for the first benchmark.



(a)



(b)

Figure 4.11. CTG and optimized mapping of the second multimedia benchmark.

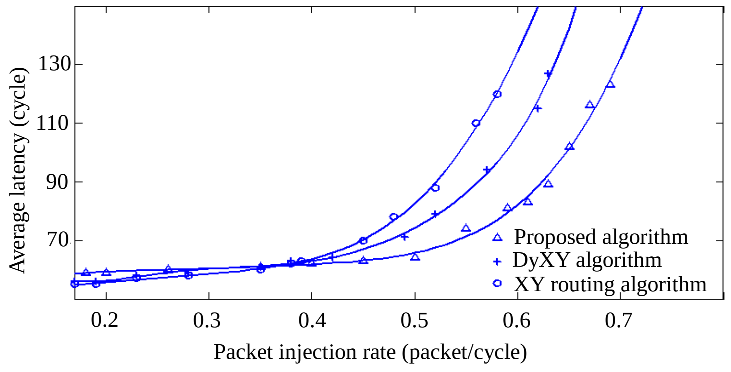


Figure 4.12. Comparison of the average latency achieved by the proposed adaptive routing and the traditional XY routing algorithms for the second benchmark.

4.6.2. Adaptive routing to address link failures

In this section we investigate the performance of the proposed adaptive routing algorithm in the presence of link failures. We investigate the fault tolerance of the proposed routing algorithm for a number of injected link failures varied between and 4. For each of these numbers, we randomly inject link failures several times and then we compute the average throughput. To keep the CPU computational runtimes of ISE tool within reasonable limits we study a simpler testcase whose mapping is shown in Fig. 4.14. Each of the injected set of faults are handled by the proposed algorithm as described in Section 4.3.5. The network throughput degradation as a function of the number of injected faults is shown in Fig. 4.15. It can be noted that the network throughput degrades gracefully, which demonstrates the ability of the proposed routing algorithm to address link failures.

Table 4.1. Comparison against XY routing

Routing algorithm	Extra hardware	Test Case	Throughput improvement
Proposed algorithm	17 %	Test case 1	21%
		Test case 2	20%
DyXY algorithm	12 %	Test case 1	11%
		Test case 2	10%

4.7. Conclusion

We proposed a new fault-tolerant and congestion-aware adaptive routing algorithm for NoCs. To implement the proposed algorithm the NoC architecture is partitioned into regions controlled by local monitoring units. Each local monitoring unit runs a shortest path computation procedure to identify the best routing path so that highly congested routers are avoided. To dynamically react to continuously changing traffic conditions the procedure is invoked periodically. Because the procedure is based on the ball-and-string model, the hardware overhead and computational times are minimal. Experimental results based on an actual Verilog implementation demonstrate that the proposed adaptive routing algorithm improves significantly the network throughput compared to traditional XY routing and DyXY adaptive algorithms.

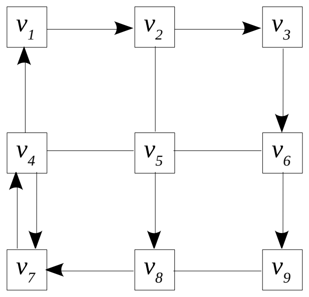


Figure 4.13. The mapping of the third benchmark.

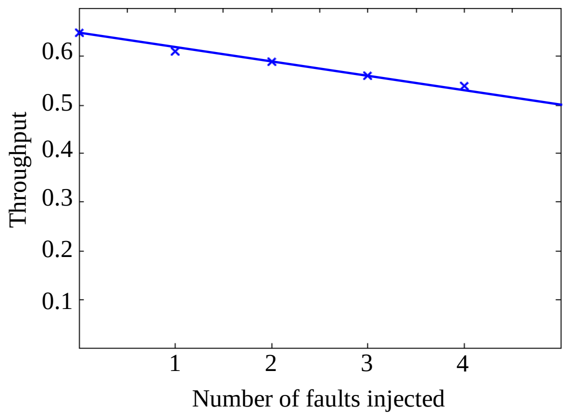


Figure 4.14. Throughput depredation for different amount of fault injection.

4.8. References

[4.1] W. J. Dally, and B. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann, 2004.

[4.2] G.D. Micheli, and L. Benini, Networks on Chips: Technology and Tools, Morgan Kaufmann, 2006.

[4.3] M. Li, Q.A. Zeng, and W.B. Jone, “DyXY: a proximity congestion aware deadlock-free dynamic routing method for network on chip,” ACM/IEEE Design Automation Conference (DAC), 2006.

[4.4] R. Marculescu, “Networks-on-chip: the quest for on-chip fault-tolerant communication,” IEEE Computer Society Annual Symposium on VLSI, 2003.

[4.5] M. Yang, T. Li, Y. Jiang, and Y. Yang, “Fault tolerant routing schemes in RDT(2,2,1)/a-based interconnection for networks on chip designs,” Int. Symposium on Parallel Architectures, Algorithms and Networks, 2005.

[4.6] J. Hu and R. Marculescu, “DyAD: smart routing for networks-on-chip,” ACM/IEEE Design Automation Conference (DAC), 2004.

[4.7] T. Mak, P.Y.K. Cheung, W. Luk, and K.P. Lam, “A DP-network for optimal dynamic routing in Network-on-Chip,” ACM/IEEE Int. Conference on Hardware Software Codesign, 2009.

[4.8] L. Tedesco, F. Clermidy, and F. Moraes, “A path-load based adaptive routing algorithm for Networks-on-Chip,” ACM Annual Symposium on Integrated Circuits and System Design, 2009.

[4.9] G. Ascia, V. Catania, M. Palesi, and D. Patti, “Implementation and analysis of a new selection strategy for adaptive routing in networks on chip,” IEEE Trans. on Computers, vol. 57, no. 6, pp. 809-820, 2008.

[4.10] F. Ge, N. Wu, and Y. Wan, “A network monitor based dynamic routing scheme for network on chip,” IEEE Asia Pacific Conference on Microelectronics and Electronics, 2009.

[4.11] M.A. Al Faruque, T. Ebi, and J. Henkel, “Run-time adaptive on chip communication scheme,” ACM/IEEE Int. Conference on Computer Aided-Design (ICCAD), 2007.

[4.12] P. Gratz, B. Grot, and S.W. Keckler, “Regional congestion awareness for load balance in networks-on-chip,” IEEE Int. Symposium on High Performance Computer Architecture, 2008.

[4.13] D. Fick, A. DeOrio, J. Hu, V. Bertacco, D. Blaauw, and D. Sylvester, “Vicis: a reliable network for unreliable silicon,” ACM/IEEE Design Automation Conference (DAC), 2009.

[4.14] C. Feng, Z. Lu, A. Jantsch, J. Li, and M. Zhang, “A reconfigurable fault tolerant deflection routing algorithm based on reinforcement learning for network-on-chip,” Int. Workshop on Network on Chip Architectures (NocArc), 2010.

[4.15] D. Fick, A. Deorio, G. Chen, D. Sylvester, and D. Blaauw, “A highly resilient routing algorithm for fault tolerant NoCs,” ACM/IEEE Design Automation and Test in Europe Conf. (DATE), 2009.

[4.16] M. Koibuchi, H. Matsutani, H. Amano, and T.M. Pinkston, “A lightweight fault tolerant mechanism for Network-on-Chip,” ACM/IEEE Int. Symposium on Networks-on-Chip (NoCS), 2008.

[4.17] H. Ishikawa, S. Shimizu, Y. Arakawa, N. Yamanaka, and K. Shiba, “New parallel shortest path searching algorithm based on dynamically reconfigurable processor DAPDNA-2,” IEEE Int. Conference on Communications, 2007.

[4.18] P. Narvaez, K.Y. Siu, and H.Y. Tzeng, “New dynamic SPT algorithm based on a ball-and-string model,” ACM/IEEE Trans. on Networking (TON), vol. 9, no. 6, pp. 706-718, 2001.

[4.19] T. Shi and J.J. Lee, “An O(L) parallel shortest path algorithm,” Int. Conference on Computer Design (CDES), pp. 119-124, 2009.

[4.20] A.D. Choudhury, G. Palermo, C. Silvano, and V. Zaccaria, “Yield enhancement by robust application-specific mapping on Network-on-Chips,” Int. Workshop on Network on Chip Architectures (NocArc), 2009.

[4.21] C. Seiculescu, S. Murali, L. Benini, G. De Micheli, “A method to remove deadlocks in Networks-on-Chips with wormhole flow control,” ACM/IEEE Design Automation and Test in Europe Conf. (DATE), pp.1625-1628, 2010.

[4.22] Xilinx ISE Tools, http://www.xilinx.com

[4.23] M. Lai, L. Gao, N. Xiao, and Z. Wang, “An accurate and efficient performance analysis approach based on queuing model for Network on Chip,” ACM/IEEE Int. Conference on Computer-Aided Design (ICCAD), 2009.

[4.24] E.B. van der Tol, E.G.T. Jaspers, “Mapping of MPEG-4 decoding on a flexible architecture platform,” SPIE, Media Processors, pp. 1-13, 2002.

CHAPTER 5. FAULT-TOLERANCE ORIENTED MULTI-LAYERED DESIGN METHODOLOGY FOR NETWORKS-ON-CHIP

This paper is submitted to IET The Journal of Engineering. The authors of the paper are Hamed Sajjadi Kia, Cristinel Ababei, and Sudarshan Srinivasan.

5.1. Abstract

We introduce a new fault-tolerance oriented design methodology for regular networks-on-chip (NoCs). The proposed methodology combines several design optimization techniques that cater to improving reliability at the CAD tool, NoC architecture, and network-routing levels. At the CAD tool level, we utilize a reliability aware mapping algorithm to assign application tasks to an NoC-based target architecture such that network reliability is improved. At the architecture and network-routing levels, we develop an NoC architecture that uses reconfigurable self-repairable links with spare wires and a distributed routing algorithm to dynamically detect permanent failures in unrepairable links and to recalculate routing paths using healthy links. It is the combination of these techniques that sets the proposed methodology apart and helps to provide a better tradeoff point between the improvement in fault tolerance and performance penalty due to the required redundancy and extra logic. A NoC prototype is implemented in Verilog-HDL and simulated to show the correct operation of the self-repairable links and of the adaptive routing.

5.2. Introduction

Network-on-Chip (NoC) has become a popular communication infrastructure in multiprocessor Systems-on-Chip (MPSoCs) [5.1, 5.2]. Borrowing a lot from the computer networks domain, on-chip networks consist of specialized routers connected via communication links in different topologies. These network topologies provide a communication medium for the processing elements (PE) connected to the network routers. Messages are organized as packets and transfered between PEs along paths established by the routing algorithm. The benefits of NoCs compared to the traditional buses include scalability, predictability, and higher bandwidth with support for concurrent communications.

Continuous shrinking of the feature size in deep submicron domains has resulted into increasingly adverse effects of process variations and aging mechanisms. These adverse effects translate into reliability issues (e.g., delay variations, transient and permanent faults, etc.) in both components of today's MPSoCs: processing cores and network-on-chip. Reliability has become a first class design concern aside from the traditional objectives that include performance, power dissipation, and area/cost [5.3]. To deal with the reliability challenge, previous research has mostly focused on the processing elements of MPSoCs. However, the NoC as the communication unit, represents a significant portion of MPSoCs and only recently, reliability of NoC started to be addressed; however, it is still largely an ongoing problem. A holistic design approach should address reliability of the whole system as a combination of both the computation (i.e., processing cores) and the communications (i.e., NoC) units.

Towards that goal, in this paper, we propose a new multi layered fault-tolerance oriented design methodology for NoCs as a hybrid solution composed of several reliability and fault tolerance design techniques applied at the CAD tool, NoC architecture, and network-routing levels. To the best of our knowledge this is one of the first fault-tolerance oriented design methods for NoCs that addresses hard failures across three different levels of abstraction. It is this layered approach, which combines the benefits of the proposed techniques at different levels that enables improved NoC resilience - it takes a very large number of hard faults to render the system inoperable. What makes the proposed method even more powerful is that it is very cost effective in terms of area and power overheads. In this paper, we combine improved versions of our previously studied techniques and propose a new fault tolerant adaptive routing algorithm to arrive to a comprehensive fault tolerant NoC solution. The proposed new adaptive routing algorithm utilizes a decentralized routing approach. The new routing strategy (compared to previous works [5.51, 5.34]) explores a larger solution space of possible routing scenarios. This improvement comes with an increase in the complexity of the routing algorithm. One of the main contributions of our new routing algorithm is the way we address this complexity. As an additional difference compared to previous works, the need to run the routing algorithm each time the system enters test mode is now eliminated. This helps to reduce the amount of time required for test mode and the power consumed during the process of updating the routing tables.

The rest of this paper is organized as follows: In Section 5.2, we discuss previous works related to reliability and fault tolerance in NoCs. In Section 5.3, we introduce the main idea behind our proposed methodology. In Section 5.4, we introduce briefly the reliability aware mapping for NoC, which represents one of the design techniques in our framework. Details about the reconfigurable link architecture are presented in Section 5.5. In Section 5.6 we present a new distributed adaptive routing algorithm. In Section 5.7, we evaluate the proposed ideas and present our experiments. Finally, some conclusions are drawn in Section 5.8.

5.3. Previous work related to fault tolerance and adaptive routing

5.3.1. Fault tolerance for NoCs

As mentioned earlier, previous work has focused mainly on processing elements as the computation units of MPSoCs. They addressed reliability by employing fault tolerant techniques based on error detection [5.4], failure prediction [5.5], and error masking [5.6]. While in the field of computer networks there has been a lot of work done on reliability, there have been only few recent attempts to estimate and indirectly optimize NoC reliability. At the physical layer, wires may be subject to delay variations [5.7], while routers may be impacted by single event upsets (SEUs) [5.8]. The data-link layer can provide the functional and procedural means to detect and possibly correct errors that may occur in the physical layer of NoC, by employing error correcting codes (ECCs) [5.9], data encoding [5.10], and redundancy based reconfiguration [5.11]. At the network layer, reliability of the routing algorithms can be enhanced by routing multiple copies of the same packet via multiple paths [5.12, 5.13] or by adaptive re-routing [5.14]. The reliability of custom switch architectures is analyzed in [5.15] while of various NoC topologies are analyzed in [5.16]. Several recent studies have investigated NoC architectures and techniques for fault tolerance [5.17, 5.18, 5.19, 5.20, 5.21, 5.22, 5.23, 5.24].

5.3.2. Adaptive routing for NoCs

Adaptive routing has been investigated as a primary mechanism to mitigate congestion and improve network performance. Many of these algorithms borrow ideas from routing in computer networks. For example, DyAD [5.25] is a combination of a static algorithm called oe-fix and an adaptive algorithm based on the turn-even model while [5.26] is based on deflection routing. Other previous adaptive routing algorithms include region-based routing (RBR) [5.27], dynamic XY (DyXY) [28] and enhanced versions of it [5.29, 5.30, 5.31], segment-routing (SR) [5.32], default backup path (DBP) [5.33], Vicis architecture [5.24, 5.34], reconfigurable fault-tolerant [5.35], and others [5.36, 5.51]. To deal with transient and permanent link failures, stochastic communication has been proposed [5.37]. However, it is not efficient in terms of power dissipation and average it latency. EDXY algorithm [5.30] increases the tolerance against single link failure compared to DyXY.

Generally, routing algorithms can be classified as truly distributed or centralized distributed. Centralized algorithms utilize a central manager (typically one of the existing on-chip processing elements or an on-chip processing unit) that collects information on the status of the links and routers. The manager computes region boundaries, new routing paths, various setup control signals, etc. to reflect changes in network traffic. The advantage of this approach is that the manager always has a global view of the network. However, additional global control signals are required for collecting information about the network status and to update routing tables. Examples include [5.38, 5.40]. Truly distributed algorithms, on the other hand, do not use a central manager [5.28]. Routing decisions are made by local routing controllers (located inside each router) based typically on congestion or stress information about the neighboring routers.

Routing can be implemented as source routing or distributed routing. In source routing, the source node computes the path and stores it in the packet header. Because the header itself is transmitted, this approach consumes network bandwidth. In distributed routing, each router computes the next link that will be used by the packets traveling across the network. The packet header contains only the destination address or ID. Distributed routing can be implemented in different ways. Hence, another classification of routing algorithms is based on whether routing tables are used or not. The main advantage of table-based routing algorithms [5.34, 5.41] is that they can support any network topology and any routing technique. However, because tables are implemented as memories, they do not scale well with network size in terms of area, performance and power. Even though this problem can be alleviated by table compression [5.42] or minimization [5.43], logic-based adaptive routing algorithms have been proposed recently [5.31, 5.38, 5.40, 5.44, 5.45].

Another categorization of adaptive routing algorithms is based on whether they are static or truly dynamic. Algorithms especially designed to achieve permanent fault tolerance rather than adaptiveness are static [5.44] in that they employ static fault models. They require shutdown or change of mode of operation for diagnosis purposes (e.g., enter testing mode to detect permanent link failures using BIST techniques). Also, routing paths recalculation and reconfiguration must be done before the faults can be tolerated. Examples include [5.31, 5.40, 5.44]. On the other hand, algorithms designed to achieve adaptiveness for congestion optimization or transient fault tolerance can be truly dynamic. In this case, the operation of the network need not be interrupted. Transient faults can be detected and corrected dynamically in real time by specialized data coding and error correcting techniques.

Finally, adaptive routing algorithms can use minimal paths or not. Minimal routing always uses shortest paths (assuming that at least one is available or is healthy) [5.38]. However, when source and destination nodes are located on the same row or column, one must change direction and route along non-shortest paths to tolerate link and router failures on the direct shortest path. This technique can also be employed when none of the shortest paths is healthy. However, the implementation of this approach is more complex. For a detailed review of adaptive routing algorithms please refer to [5.39].

Most previous works on adaptive routing report simulation results achieved with simulators developed in a high-level programming language (e.g., C++, Java, SystemC, etc.). Therefore, they typically do not report actual area overheads due to the lack of actual hardware implementation details.

5.4. Proposed fault tolerance oriented design methodology for NoCs

The majority of previous fault tolerance design techniques for NoCs are concentrated on individual levels of the system stack. A different approach is to design solutions that span multiple abstraction layers. This is the key idea of our fault tolerance oriented design methodology, which is a collaborative approach of reliability aware mapping (design tool level), reconfigurable links, and distributed adaptive routing (architecture and network layer). This is illustrated in the block diagram shown in Fig. 5.1.

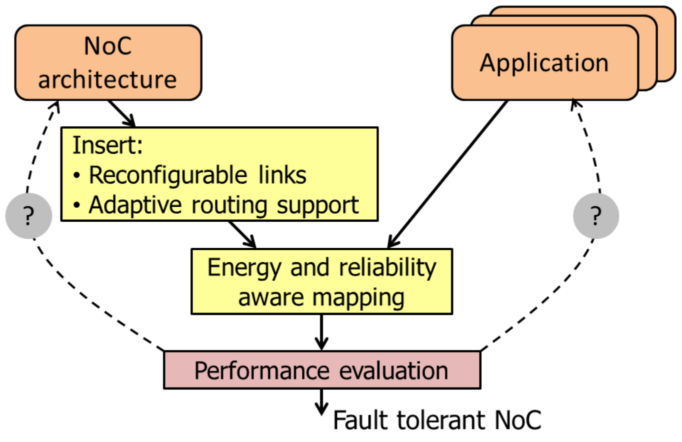


Figure 5.1. Modified Y Chart diagram of the proposed fault tolerance design methodology for NoCs.

At the CAD tool level, we employ a multi-objective mapping algorithm, which achieves mapping solutions with improved reliability. At the architecture level, we propose to utilize special self-healing links. Finally, we introduce a distributed adaptive routing algorithm to identify new routing paths and therefore to continue to maintain network operation when individual link repair is not possible due to the unavailability of spare resources. It is the combination of these three elements that makes our methodology powerful and distinguishes it from previous works. In the next sections we describe each of these three elements.

5.5. Energy and reliability aware mapping for regular NoCs

In this section we present the multi-objective mapping algorithm that we utilize in the proposed design methodology. Because we reported preliminary results of this algorithm in a conference paper [46], we present here only an overview discussion of it for the sake of completeness of the current presentation. Definitive details can be found in the aforementioned paper.

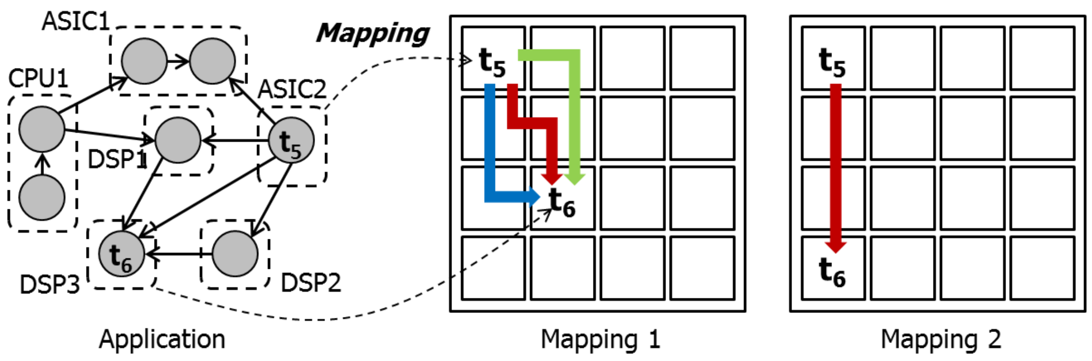


Figure 5.2. Illustration of the problem of mapping for regular mesh NoCs. Assigning application tasks t_5, t_6 as shown in the solution to the left, leads a higher minimal length path diversity which translates into better network reliability.

The problem of energy consumption and reliability aware application mapping is formulated for regular mesh NoC architectures. An application is given as an application characterization graph (APCG) G(C,A), which is a directed graph, where each vertex c_i ε C represents an IP core, and each directed arc a_{ij} ε A represents the communication between source core c_i and destination core c_j. Each a_{ij} can be tagged with application specific information (such as communication volume in bits $v(a_{ij})$ or communication rate) and specific design constraints (such as communication bandwidth, latency requirements, etc.). An APCG is derived from an application communication task graph (CTG) whose concurrent tasks have been already assigned and scheduled onto a list of selected IP cores. The mapping problem then is to decide how to topologically place the selected set of IP cores onto the PEs (or tiles) of the NoC array such that the metrics of interest are optimized (see Fig. 5.2).

The main idea of our reliability aware mapping algorithm relies on the following key observation: when network reliability is defined based on the concept of path diversity, then, a placement of source and destination nodes (of a given APCG) that forms a bounding box whose shape is as close as possible to a square should be preferred to for example a placement in a straight line (both tasks on the same row or column of the NoC). For the example in Fig. 5.2, the mapping of the tasks t_5 and t_6 as shown in the mapping solution to the left offers a better network reliability because the path diversity is better. In the second mapping solution shown in Fig. 5.2, if any link along the straight path is permanently damaged, then only non-minimal length routing paths could reconnect the two tasks. Note that this discussion is valid only under the assumption that the NoC architecture is designed to provide support for adaptive routing to directly benefit from such path diversity. We will discuss our adaptive routing algorithm designed for such purposes in Section 5.6.

We solve the energy consumption and reliability aware mapping problem with a novel branch-and-bound based algorithm. The reliability of a given mapping solution is estimated by an efficient Monte Carlo algorithm based on the so called destruction spectrum of the network. To model energy consumption, we utilize the so called bit energy metric model [5.56]:

$$E_{cost} = E_{Lbit} \sum_{a_{ij} \varepsilon A} v(a_{ij})\, dist(t_i, t_j) + E_{Rbit} \sum_{a_{ij}} v(a_{ij})\, [dist(t_i, t_j) + 1] \qquad (5.1)$$

where E_{Rbit} and E_{Lbit} represent the energy consumed when one bit of data is transported through one router and one physical link between two neighboring routers of the network. $v(a_{ij})$ is the communication volume between two cores of the application communication graph (APCG).

dist(t_i, t_j) represents the Manhattan distance between tiles t_i and t_j . This energy cost is directly utilized inside the cost function of the branch-and-bound mapping algorithms. Detailed presentations of both the algorithms and the network reliability estimation technique can be found in [5.46], where we have found that network reliability can be improved with minimal penalty in energy consumption compared to the case when the mapping problem would have energy consumption as the only objective.

5.6. Reconfigurable NoC links with spare wires

A popular technique to improve fault tolerance of communication links is to add spare wires and additional logic that can detect hardware faults and repair the link by replacing broken wires with spare ones. These techniques have been utilized in the past by several previous studies [5.11, 5.23, 5.47]. Although the described self-healing link has a simple structure, it is not scalable. To reduce the area overhead and thereby improve the scalability of such a fault tolerance technique, we propose to split a given link into two segments and assign spare wires on a subgroup basis to the wires from the main link. This is illustrated in the block diagram in Fig. 5.3.

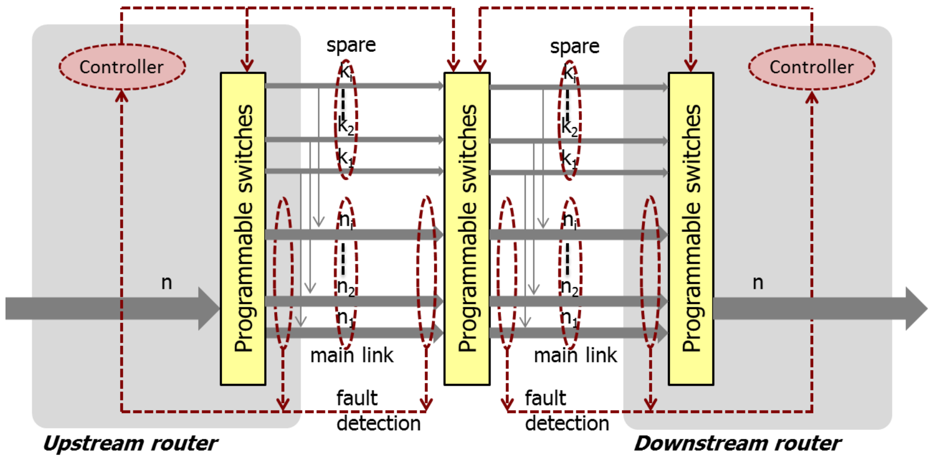


Figure 5.3. Block diagram of a reconfigurable link structure with spare wires and two segments.

We have shown in our preliminary report [5.48] of this technique that splitting the link into two segments offers the best compromise between how much the link reliability can be improved and the area penalty to support the additional spare wires, fault detection logic, controllers, and programmable switches (implemented with multiplexers and demultiplexers). In this case the area overhead for the link connecting the upstream and down-stream routers shown in Fig. 5.3 compared to a fault tolerant link implemented with only one segment is 3.71%, the power consumption is 23.19% higher, and the link delay (i.e., latency) increases 16.41%. However, link reliability is improved up to 20% for a link failure probability of 0.1%.

This tradeoff between improvement in link reliability and area penalty is better than any previously reported result. A detailed investigation of this tradeoff can be found in [5.48]. Table 5.1 shows a comparison between the area and power overhead of the proposed link and the available fault tolerant links [5.49] applied in the context of NoC.

Table 5.1. Area and power overhead of different fault tolerant links.

Link fault tolerant methods	Area overhead	Power overhead
Proposed link	12.8%	9.7%
Partially faulty link recovery mechanism	19.8%	6.15%
Simple flit quad splitting	16.9%	11.87%
Sectioned serialization method [5.49]	28.8%	15.39%

5.7. Proposed distributed adaptive routing

When the technique presented in the previous section runs out of spare wires for a given link, then in order to maintain the overall NoC operational we propose to use an adaptive routing algorithm. The proposed adaptive algorithm is a distributed routing approach to improve its scalability and therefore serves two purposes: 1) We utilize it as a primary technique to improve fault tolerance. It is triggered only when at least a link cannot self-repair through the fault tolerance technique described in the previous section. 2) It provides architectural support for cost-effective adaptive routing, which is necessary to enable the benefits of the network reliability aware mapping solutions achieved with the mapping algorithm discussed in Section 5.4. Only with an NoC architecture that supports adaptive routing we can take advantage of the improved path diversity made available to us by the reliability aware mapping solutions.

Again, fault tolerance via adaptive routing comes at the expense of an increase in area and power consumption required by the additional hardware. To minimize this penalty, we design our adaptive algorithm as a distributed (i.e., decentralized) routing approach. We partition the NoC architecture into several regions or partitions. Each region has an associated local control unit (LCU), which is responsible with the routing activities of all packets that enter any router contained within the region. The NoC example in Fig. 5.4 is partitioned into 9 equally sized regions. However, these regions do not need to be of equal area. The proposed adaptive routing can be deployed to NoCs partitioned asymmetrically. We will present our discussion however in the situation when all partitions are equal for convenience and ease of explanations.

Each LCU is responsible with the routing of data to routers in its designated region and to the first-order neighboring routers adjacent to its designated region. For example in Fig. 5.4, LCU_5 is responsible for routing packets that enter the region formed by the routers {14, 15, 20, 21}. The origin of the packets can be either the PEs connected to these routers or the routers in the neighboring adjacent regions.

Once a packet is injected into a router at one of its input ports, its corresponding output port is determined by simply reading from the routing table also located inside the router. When links suffer from permanent faults that cannot be addressed via link reconfiguration anymore, routing paths are recalculated and routing tables are updated with new routing paths information. The updates are done by the routing controllers located inside each router. The routing paths recalculation is performed by LCUs. This is done by switching the whole system periodically to a testing mode state or on demand by the detection of a new hardware fault. Initially, when the network is hardware fault free, all routing tables implement a traditional static XY routing algorithm. As the system ages and hardware faults occur, the reconfigurable links address a first set of the faults. The size of this set depends on the number of available spare wires in each link. Once the system runs out of available spare wires, the adaptive routing algorithm is triggered and new routing paths are recalculated. Note, that this new routing paths may still have links, which if affected by hardware faults may still be repairable via link reconfiguration. When such reconfiguration cannot be done because the systems again runs out of available spare wires, the adaptive routing algorithm is triggered again and so on.

5.7.1. Description of the local control unit (LCU)

Once the system enters the test mode, the status of each link is determined by an error detection mechanism integrated with each link and the results are reported to LCUs. Each LCU uses the acquired data about the status of the 16 links in its designated region to determine the output port for the packets that enter the routers in its region. In our design, a given LCU does not need to know the status of the links that enter its designated region from neighboring regions; hence, the total number of links an LCU has to monitor is 16.

To easier describe how an LCU and the adaptive routing algorithm work, let us focus our discussion on router$_{14}$, which is part of the partition controlled by LCU$_5$ of the NoC illustrated in Fig. 5.4. Fig. 5.5 describes the routing algorithm to determine the output port for the packets injected to router$_{14}$ from either PE$_{14}$ or any of the neighboring routers based on their destinations and the status of the links in the region. Note that similar algorithms are run at the level of routers 15, 20, and 21. Also note that even though there are 16 links in each region, we only need the information about the status of 9 of them to determine the output port for the packets injected to each router in each region. These 9 links for router$_{14}$ are shown (numbered from 0 to 8) in Fig. 5.6. The number of links used is dependent on the shape and size of the region. As we will explain shortly, the number of these links has a direct relation with the hardware redundancy required to implement the adaptive routing algorithm. Therefore, we want to use the minimum number of links in our algorithm. In the case of router$_{14}$, the routing algorithm requires knowledge only about the status of the nine links depicted in Fig. 5.6. In other words, the algorithm would not be able to execute if the information about the status of any of these links is missing. On the other hand, the information about the status of other links is redundant. For example, the links that enter router$_{14}$ from south and east in this figure can bring data to it but because we are interested in routing data out of router$_{14}$ their status is not considered in the routing algorithm of router$_{14}$.

While easy to understand (as it will be described shortly), the algorithm from Fig. 5.5 is rather complex from the point of view of required area and power overheads if implemented in hardware. Therefore, to address this issue, we propose to run the algorithm offline and save the results in a look up table (implemented as a memory array). This look up table will then be later utilized on-line directly by LCU$_5$ to update (during test mode) the routing tables located inside the routers that it monitors.

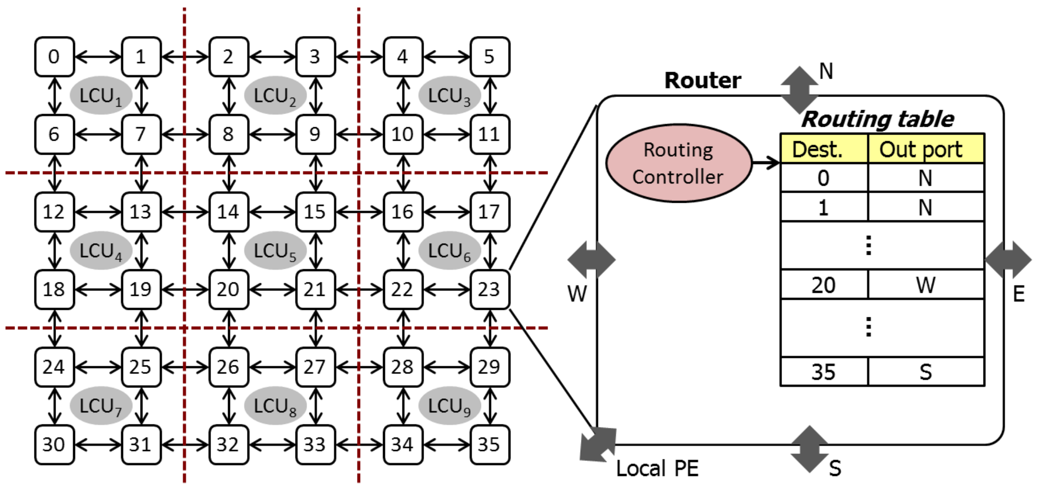


Figure 5.4. Illustration of partitioning of a given NoC architecture into regions. Each region is managed by a local control unit (LCU).

We developed the proposed routing algorithm based on a few assumptions. First, once the number of hard failures in a given link becomes larger than the number of available spare wires to repair it, a failure signal will be set to indicate a link failure. We refer to this signal as $error_i$. For example, an assertion of the error1 signal means that link 1 is broken. These error signals indicate the status of all links in the network and are used by the routing algorithm in deciding the output ports in a given router where packets can be forwarded. They are also used as the address bits to indicate the location inside the look up table where to save/store the results of running offline the algorithm from Fig. 5.5. When the number of failures and their location is such that LCUs cannot route packets toward their destinations, a special system failure signal is asserted to signal that the system cannot be utilized anymore.

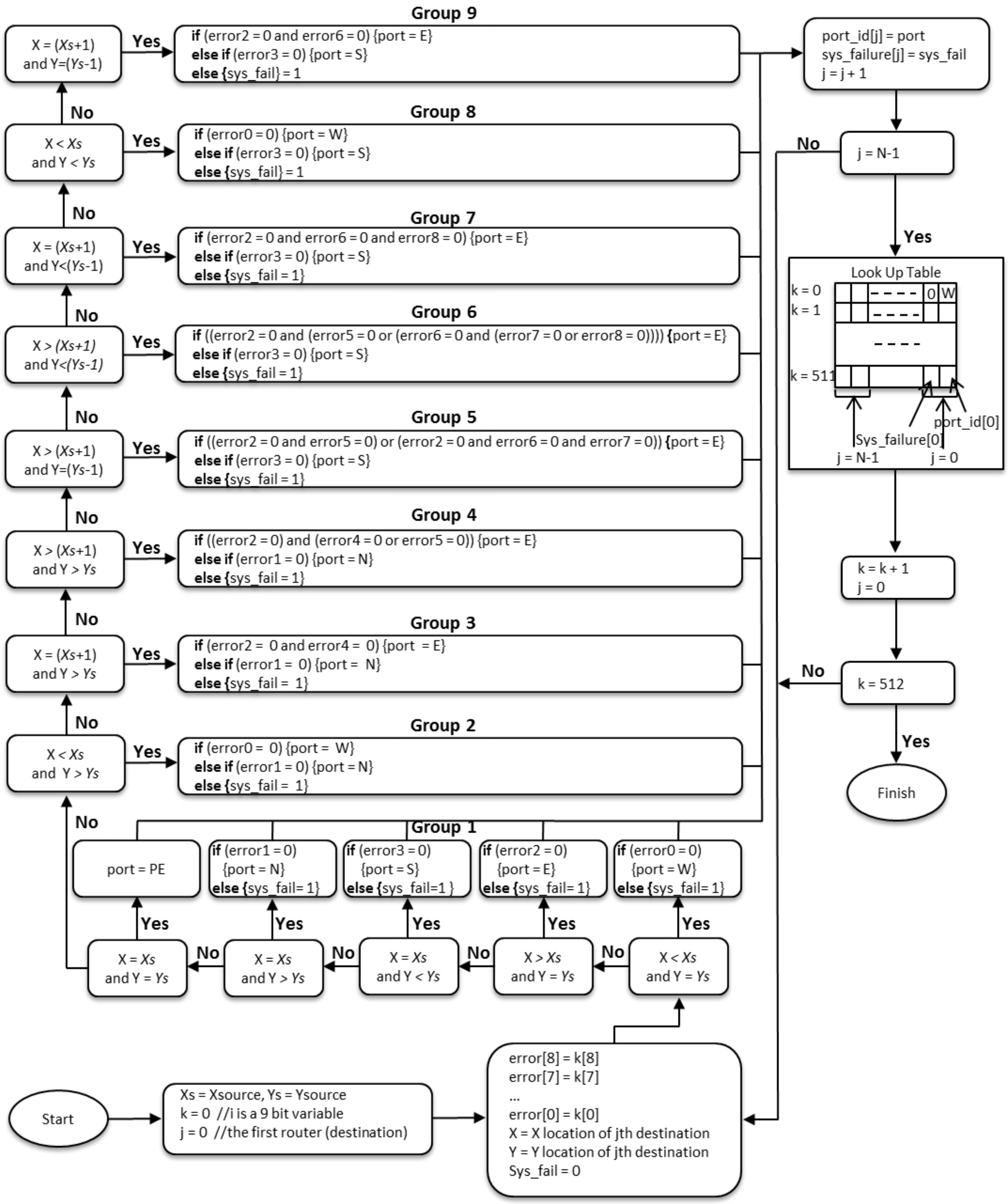

Group 9
X = (Xs+1) and Y=(Ys-1)
Yes
if (error2 = 0 and error6 = 0) {port = E}
else if (error3 = 0) {port = S}
else {sys_fail} = 1
port_id[j] = port
sys_failure[j] = sys_fail
j = j + 1
No
Group 8
X < Xs and Y < Ys
if (error0 = 0) {port = W}
else if (error3 = 0) {port = S}
else {sys_fail} = 1
j = N-1
Group 7
X = (Xs+1) and Y<(Ys-1)
if (error2 = 0 and error6 = 0 and error8 = 0) {port = E}
else if (error3 = 0) {port = S}
else {sys_fail = 1}
Look Up Table
k = 0
k = 1
k = 511
0 W
Group 6
X > (Xs+1) and Y<(Ys-1)
if ((error2 = 0 and (error5 = 0 or (error6 = 0 and (error7 = 0 or error8 = 0)))) {port = E}
else if (error3 = 0) {port = S}
else {sys_fail = 1}
port_id[0]
Sys_failure[0]
j = N-1
j = 0
Group 5
X > (Xs+1) and Y=(Ys-1)
if ((error2 = 0 and error5 = 0) or (error2 = 0 and error6 = 0 and error7 = 0)) {port = E}
else if (error3 = 0) {port = S}
else {sys_fail = 1}
Group 4
X > (Xs+1) and Y > Ys
if ((error2 = 0) and (error4 = 0 or error5 = 0)) {port = E}
else if (error1 = 0) {port = N}
else {sys_fail = 1}
k = k + 1
j = 0
Group 3
X = (Xs+1) and Y > Ys
if (error2 = 0 and error4 = 0) {port = E}
else if (error1 = 0) {port = N}
else {sys_fail = 1}
k = 512
Group 2
X < Xs and Y > Ys
if (error0 = 0) {port = W}
else if (error1 = 0) {port = N}
else {sys_fail = 1}
Finish
Group 1
port = PE
if (error1 = 0) {port = N} else {sys_fail= 1}
if (error3 = 0) {port = S} else {sys_fail=1 }
if (error2 = 0) {port = E} else {sys_fail= 1}
if (error0 = 0) {port = W} else {sys_fail= 1}
X = Xs and Y = Ys
X = Xs and Y > Ys
X = Xs and Y < Ys
X > Xs and Y = Ys
X < Xs and Y = Ys
Start
Xs = Xsource, Ys = Ysource
k = 0 //i is a 9 bit variable
j = 0 //the first router (destination)
error[8] = k[8]
error[7] = k[7]
...
error[0] = k[0]
X = X location of jth destination
Y = Y location of jth destination
Sys_fail = 0

Figure 5.5. Block diagram of the proposed adaptive routing algorithm for router$_{14}$.

Second, each routing path should satisfy the following condition: packets should always be routed toward their destination in either X or Y direction. Packets are routed first in the X direction, unless that is not possible due to failed links or the packets are at a router that has already the same Y coordinate as the destination - cases in which packets are attempted to be routed in the Y direction. If routing in the Y direction is not possible, again due to possibly failed links, the LCU sets the system failure signal.

Third, based on the (X, Y) location of the destination, all possible destinations (of the packets injected in any given router) are divided into 9 groups (see Fig. 5.5). For example, group 1 includes all routers that are in the same row or column as router14 (have the same X or Y coordinates). There is only one routing path to these routers. Group 2 includes the routers whose X coordinate is smaller than the X coordinate of router$_{14}$ and their Y coordinate is larger than the Y coordinate of router$_{14}$ (routers {0, 1, 6, 7} in Fig. 5.4).

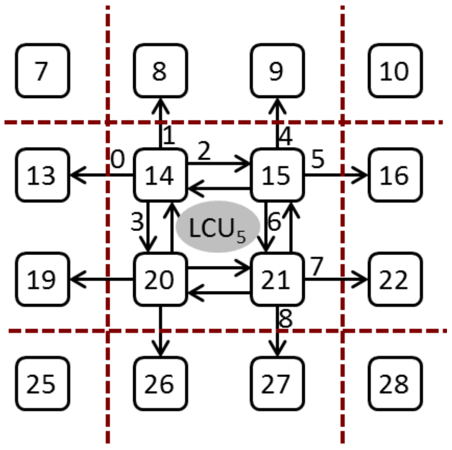


Figure 5.6. The 9 links used for routing data entered router$_{14}$.

The algorithm from Fig. 5.5 has primarily two for loops. It starts with the first destination j = 0 (router$_0$) under the assumption that all 9 links (Fig. 5.6) are functional i = 0, and repeats the process for all destinations (j = 0 … (N - 1)). Once the loop is completed, the results are saved in the first row of the look up table in the following order: first 3 bits of the first row of the look up table hold the output port id for the packets that enter router$_{14}$ and their destination is router0 (under the assumption that all 9 links are functional). The fourth bit indicates if the system can identify a routing path for these packets or not. In this case, because all of the 9 links are functional it is set to 0. The routing information for the remaining destinations is saved in the same order in the first row of the look up table - as shown in Fig. 5.5. Next, the above process is repeated under the assumption that only link 0 is broken (i = 1 or i[0] = 1) and save the routing information in the second row of the look up table. This process is repeated until we generate and save the routing information for all possible destinations under all link failure situations.

The operation of the routing algorithm is further described in the following scenario examples. Consider a situation where a packet arrives to router_{14} and its destination is router_{17}. router_{17} is in the first destination group. Based on the assumption that packets should always move toward their destination either in X or Y direction, we see that there is only one path to destination. The algorithm checks the status of link 2 (Fig. 5.6) if it is healthy, the packet will be routed to the east output port (to router_{15}). At this stage, the algorithm will not check the status of link 5. Once the packet enters router_{15}, the algorithm for router15 will check the status of link 5 and if it is broken it will set the system failure. This example illustrates the reliability issue discussed earlier in Fig. 5.2. Because source and destination have the same X or Y coordinates, there is only one routing path between them and therefore the communication pair has lower reliability. As another example, let us consider the s - d pair, router_{14} and router_{5} in Fig. 5.4. router_{5} is located in group 4 ($\text{Xrouter}_{5} > (X_{\text{router14}} + 1)$ and $Y_{\text{router5}} > Y_{\text{router14}}$ (Fig. 5.5)). Because the destination is in region 3, the LCU_{5} should send the packets to one of the routers {8, 9, 16} in regions 2 and 6. The routing algorithm first tries the X direction. If link 2 and one of the links 4 or 5 (Fig. 5.6) are healthy, the packets will be routed to the east output port (this is because once the packets enter router_{15} there will be at least one path open to send the packets toward region

3), otherwise the algorithm will check the status of link 1 - if it is healthy, packets will be routed to the north output port. If link 1 is also broken, the system failure signal will be asserted.

As mentioned earlier, the routing algorithm is run offline for all destinations under all possible link failure scenarios and the results are recorded in a look up table (as is illustrated in Fig. 5.5). Note that the algorithm is run once offline for all possible failure scenarios and the results are stored in a look up table. The algorithm will not be invoked during the operation of the NoC. As already described, the error bits are utilized to address different locations in the look up table. This technique effectively allows the LCUs to get pre-computed routing information easily under different link failure situations. Fig. 5.7 illustrates the process of updating the routing tables by the routing controllers integrated inside the routers using the look up tables in LCUs that contain the results of running the adaptive routing algorithm offline.

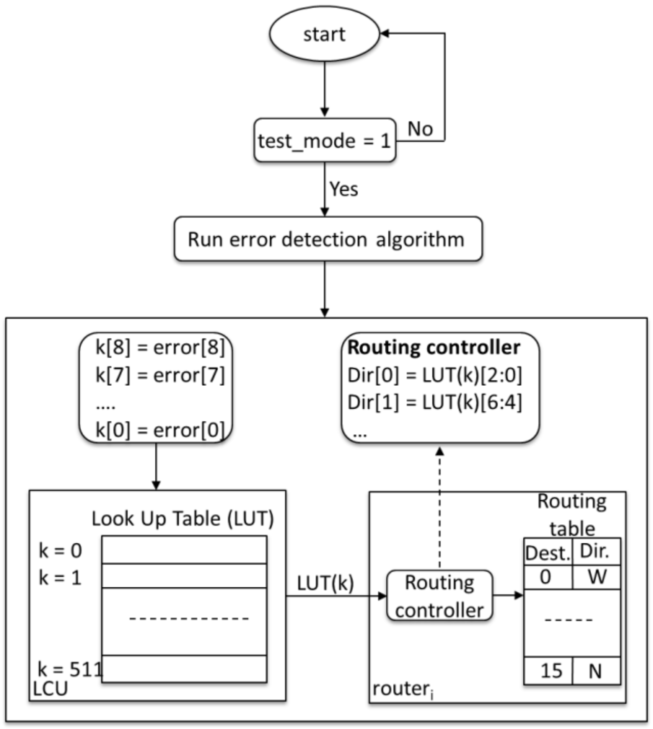


Figure 5.7. Illustration of the process of updating the routing tables.

5.8. Experimental results

5.8.1. Simulations of the mapping algorithm

In the first set of experiments, we follow the block diagram from Fig. 5.1. We apply the proposed mapping algorithm to several testcases whose characteristics are shown in Table 5.2. The implementation, as a C++ computer program, of our mapping algorithm from Section 5.4 is publicly available at [5.57].

The results of our simulations are shown in Table 5.3 where we investigate how much the proposed mapping algorithm can improve network reliability. Refer to our conference paper [5.46] for details on the models utilized for the energy and reliability estimations. It can be observed that the overall network reliability of the mapped application is improved when reliability is also included into the cost function of the mapping algorithm. However, the improvement in reliability is at the expense of increasing the energy consumption. Because, here we refer to the energy consumption of the NoC (which accounts for only up to 28% of the overall energy consumption of a multiprocessor SoC [5.52]), the energy consumption overhead is still within reasonable limits. The tradeoff between reliability and energy consumption can be controlled via a user-set parameter, which weights the importance of the reliability cost component in the multi-objective cost function of the mapping algorithm.

In our simulation experiments, we found that network reliability decreases as the connectivity of the APCG increases (for example testcases ami25 and ami49 have almost all cores communicating with each other). This can be explained in part by the fact that each link is utilized by more routing paths. Thus, a link failure has a bigger impact on the state of the network. Another factor which affects the overall network reliability is the probability of link failure q. The results from Tables 3 were achieved for a q = 0.01. If this is increased, then the improvement in reliability achieved with the reliability aware mapping algorithm becomes larger.

Table 5.2. Testcases and their characteristics used for mapping simulations.

Testcase	Num. of cores	APCG connectivity	Min/max comm. vol.
mpeg4	9	low	1/942
telecom	16	medium	11/71
ami25	25	high	1/4
ami49	49	high	1/14

Even though it may seem that reliability improvement of only a few percentages is not much, this improvement cannot be directly compared as a percentage against the percentage of energy or performance overhead. Instead, the significance of such an improvement of reliability lies in its economic potential (what is the cost savings when for example 3% out of the total population of systems do not fail - i.e., when yield is improved by 3%), consequences (which can be severe in critical applications), and user satisfaction.

5.8.2. Cost estimations

Next, we estimate the extra hardware cost to implement the reconfigurable links as well as the proposed adaptive algorithm described in Fig. 5.5.

We developed a complete regular mesh NoC with reconfigurable links that utilize 4 spare wires in addition to the 16 wires that form the regular links. We compare this custom NoC against a traditional NoC implementation that has simple regular links and no support for adaptive routing. Both these NoC architectures are coded in Verilog-HDL, synthesized and simulated with Xilinx ISE WebPack [5.60]. We find that the proposed custom NoC requires 17.94% more area compared to the traditional NoC, it consumes on average 16.81% more power, and it can operate at a maximum operation frequency that is 20.84% smaller. It is worth mentioning that the cost of 17.94% in area increase is actually not too much given that the custom NoC implementation has all its links as reconfigurable links with spare wires.

Table 5.4 depicts a comparison of the area and power overheads between the proposed fault tolerant algorithm and some of the previously proposed fault tolerant routing algorithms. As an additional experiment, we have also implemented the proposed routing algorithm with regular rather than reconfigurable links. In this case, the maximum achievable operation frequency is with only 3.86% (rather than 20.84%) lower compared to the traditional XY routing algorithm.

Table 5.3. Simulations results achieved with the energy and reliability aware mapping algorithm.

Testcase	CPU runtime (s)	Objective: energy		Objective: energy & reliability	
		Energy (J)	Reliability (%)	Energy (J)	Reliability (%)
mpeg4	0.23	5.73	97.96	5.74	98.953
telecom	2.1	13.969	95.163	14.794	98.064
ami25	7.25	6.907	96.91	7.185	97.932
ami49	124.1	15.402	91.378	15.942	94.111

Table 5.4. Area and power overhead of different fault tolerant routing algorithms for NoC.

Fault tolerant routing algorithm	Area overhead (compared to XY algorithm)	Power overhead (compared to XY algorithm)
Proposed adaptive routing algorithm with reconfigurable links	17.94%	16.81%
Proposed adaptive routing algorithm without reconfigurable links	5.1%	7.1%
DyRS-NM [5.55]	5.64%	NA
DSPIN [5.35]	8%	NA
FT XY3 [5.54]	6.1%	5.2%
Negative first [5.53]	33.2%	NA

5.9. Conclusion

We proposed a hybrid fault tolerance oriented design methodology for regular networks-on-chip. The key contribution is the combination of three design optimization techniques to improve reliability at three different levels: 1) CAD tool - via network reliability oriented mapping, 2) NoC architecture - via redundancy based self-repairable communication links, and 3) network-routing - via cost-effective distributed adaptive routing. It is the combination of these techniques that distinguishes the proposed methodology from previous work and helps to provide improved fault tolerance and graceful performance degradation in the face of increasingly adverse hardware faults due to aging mechanisms.

5.10. References

[5.1] W.J. Dally and B.P. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann, 2004.

[5.2] G. De Micheli and L. Benini, Networks on Chip, Morgan Kaufmann, 2006.

[5.3] J.D. Owens, W.J. Dally, R. Ho, D.N. Jayasimha, S.W. Keckler, and L.S. Peh, Research challenges for on-chip interconnection networks, IEEE Micro, vol. 27, no. 5, pp. 96-108, 2007.

[5.4] S. Feng, S. Gupta, A. Ansari, and S. Mahlke, Shoestring: probabilistic soft error reliability on the cheap, International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2010.

[5.5] Y. Li, Y.M. Kim, E. Mintarno, D.S. Gardner, and S. Mitra, Overcoming early-life failure and aging for robust systems, IEEE Design & Test of Computers, vol. 26, no. 6, pp. 28-39, 2009.

[5.6] M. Choudhury, V. Chandra, K. Mohanram, and R. Aitken, TIMBER: time borrowing and error relaying for online timing error resilience, Proceedings of Design Automation and Test in Europe Conference (DATE), 2010.

[5.7] C. Hernandez, F. Silla, and J. Duato, A methodology for the characterization of process variation in NoC links, Proceedings of Design Automation and Test in Europe Conference (DATE), 2010.

[5.8] A. Ejlali, B.M. Al-Hashimi, P. Rosinger, and S.G. Miremadi, Joint consideration of fault-tolerance, energy-efficiency and performance in on-chip network, Proceeding of Design Automation and Test in Europe Conference (DATE), 2007.

[5.9] S.R. Sridhara and N.R. Shanbhag, Coding for system-on-chip networks: a united framework, IEEE Trans. on Very Large Scale Integration Systems (TVLSI), vol. 13, no. 6, pp. 655-667, 2005.

[5.10] D. Bertozzi, L. Benini, and G. De Micheli, Error control schemes for on-chip communication links: the energy-reliability tradeoff, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD), vol. 24, no. 6, pp. 818-831, 2005.

[5.11] T. Lehtonen, D. Wolpert, P. Liljeberg, J. Plosila, and P. Ampadu, Self-adaptive system for addressing permanent errors in on-chip interconnects, IEEE Trans. on Very Large Scale Integration Systems (TVLSI), vol. 18, no. 4, pp. 527-540, 2010.

[5.12] S. Murali, D. Atienza, L. Benini, G. De Micheli, A multi-path routing strategy with guaranteed in order packet delivery and fault-tolerance for networks on chip, Design Automation and Test in Europe Conference (DATE), 2006.

[5.13] A. Patooghy and S.G. Miremadi, Complement routing: a methodology to design reliable routing algorithm for network on chips, Microprocessors and Microsystems, vol. 34, no. 6, pp. 163-173, 2010.

[5.14] A.D. Choudhury, G. Palermo, C. Silvano, and V. Zaccaria, Yield enhancement by robust application specific mapping on Network-on-Chips, International Workshop on Network on Chip Architectures (NocArc), 2009.

[5.15] A. Dalirsani, M. Hosseinabady, and Z. Navabi, An analytical model for reliability evaluation of NoC architectures, International On-Line Testing Symposium (IOLTS), 2007.

[5.16] T. Lehtonen, P. Liljeberg, and J. Plosila, Fault tolerance analysis of NoC architectures, International Symposium on Circuits and Systems (ISCAS), 2007.

[5.17] J. Kim, D. Park, C. Nicopoulos, N. Vijaykrishnan, and C.R. Das, Design and analysis of an NoC architecture from performance, reliability and energy perspective, Symposium on Architecture for Networking and Communications Systems (ANCS), 2005.

[5.18] D. Park, C.A. Nicopoulos, J. Kim, N. Vijaykrishnan, and C.R. Das, Exploring fault-tolerant Network-on-Chip architectures, International Conference on Dependable Systems and Networks (DSN), 2006.

[5.19] F. Worm, P. Thiran, G. de Micheli, and P. Ienne, Self-calibrating Networks-On-Chip, International Symposium on Circuits and Systems (ISCAS), 2005, pp. 2361-2364.

[5.20] S. Shamshiri and K.T. Cheng, Yield and cost analysis of a reliable NoC, VLSI Test Symposium, 2009.

[5.21] H. Elmiligi, A.A. Morgan, M.W. El-Kharashi, and F. Gebali, A reliability-aware design methodology for Networks-on-Chip applications, International Conference on Design and Technology of Integrated Systems in Nanoscale Era, 2009.

[5.22] A. Patooghy, S.G. Miremadi, and M. Fazeli, A low-overhead and reliable switch architecture for Network-on-Chips, Integration, the VLSI Journal, vol. 43, no. 3, pp. 268-278, 2010.

[5.23] Q. Yu and P. Ampadu, Transient and permanent error co-management method for reliable Networks-on-Chip, International Symposium on Networks-on-Chip (NOCS), 2010.

[5.24] A. DeOrio, D. Fick, V. Bertacco, D. Sylvester, D. Blaauw, J. Hu, and G. Chen, A Reliable routing architecture and algorithm for NoCs, IEEE Trans. on Computer-Aided Design (TCAD), vol. 31, no. 5, pp. 726-739, 2012.

[5.25] J. Hu and R. Marculescu, DyAD - smart routing for Networks-on-Chip, Design Automation Conference (DAC), 2004.

[5.26] A. Kohler and M. Radetzki, Fault-tolerant architecture and detection routing for degradable NoC switches, International Symposium on Networks-on-Chip (NOCS), 2009.

[5.27] J. Flich, A. Mejia, P. Lopez, and J. Duato, Region Based Routing: An efficient routing mechanism to tackle unreliable hardware in network on chip, International Symposium on Networks-on-Chip (NOCS), 2007.

[5.28] M. Li, Q.-A. Zeng, and W.B. Jone, DyXY - a proximity congestion-aware deadlock free dynamic routing method for Network-on-Chip, Design Automation Conference (DAC), 2006.

[5.29] P. Gratz, B. Grot, and S.W. Keckler, Regional congestion awareness of load balance in Network-on-Chips, International Symposium on High Performance Computer Architecture, 2008.

[5.30] P. Lot-Kamran, A.M. Rahmani, M. Daneshtalab, A. Afzali-Kusha, and Z. Navabi, EDXY- A low cost congestion-aware routing algorithm for network-on-chips, Journal of Systems Architecture, vol. 56, no. 7, pp. 256-264, 2010.

[5.31] M. Valinataj, S. Mohammadi, J. Plosila, P. Liljeberg, and H. Tenhunen, A reconfigurable and adaptive routing method for fault-tolerant mesh-based networks-on-chip, International Journal of Electronics and Communications, vol. 65, no. 7, pp. 630-640, 2011.

[5.32] A. Mejia, J. Flich, J. Duato, S.A. Reinemo, and T. Skeie, segment-based routing: An efficient fault-tolerant routing algorithm for meshes and Tori, International Parallel & Distributed Processing Symposium (IPDPS), 2006.

[5.33] M. Koibuchi, H. Matsutani, H. Amano, and T. Pinkston, A lightweight fault-tolerant mechanism for network-on-chip, International Symposium on Networks-on-Chip (NOCS), 2008.

[5.34] D. Fick, A. DeOrio, G. Chen, V. Bertacco, D. Sylvester, and D. Blaauw, A highly resilient routing algorithm for fault-tolerant NoCs, Design Automation and Test in Europe Conference (DATE), 2009.

[5.35] Z. Zhang, A. Greiner, and S. Taktak, A reconfigurable routing algorithm for a fault-tolerant 2D-mesh network-on-chip, Design Automation Conference (DAC), 2008.

[5.36] A. Hosseini, T. Ragheb , and Y. Massoud, A fault-aware dynamic routing algorithm for on-chip networks, International Symposium on Circuits and Systems (ISCAS), 2008.

[5.37] T. Dumitras and R. Marculescu, On-chip stochastic communication, Design Automation and Test in Europe (DATE), 2003.

[5.38] J. Flich, S. Rodrigo, and J. Duato, An Efficient Implementation of Distributed Routing Algorithms for NoCs, International Symposium on Networks-on-Chip (NOCS), 2008.

[5.39] J. Flich, T. Skeie, A. Mejia, O. Lysne, P. Lpez, A. Robles, J. Duato, M. Koibuchi, T. Rokicki, J.C. Sancho, A Survey and Evaluation of Topology agnostic deterministic routing algorithms, IEEE Transaction on Parallel and Distributed Systems, vol. 23, no. 3, pp. 405–425, 2012.

[5.40] T. Skeie, F.O.S. Jacobsen, S. Rodrigo, J. Flich, D.Bertozzi, and S. Medardoni, Flexible DOR routing for virtualization of multicore chips, International Symposium on System-on-Chip, 2009.

[5.41] T. Schonwald, J. Zimmermann, O. Bringmann, W. Rosenstiel, Fully adaptive fault-tolerant routing algorithm for Network-on-Chip architectures, Euromicro Conference on Digital System Design Architectures, Method sand Tools (DSD), 2007.

[5.42] M. Palesi, S. Kumar, and R. Holsmark, A method for router table compression for application specific routing in mesh topology NoC architectures, embedded computer systems: architectures, modeling, and simulation, vol. 4017, pp. 373-384, 2006.

[5.43] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, Routing Table Minimization for Irregular Mesh NoCs, Design Automation and Test in Europe (DATE), 2007.

[5.44] 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, International Symposium on Networks-on-Chip (NOCS), 2010.

[5.45] F.O.S. Jacobsen, S. Rodrigo, and T. Skeie, iFDOR: dynamic rerouting on-chip, International Workshop on Interconnection Network Architecture: On-Chip, Multi-Chip (INA-OCMC), 2011.

[5.46] C. Ababei, H.S. Kia, O.P. Yadav, and J. Hu, Energy and reliability oriented mapping for regular Networks-on-Chip, International Symposium on Networks-on-Chip (NOCS), 2011.

[5.47] T. Lehtonen, P. Liljeberg, and J. Plosila, Online reconfigurable self-timed links for fault tolerant NoC, VLSI Design, 2007.

[5.48] H.S. Kia and C. Ababei, Improving fault tolerance of Network-on-Chip links via minimal redundancy and reconfiguration, International Conference on Reconfigurable Computing and FPGAs, 2011.

[5.49] C. Chen, Y. Lu, and S.D. Cotofana, A novel it serialization strategy to utilize partially faulty links in networks-on-chip, Proceeding of International Symposium on Networks-on-Chip (NOCS), 2012.

[5.50] T. Lehtonen, D. Wolpert, P. Liljeberg, J. Plosila, and P. Ampadu, Self-adaptive system for addressing permanent errors in on-chip interconnects, IEEE Trans. on VLSI Systems, vol. 18, no. 4, pp. 527-540, 2010.

[5.51] H.S. Kia and C. Ababei, A new fault-tolerant and congestion aware adaptive routing algorithm for regular Networks-on-Chi, Proceeding of Congress on Evolutionary Computation (CEC), 2011.

[5.52] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, A 5-GHz mesh interconnect for a teraflops processor, IEEE Micro, vol. 27, no. 5, pp. 51-61, 2007.

[5.53] H. Zhu, P.P. Pande, and C. Grecu, Performance evaluation of adaptive routing algorithms for achieving fault tolerance in NoC fabrics, International Conference on Application Specific Systems, Architectures and Processors, pp. 42-47, 2007.

[5.54] M. Valinataj, S. Mohammadi, and S. Safari, fault-aware and reconfigurable routing algorithms for networks-on-chip, IETE Journal of Research, vol. 57, no. 3, pp. 215-224, 2011.

[5.55] F. Ge, N. Wu, and Y. Wan, A Network Monitor based Dynamic Routing Scheme for Network on Chip, Proceeding of Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics, pp. 133-136, 2009.

[5.56] T.T. Ye, L. Benini, and G. De Micheli, Analysis of power consumption on switch fabrics in network routers," ACM/IEEE Design Automation Conference (DAC), 2002.

[5.57] ReliableNoC tool, http://venus.ece.ndsu.nodak.edu/cris/software.html

[5.58] E.B. van der Tol and E.G.T. Jaspers, Mapping of MPEG-4 decoding on a flexible architecture platform, SPIE Media Processors, 2002.

[5.59] MCNC Benchmarks, http://vlsicad.eecs.umich.edu/BK/MCNCbench

[5.60] ISE Design Suite, Xilinx, http://www.xilinx.com/tools/webpack.htm

CHAPTER 6. CONCLUSION

Recent advances in development and fabrication of integrated circuit technology has made integration of tens of processing elements (PE) in one chip possible. While this trend is improving their performance, it is facing some challenges that may limit these improvements. One of the main facing challenges is the increase in the probability of transistor failure and fault occurrence [6.1], [6.2]. This is due to the increase in the adverse effects of processing faults which make chips susceptible to manufacturing faults (that have become unavoidable in current submicron CMOS technology [6.3], [6.4]) and also to the aging mechanisms that reduce the life time reliability of integrated circuits [6.5], [6.6]. Therefore reliability alongside area and power consumption has become an important design objective [6.7]. Although fault tolerance (the ability of the system to continue its operation despite the failure of some of its components [6.8]) has been one of the popular research areas [6.9], [6.10], the increase in failure probability of transistors in submicron CMOS technology has highlighted its importance more than ever. Fault tolerance can improve reliability by employing design techniques that can compensate for the failure of system components [6.11], [6.12]. Different fault tolerant design techniques and architectures have been developed over the time to work around the facing reliability challenge. One reason for this is that systems are affected with different failure types which require different techniques to deal with them [6.13], [6.14]. The purpose of this research was to maintain the downscaling benefits of integrated circuits by addressing the challenges we are facing in design of sub-micron integrated circuits. To achieve this goal we focused on developing design techniques to proactively improve the lifetime reliability of system on chips (SoC).

One of the main difficulties in reliability aware system design is the estimation of reliability. This is due to the fact that reliability is affected by numerous factors including aging mechanisms (e.g., time-dependent dielectric breakdown (TDDB), negative bias temperature instability (NBTI), electromigration (EM), thermal cycling (TC), and stress migration (SM)), process variations, dynamic power and thermal management, workload, and system architecture and configuration. Typically, resilience techniques to harden a system against different failure mechanisms require some form of redundancy. Such redundancy comes with area, power, and design time overheads. Therefore it is not practical to develop systems with resilience technique to all types of failure mechanisms. However if we identify reliability critical subblocks and transistors, we can concentrate our design efforts on the critical subblocks and save area and power resources. To achieve this goal we developed a circuit-level vulnerability and reliability evaluation methodology that is capable of identifying the vulnerable subblocks of the system. In the core of the algorithm we employ a Monte Carlo algorithm, which works with failure times modeled realistically as Weibull and lognormal distributions for five different aging failure mechanisms: TDDB, NBTI, EM, TC, and SM. Hence, our results are more accurate and realistic compared to previous works that are based on the assumption that lifetime distributions are exponential. We also utilized the proposed reliability evaluation methodology to develop a new lifetime aware floorplanning strategy that is capable of identifying the most reliable floorplan for a given design. We consider this an essential step toward a design approach where reliability is a primary objective.

The shrinking size of transistors and wires and the possibility of integrating millions of transistors in one chip has also raised the need for new and more efficient communication system for multiprocessor systems on chip (MPSoCs). Traditional MPSoCs used bus based systems as their communication medium. However buses are not scalable and do not support concurrent communication which is required by most of modern MPSoCs [6.15], [6.16]. Networks on Chip (NoC) has been introduced as a new communication structure for MPSoCs as a solution to this challenge [6.17], [6.18]. NoC is scalable, predictable, have higher bandwidth (compared to bus based system), and support concurrent communications. While NoC forms an efficient communication infrastructure for SoC, it is no exception to the effects of failure mechanisms. We proposed a new multi layered fault-tolerance oriented design methodology for NoCs as a hybrid solution composed of several reliability and fault tolerance design techniques applied at the software, architecture, and network-routing levels. The proposed structure for NoCs can address hard failures across 3 different levels of abstraction. At the software level, we utilize a reliability aware mapping algorithm to assign application tasks to an NoC based target architecture such that network reliability is improved. At the architecture and network-routing levels, we developed an NoC architecture that uses reconfigurable self-repairable links with spare wires and a distributed routing algorithm to dynamically detect permanent failures in unrepairable links and to recalculate routing paths using healthy links. It is the combination of these techniques and the layered approach that sets the proposed methodology apart and helps to provide a better tradeoff point between the improvement in fault tolerance and performance penalty due to the required redundancy and extra logic.

6.1. Future work

Our proposed reliability evaluation algorithm focuses on aging mechanisms that lead to hard failures. This algorithm can be extended to include early life failures which are caused during the manufacturing process. Similar techniques can also be developed to consider the effect of soft failures on the system.

Dynamic reliability management (DRM) and dynamic thermal management (DTM) design techniques are also interesting research topics that require further research and investigation. For example most of the currently available techniques like proactive temperature balancing (PTB) methods are not scalable to complex MPSoCs.

6.2. References

[6.1] D. Park, C. Nicopoulos, J. Kim, N. Vijaykrishnan, and C.R. Das, “Exploring fault-tolerant network-on-chip architectures,” Int. Conference on Dependable Systems and Networks, 2006

[6.2] J.H. Collet, A. Louri, V.T. Bhat, and P. Poluri, “ROBUST: A new self-healing fault-tolerant NoC router,” Int. Workshop on Network on Chip Architectures, 2011.

[6.3] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, “Parameter variations and impact on circuits and microarchitecture,” ACM/IEEE Design Automation Conference (DAC), 2003.

[6.4] C. Visweswariah, ”Death, taxes and failing chips,” ACM/IEEE Design Automation Conference (DAC), 2003.

[6.5] M. White and J.B. Bernstein, “Microelectronics reliability: physics of failure based modeling and lifetime evaluation,” Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, JPL publication, Feb. 2008.

[6.6] J.H. Stathis, “Reliability limits for the gate insulator in CMOS technology,” IBM Journal of Research and Development, vol. 46, no. 2/3, pp. 265, 2002.

[6.7] J.D. Owens, W.J. Dally, R. Ho, D.N. Jayasimha, S.W. Keckler, and L.S. Peh, “Research challenges for on-chip interconnection networks,” IEEE Micro, vol. 27, no. 5, pp. 96-108, 2007.

[6.8] J. Duato, S. Yalamanchili, and L. Ni, “Interconnection networks: An Engineering approach,” Published by Morgan Kaufmann, 2003.

[6.9] J. von Neumann, “Probabilistic logic and the synthesis of reliable organisms from unreliable components,” Automata Studies, C.E. Shannon and J. McCarthy Eds., Princeton Univ. Press, 1956.

[6.10] E. Moore and C.E. Shannon, “Reliable circuits using less reliable relays,” Journal of the Franklin Institute, vol. 262, no. 3, pp. 191-208, 1956.

[6.11] S. Feng, S. Gupta, A. Ansari, and S. Mahlke, “Shoestring: probabilistic soft error reliability on the cheap,” Int. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS), March 2010.

[6.12] J.P.G. Sterbenz, D. Hutchison, E. Cetinkaya, A. Jabbar, J.P. Rohrer, M. Scholler, and P. Smith, “Resilience and survivability in communication networks: strategies, principles, and survey of disciplines,” Computer Networks, vol. 54, no. 8, 2010.

[6.13] C. Constantinescu, “Trends and challenges in VLSI circuit reliability,” IEEE Micro, vol. 23, no. 4, pp. 1419, 2003.

[6.14] K. Latif, A.M. Rahmani, K.R. Vaddina, T. Seceleanu, P. Liljeberg, and H. Tenhunen, “Enhancing performance sustainability of fault tolerant routing algorithms in NoC-based architectures,” Euromicro Conference on Digital System Design, 2011.

[6.15] M. Agarwal, R. Dubey, N. Jain, and D. Raghuvanshi, “Comparative analysis of different topologies based on Network-on-Chip architectures,” International Journal of Electronics and Communication Engineering, vol. 6, no. 1, pp. 29-40, 2013.

[6.16] R. Ho, K.W. Mai, and M.A. Horowitz, “The future of wires,” Proceedings of the IEEE, vol. 89, no. 4, pp. 490-504, 2001.

[6.17] W.J. Dally and B.P. Towles, “Principles and Practices of Interconnection Networks,” Morgan Kaufmann, 2004.

[6.18] G. De Micheli and L. Benini, “Networks on Chip,” Morgan Kaufmann, 2006.