Towards Dependable Network-on-Chip Architectures

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op

maandag 18 mei 2015 om 10:00 uur

door

Changlin CHEN

Master of Engineering in Information and Communication Engineering
National University of Defense Technology, China
geboren te Taian, China

This dissertation has been approved by the promotor:
Prof. dr. K. L. M. Bertels

Copromotor:
Dr. S. D. Cotofana

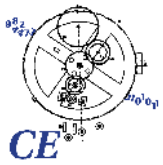
Composition of the doctoral committee:

Rector Magnificus	voorzitter
Prof.dr. K. L. M. Bertels	Technische Universiteit Delft, promotor
Dr. S. D. Cotofana	Technische Universiteit Delft, copromotor

Independent members:

Prof.dr. Y. Fu	National Universiteit of Defense Technische, China
Prof.dr. A. Rubio	Universitat Politecnica de Catalunyam, Spain
Prof.dr. K. Goossens	Technische Universiteit Eindhoven
Prof.dr. M. Berekovic	Technische Universitat Braunschweig, Germany
Prof.dr. H. Sips	Technische Universiteit Delft
Prof.dr. P. French	Technische Universiteit Delft, reservelid

The work described in this thesis has been carried out in the Computer Engineering (CE) lab. This work was supported by Delft University of Technology (TUDelft) and China Scholarship Council (CSC).



ISBN 978-94-6186-470-3

Keywords: Network on Chip, Dependability, Partially Faulty Link Utilization, Fault Tolerance, Routing Algorithm, Task Mapping, Resource Utilization

Printed in The Netherlands

Dedicated to my
Grandparents, Parents, and Wife

Towards Dependable Network-on-Chip Architectures

Changlin Chen

Abstract

The aggressive semiconductor technology scaling provides the means for doubling the amount of transistors on a single chip each and every 18 months. To efficiently utilize these vast chip resources, Multi-Processor Systems on Chip (MPSoCs) integrated with a Network-on-Chip (NoC) communication infrastructure have been widely investigated. However, the transistor miniaturization also significantly increases the possibility of transient and permanent faults occurrence inside the chip, especially for NoCs as they geometrically spread all over the chip real estate. To provide dependable communication service, the NoC must maintain its functionality and gracefully degrade its performance in the presence of faults. In this dissertation, we propose several novel NoC tailored mechanisms to tolerate faults induced by, e.g., variability agents, ageing, environmental aggression factors, as well as to efficiently utilize still functional NoC components. We first introduce a low cost method to allow for correct flit transmission even when soft errors are occurring in the router control plane. Then we propose a Flit Serialization (FS) strategy to tolerate broken link wires and to efficiently utilize the remaining link bandwidth. Within the FS framework heavily defected links whose fault levels exceed a certain threshold value are deactivated to diminish the congestion in their upstream routers. Moreover, we design a distributed logic based routing algorithm able to tolerate totally broken links as well as to efficiently utilize UnPaired Functional (UPF) Links in partially defected interconnects. We also introduce a link bandwidth aware run-time task mapping algorithm to improve the mapping quality for newly injected applications in the MPSoCs. Last but not least, we discuss the application of aforementioned strategies in 3D NoC systems and propose a Bus Virtual channel Allocation (BVA) mechanism to enable vertical wormhole switching to improve the performance of 3D NoC-Bus hybrid systems. All proposals are evaluated in our mixed language NoC simulation platform and their advantage over state of the art counterparts are proved by means of experimental results.

Acknowledgments

Here it comes the end of the four and a half years PhD study which has been full with moments of joy, sorrow, confusing, struggling, and happiness. Such a wonderful period definitely will be one of the most cherishable part of my life. However, it would be impossible for me to make it without the support of my colleagues, friends, and family. I would like to take this opportunity to express my gratitude to each and every one of you.

First of all, I own my deepest thankfulness to my supervisor Dr. Sorin Cotofana for his patient guidance, enthusiastic encouragement, and useful critiques of this research work. He gave me all the freedom to carry out the research that I am interested in; he cheered me up every time when I came across failures; he sacrificed leisure time to correct and improve my technical writing; he helped me to find the direction whenever I was confused. It is a great honor to do PhD research under his supervision and what he taught me will keep on guiding me in my future career.

I would like to thank my promotor, Prof. dr. Koen Bertels. Not every boss gives the employees more encouragement to play than that to work, and he is one of the few who does. He organized so many social events, e.g., carting, bowling, football, barbeque, Belgium beer, and spaghetti, to make the tedious research work colorful. He is always willing to help me whenever I turn to him. I also extend my thanks to the thesis committee for they spent their precious time to review my thesis and gave me valuable feedback in such limited time.

Many thanks to our CE secretary Lidwina Tromp and the associate coordinator Franca Post from the TU CICAT office, they helped me solved so many trivial but important affairs like visa application, settle down at the beginning, resident permit extension, etc. Sincere thanks to Eric and Eef for their technical support to ensure the HPC clusters run steadily throughout these years.

I really appreciate the international environment created by all CE colleagues. We came from more than twenty countries and regions, yet we were able to work together harmoniously. Special thanks go to my previous and current office-mates: Laiq Hasan, Yao Wang, Nicoleta Cucu Laurenciu, Mahroo Zan-

drahimi, Thomas Marconi, and Jiaoyan Chen. Laiq is really talented with languages. My oral English was quickly improved through chatting with him in the first few months. Yao was always generous in sharing his experience of research and life, he is truly a brother to me. Collaboration with Nicoleta was always enjoyable. Mahroo was not just a nice office-mate but also a nice neighbor. Many thanks to Marius Enachescu, George Razvan Voicu, and Mihai Lefter for our interesting chatting about history, geography, and culture, and for helping me to synthesize my design, build up the work platform, and improve my presentation. Thank Mottaqiallah Taouil for numerous jokes and laugh you brought to us and for found the glasses. God is with you all the time, Motta. Thank Mottaqiallah Taouil and Joost Hoozemans for translating my thesis abstract and propositions into Dutch. Thank Chunyang Gou very much for all the meals, barbeques, and travels we had together. My appreciation also goes to Razvan Nane, Imran Ashraf, Mafalda Cortez, Innocent Agbo, Shanshan Ren, Lei Xie, Berna Torun, M. Faisal Nadeem, Nor Zaidi Haron, and numerous other friends that I have failed to name here for the interesting talks and activities.

Life in the Netherlands would be incomplete without my dear Chinese fellows. Sincere Thanks to Zhang Li, Jianfei Yang, Shaoying Wang, Shanfei Li, Wenbo Wang, Yan Ren, Ting Hao, and Ling Zhang for the precious time we spent together and all the delicious meals we had together. Many thanks to Jing Chu for teaching me how to do physical exercise scientifically. Thanks to Chang Wang, Siqi Shen, Yunlong Li, Song Yang, Yuan He, and Jianbin Fang for our friendship.

I would also like to thank China Scholarship Council and my teachers from National University of Defense Technology: Xiang Li, Hongqiang Wang, Yaowen Fu, Weidong Jiang, and Zhaokun Qiu, without their support I would not be possible to come to TU Delft in the first place. During my stay in the Netherlands, my friends Chengguang Wu and Tianpeng Liu helped me solved many personal affairs in China, my sincere gratitude also goes to them.

Last but not least, my special thanks go to my family. Thanks to my grandparents and parents, they have always been supporting me to pursuit my dreams. In the last few years, I spent such a little time with them yet they never complained. My dear wife, Hongling Wang, thank you so much for your sacrifice and supports during my PhD study, you are the most beautiful part of my life.

Changlin Chen Delft, The Netherlands

Table of Contents

List of Tables

List of Figures

List of Algorithms

List of Acronyms and Symbols

2D	2 Dimensional
3D	3 Dimensional
AC	Allocation Comparator
ALL	Average Link Load
AP	Application
AP/S	Area*Power/Saturation_throughput
ASIC	Application Specific Integrated Circuit
AWeMD	Average Weighted extended Manhattan Distance
AWMD	Average Weighted Manhattan Distance
BIST	Built-In-Self-Test
BN	Best Neighbor
BVA	Bus Virtual Channel Allocation
BW	Buffer Write
CASqA	Contiguity Adjustable Square Allocation
CDG	Channel Dependency Graph
CeMD	Congested extended Manhattan Distance
CM	Central Manager
CMIT	Cluster Mesh Inter-layer Topology
CS	Circuit Switching
CSL	Configurable Fault-Tolerant Serial Link
CT	Crossbar Transverse
dTDMA	dynamic Time Division Multiple Access
E	East
E2E	End-to-End
ECC	Error Correcting Code
EDC	Error Detection and Correction
ELU	Effective Link Utilization
eMD	extended Manhattan Distance
FF	First Free
FIR	Flit Injection Rate
FPGA	Field Programmable Gate Array
FS	Flit Serialization
FSM	Finite State Machine
FTRA	Fault Tolerate Routing Algorithms
GALS	Globally Asynchronous and Locally Synchronous

HBH	Hop-By-Hop
HD	Heavily Defected
HDL	Hardware Description Language
HIBS	High-performance Inter–layer Bus Structure
HOL	Head Of Line
IC	Integrated Circuit
ICEB	Internal Congestion and Energy per Bit
$I - IVAD$	Input side Invalid VC Allocation Detection
INC	Incremental mapping heuristic
ITRS	International Technology Roadmap for Semiconductors
LFSR	Linear Feedback Shift Register
LSH	Least Significant Half
LT	Link Transverse
MD	Manhattan Distance
MPSoC	Multi-Processor Systems on Chip
MSH	Most Significant Half
N	North
NACK	Negative Acknowledgement
NF	Neighbor-aware Frontier
NI	Network Interface
NMRD	Normalized Mapped Region Dispersion
NN	Nearest free Neighbor
NoC	Network-on-Chip
$O - DVAD$	Output side Duplicated VC Allocation Detection
PC	Physical Channels
PFLRM	Partially Faulty Link Recovery Methods
PFLUM	Partially Faulty Link Utilization Methods
PL	Path Load
PPV	Process Parameter Variation
PS	Packet Switching
RA	Routing Algorithm
RC	Routing Computation
RC	Receiver Concave
RU	Routing Unit
RUS	Routing Unit Sharing
RX	Receiver
S	South
SA	Switch Allocation
SEC/DED	Single Error Correction and Double Error Detection

SER	Soft Error Rate
SET	Single Event Transient
SEU	Single-Event Upset
SFHS	Simple Flit Half Splitting
SFRT	Solid Fault Region Tolerant
SHiC	Smart Hill Climbing
SLLD	Standard Link Load Deviation
SoC	System-on-Chip
TC	Transimitter Concave
TDG	Test Data Generator
TDMA	Time Division Multiple Access
TED	Test Error Detector
TMR	Triple Modular Redundancy
TSV	Through Silicon Via
TX	Transmitter
ULSI	Ultra Large Scale Integration
UPF	UnPaired Functional
$UPF - FTRA$	UPF link aware Fault Tolerant Routing Algorithm
VA	VC Allocation
VC	Virtual Channel
VCID	Virtual Channel Index
VCT	Virtual Cut-Through
VNT	Vertical Node Tree
W	West
WS	Wormhole Switching

1 Introduction

Since the first Integrated Circuit (IC) prototype was demonstrated in 1958 [46], the semiconductor fabrication technology feature size has been continuously scaled down driven by consumers' demands for higher performance and lower power consumption. This spectacular evolution enables the transistors amount on a single chip to be doubled in every 18 months [105]. To efficiently utilize the vast amount of chip resources and address issues like long global wire delay, system synchronization, and design productivity, the digital system paradigm has been evolved and sequentially experienced the room-, rack-, board-, and chip-level systems. As the number of processors in chip-level systems increases towards Multi-Processor Systems on Chip (MPSoC), the Network-on-Chip (NoC) paradigm [23] has been proposed and is still widely investigated as a scalable and reliable communication infrastructure replacement of buses and crossbars [11]. However, transistor miniaturization also makes the manufacturing yield and chip dependability increasingly serious concerns. The chips are becoming more prone to various kinds of failures caused by issues like single event upset [52], manufacturing defects [40], chip wear-out effects [14], Process Parameter Variations (PPVs) [42, 98], etc., especially for NoCs, which geometrically spread all over the chip real estate. Given that the NoC is the MPSoCs backbone, to avoid significant system performance degradation due to fault occurrence, its dependability need to be improved by means of mechanisms located at different NoC abstraction levels, e.g., circuit, architecture, and their introduction constitutes the focal point of this dissertation.

In this chapter, we discuss the necessity to implement NoC based MPSoCs in modern ICs in Section 1.1, present state of the art ICs dependability issues and their corresponding NoC design challenges in Section 1.2, highlight the dissertation contributions in Section 1.3, and introduce the dissertation organization in Section 1.4.

1.1 Network-on-Chip

Conventionally, a System-on-Chip (SoC) consists of a single processor, required peripherals, and buses or crossbars to connect the processor and peripherals. As the chip size and required performance soar, the design paradigm shifts towards Multi-Processor Systems on Chip (MPSoCs) in which the single high performance processor is replaced by multiple low performance ones and the buses/crossbars are replaced by a Network-on-Chip (NoC). We note that MPSoCs are also multi-core systems thus in this dissertation, we deem an MPSoC processor as being equivalent with a core in the multi-core system and we use the two terms, processor and core, interchangeably unless otherwise stated.

1.1.1 From Single Processor to Multi-Processor SoCs

Many techniques have been utilized to improve the single processor SoC performance, with increasing chip frequency being the most straightforward one. However, the maximum clock frequency cannot be increased "ad infinitum" without any limitations and undesired consequences [75]. Even though it is possible to run a processor core at a high frequency, e.g., 6GHz, it is not wise to run the entire chip at such high speed as this significantly increases the chip power density [50]. Other strategies to improve a processor's performance include adding architectural features like hyper-threading, superscalar, out-of-order execution, branch prediction, etc., at the expense of higher design and validation efforts [12].

In fact, parallelism is always one of the best ways to improve performance and this concept has been successfully applied to SoCs resulting in the introduction of MPSoCs [75]. By replacing the single high performance processor with multiple low performance ones, the same or even higher computation power can be achieved while operating at lower voltage, frequency, and power density. Moreover, when compared with single processor SoCs, MPSoCs are more reliable due to their inherent redundancy, i.e., when one processor is broken, its tasks can be taken over by other functional ones. The strict global synchronization requirement can also be released if MPSoCs are implemented as Globally Asynchronous and Locally Synchronous (GALS) systems [18]. Last but not least, MPSoCs can speed up the time-to-market as they enable the existing processors reuse.

Nowadays, MPSoCs embedding tens to hundreds of processing cores have

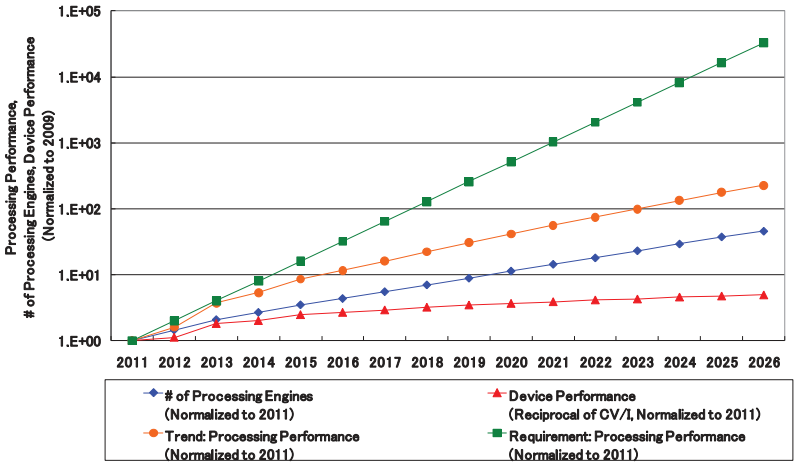


Figure 1.1: ITRS roadmap [48] for the number of processing cores, provided and required processing performance.

been fabricated, e.g., Teraflops [106] with 80 cores and Ambric [104] with 336 cores. As illustrated in Fig. 1.1, the International Technology Roadmap for Semiconductors (ITRS) predicts that by 2026, there will be chips with upwards of 10x more cores than current MPSoCs, while the gap between the required and provided processing performance is still widening. ITRS also predicts that the number of cores increases linearly in the foreseeable future, and by implication the intra-system communication requirements, thus the on chip interconnection infrastructure must be scalable in order not to become the system bottleneck.

1.1.2 From Bus and Crossbar to Network-on-Chip

As transistor size shrinks and the SoC design paradigm shifts from computation centric towards communication centric, the on chip communication, which was considered to be cheaper than computation, starts to become a major contributor to the SoC performance and implementation cost.

Traditionally, the SoC on-chip interconnects have followed the conventional bus or crossbar structures. However, as the number of processing cores increases, buses and crossbars are becoming the system bottleneck due to their low scalability.

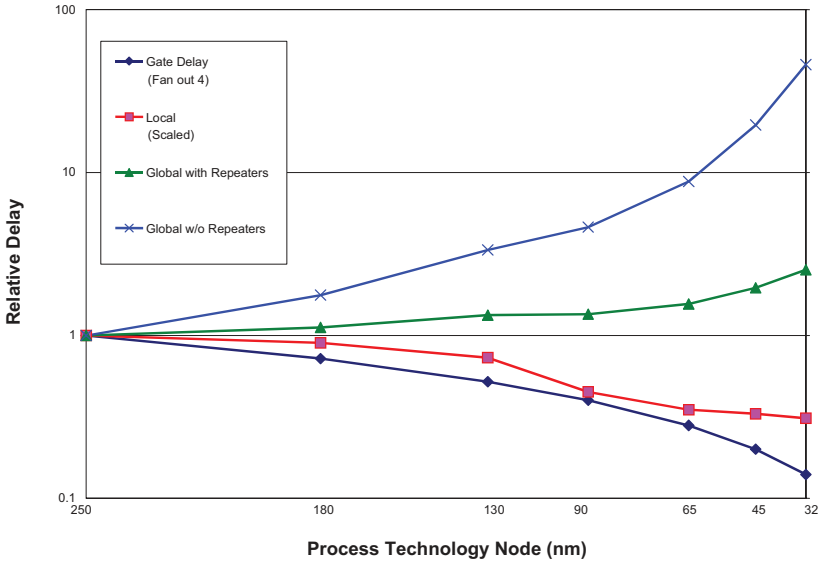


Figure 1.2: ITRS [47] projected relative delay for wires and logical gates in different technologies.

A bus is by nature a sequential data transport medium as by an arbitration process it is exclusively assigned to one source and destination pair in each and every clock cycle. To enable concurrent data transmission, buses can be segmented [64] at the expense of higher arbitration complexity or replaced with crossbars at the expense of a significantly larger number of wires. However, segmented buses and crossbars are not fully scalable and thus should be perceived as intermediate solutions.

As (MP)SoCs are getting more and more complex, long wires are extensively utilized in buses and crossbars. However, as semiconductor technology scales down, the wire resistance per mm is increasing and long wires become more expensive in terms of power consumption. Moreover, wires scale much slower than transistors do [11], thus wire delays rather than gate delays are becoming the dominant contributors to the clock period length [99]. As illustrated in Fig. 1.2, for the $32nm$ technology node, global wires are already more than $100x$ slower than gates, and still about $10x$ slower even when repeaters are utilized. This makes the system level synchronization very challenging and limits the bus and crossbar maximum operating frequency.

The aforementioned issues can be addressed by interconnecting the cores with an NoC [23], which is constructed from multiple point to point data links inter-

connected by routers, such that messages can be relayed from any source node to any destination node over several links, by routing decisions performed by the involved routers [107]. Network Interfaces (NIs) are implemented between cores and the NoC to decouple computation from communication by packing and unpacking the messages. Note that messages are usually transmitted in the form of packets, and each packet is further split into several flits, which have the same width with the NoC's links. More details about NoC architectures are presented in Chapter 2.

When compared with bus and crossbar, NoC is not just more scalable, but also can operate at much higher clock frequency and has lower power consumption [3]. For example to interconnect 64 cores, both bus and crossbar require large arbiters to control the data flow, while in a 2 dimensional (2D) mesh NoC, each router just connects with the local core and 4 neighboring routers thus small arbiters are required. In addition, neighboring routers are interconnected with short wires thus NoCs can work at higher frequency. The wires in buses and crossbars are fanned out to all their targets while NoC links offer point to point connection between adjacent routers, thus NoCs also have lower dynamic power consumption.

NoC is also intrinsically more reliable than bus and crossbar. An NoC usually provides multiple routing paths between any source and destination pair, thus if one path is broken, the messages can be detoured along alternative paths. Nevertheless, as the transistor size scales down, the fault variety and occurrence frequency are increasing, which make the design of dependable NoCs a real challenge.

1.2 Research Challenges

A system is dependable if it is able to offer service without failures that are more frequent and more severe than acceptable [6]. The acceptability level is very much dependent on the application nature, demands, and operating environment, thus dependability requirements for different chips may be vary quite significantly. However, as the transistor size keeps on scaling, the chips are generally becoming more prone to various kinds of dependability issues and thus have to deal with more errors [22].

Generally speaking, the dependability issues can be classified into the following categories:

- **Single-Event Upset (SEU).** An SEU happens when the normal state of

a logic unit is flipped by high energy particles, e.g., neutrons and protons, and the resulting logic glitch is propagated to an output or captured by memory units [52]. SEU occurrence rate is related to, e.g., transistor size, power supply value, chip area, and increases with technology scaling. Thus state of the art ICs, MpSoCs included, are more sensitive to SEU occurrence in both computation and data transport parts.

- **Process Parameter Variations (PPVs).** PPVs are sourced from random dopant fluctuations, sub-wavelength lithography, and heat flux which is time and context variant [14]. As the worst-case design strategy is usually employed to ensure correct system functionality in all potentially possible operating conditions, increased PPVs not just bring more design challenges but also reduce the manufacturing yield, which increases the costs and diminishes the technology scaling benefits [98].

- **Manufacturing Defects.** Manufacturing defects occur due to the imperfection of the chip production steps. As the transistor size scales down, the expectation of getting a fault-free chip from manufacturing process drops significantly. Note that manufacturing defects need to be detected and masked to avoid abandoning the entire chip [49].

- **Wear-out Effects.** As technology scales, time-dependent wear-out effects, e.g., electromigration, hot carrier degradation, and time dependent oxide breakdown, are getting stronger and the chip lifetime is obviously shortened. Although the chip aging process can be potentially predicted and monitored by various kinds of aging models [59, 102] and sensors [58, 103], the chips must be periodically diagnosed to detect worn-out components and to deal with them by means of proper techniques.

Due to the aforementioned dependability issues, transient, intermittent, and permanent faults may occur in MPSoCs which could lead to computation and data transmission errors and eventually in service failures. We note that: (i) transient faults occur randomly but rarely at the same location, (ii) permanent faults do not disappear once they happened and their amount increases with chip aging, and (iii) intermittent faults exhibit the same syndromes as permanent faults but they vanish after a short time period.

As it is impossible to prevent fault occurrence, they must be dealt with to maintain the system functionality and ensure that the system performance gracefully degrades during its operational lifetime. As the SoC design shifts from computation-centric to communication-centric, and NoCs geometrically

spread all over the chip real estate, NoC dependability is becoming a key contributor to the dependability of the entire system. In view of this, in this dissertation, we combat NoC dependability issues at the architecture level, and mainly address the following:

- **Soft error occurance in the router control plane.**
  Message data bits can be flipped when SEU induced soft errors happen in links and/or routers' data path, i.e., input/output buffers and crossbars. Such errors can be detected and corrected by means of various kinds of Error Correcting Codes (ECC) [80]. However, when soft errors happen in the router control plane, packets or flits could be transmitted to wrong output ports even that the data correctness is not affected. Note that such errors cannot be detected by ECC means and require novel soft error tolerant strategies.

- **Effective utilization of still functional resources in permanent fault affected NoCs.**
  Due to dependability issues, NoC links and routers may be affected by permanent faults, which need to be tolerated to maintain the basic NoC functionality. Equally important, the still functional resources should be effectively utilized to achieve graceful performance degradation. In this line of reasoning links with a small portion of broken wires are dealt with Partially Faulty Link Utilization Methods (PFLUMs), e.g., [61, 72, 100], and totally broken links are bypassed by detouring the packets along alternative fault free paths by means of Fault Tolerate Routing Algorithms (FTRAs), e.g., [1, 16, 17, 19, 29, 39, 56, 81, 101, 116]. However, state of the art proposals focus on fault tolerance give little attention to the effective utilization of the remained partially functional NoC resources. For example, all PFLUMs double the link transmission latency even if the link contains only one broken wire, and most FTRAs discard the entire interconnect between two adjacent routers despite the fact that only one of the two links is broken. Note that we assume that an interconnect between two adjacent routers is composed of a pair of unidirectional links, each link having its own control flow wires and handling either outgoing or incoming traffic. In view of the previous discussion we can conclude that PFLUMs that can utilize the remained link bandwidth more efficiently and FTRAs that can make use of UnPaired Functional (UPF) links in partially defected interconnects are required.

- **NoC link bandwidth variation aware application mapping.**
  In NoC based MPSoCs, applications are usually split into sets of con-

current tasks, which are mapped onto different processor nodes to enable their parallel execution. Existing run time task mapping heuristics, e.g., [15, 20, 21, 33, 34, 36], perceive NoC links as being either fully functional or totally broken despite of the fact that when partially broken links are utilized and their diminished bandwidth is carefully considered, a better mapping quality could be achieved. In view of the fact that partially broken link utilization is a centric point of our research the identification of novel mapping heuristics able to take advantage of such links can be viewed as its natural continuation at a higher abstraction level. An essential aspect in this context is that the identification/definition of new task mapping metrics able to better reflect link bandwidth variations is required to select the best processing node candidate for each application task.

- **Vertical link dependability improvement in 3-dimensional (3D) NoC.**
  With the emerging of 3D IC stacking, various 3D NoC architectures have been proposed [82]. In 3D chips, silicon tiers are vertically stacked and connected with Through Silicon Vias (TSVs) [7] which, when compared with moderate size planar wires, exhibit extremely low data transmission latency, but suffer from low manufacturing yield [63]. As most 2D NoC principles can be applied to each silicon layer, the main challenge in 3D NoCs relates to the vertical links' implementation and utilization. Thus 3D NoC designs that can exploit the benefit of negligible TSV delay while improving the vertical link dependability are essential in 3D MPSoC implementations.

We note that in this dissertation, we focus on generic NoC dependability issues which are independent to the cores, while the dependability issues related with NIs are quite specific to the type of cores they attached to [11] that NIs are not considered. However, as long as NIs pack/unpack messages according to the NoC required packet structure, all NI dependability improvement strategies, e.g., [38, 84], are applicable in conjunction with our proposals in this dissertation.

1.3 Dissertation Contributions

The main goal of this dissertation is to augment the NoC dependability at, but not limited to, the architectural level by: (i) improving the NoC fault toler-

ance capabilities, (ii) efficiently utilizing remained partially functional NoC resources, (iii) mapping applications into the NoC in awareness of the faults and link bandwidth variation, and (iv) designing new dependable NoC infrastructures. The final goal being to construct NoCs able to deliver trusted communication service to the MPSoCs computation units during their expected lifetime. In this section, we highlight the main contributions of the research work described in this dissertation, as follows:

- We propose a low cost method to tolerate soft errors potentially occurring in router control plane functional units, i.e., routing units, Virtual Channel (VC) allocators, and switch allocators. Rather than relying on a Triple Modular Redundancy based implementation of each functional unit, we choose to exploit the intrinsic redundancy available in the router hardware structures and signals. In essence we detect Routing Computation (RC) errors by comparing RC results from the local Routing Unit (RU) and idle RUs available at neighboring input ports. The RC results are recalculated in case errors are detected or neighboring RUs are not available. We detect errors in the VC Allocation (VA) and Switch Allocation (SA) results by checking if they are consistent with the correct RC results, each NoC resource is exclusively assigned to one request initiator, and each request initiator is allocated only one NoC resource. VA/SA errors are corrected by redoing the failed procedures and retransmitting the flits. Experimental results on an 8×8 2D NoC indicate that: (i) in the routing units, the proposed method requires 38% more silicon real estate than the Σ & Branch method when the XY routing algorithm is utilized, but it is more general and can be utilized in conjunction with other routing algorithms; and (ii) in the combined VA/SA units, the proposed method is simpler and more effective than state of the art counterparts. When compared with the Triple Modular Redundancy strategy, for similar error detection and correction capabilities, the proposed method can reduce the area and power overhead in routing units by 53% and 38%, respectively, and in combined VA/SA units by 45% and 46%, respectively. The average packet transmission latency is less than 5% higher than the one of the baseline router with no soft error detection/correction mechanisms even if the soft error rate is as high as 0.1 errors/router/cycle.

- We propose a Flit Serialization (FS) method to tolerate broken link wires and to effectively utilize the remained link bandwidth. The FS approach divides the links and flits into several sections, and serializes sections

of adjacent flits to transmit them on all available fault-free link sections to avoid the complete waste of partially defective links. The proposed transmitter and receiver are transparent to the router such that their utilization is not constrained by the router architecture and implementation or network topology. Experimental results obtained on synthetic traffic and PARSEC benchmarks indicate that FS reduces the latency overhead significantly and enables graceful performance degradation when compared with related partially faulty link utilization proposals. It reduces area cost and power consumption by up to 29% and 43.1%, respectively, when compared with spare wire replacement methods, and can achieve lower area*power/saturation_throughput values than all state of the art link fault tolerant strategies. We also propose the link augmentation with one redundant section as a low cost mechanism to further increase the link dependability. Experimental results indicate that when 10% of the NoC wires are broken, adding a redundant section to each link can improve the NoC saturation throughput by 18%.

- We propose a distributed logic based Fault Tolerant Routing Algorithm (FTRA) to tolerate broken links and efficiently utilize the UnPaired Functional (UPF) links in partially defected interconnects. The basic fault pattern tolerated by the UPF link aware FTRA (UPF-FTRA) is a fault wall, which is composed of adjacent broken links with the same outgoing direction. Messages are routed around the fault walls along the misrouting contours of the broken links. The proposed Routing Algorithm (RA) requires at least 3 VCs and dynamically reserve them to the detoured messages to avoid deadlock. Our experiments indicate that, for random and localized traffic patterns, we achieve an average saturation throughput 20% higher than the Solid Fault Region Tolerant (SFRT) RA, and 22% and 14% higher than the Ariadne routing table based RA, respectively. Simulation results on PARSEC benchmarks also suggest that UPF-FTRA provides much lower packet transmission latency than SFRT and Ariadne. Synthesis results with Synopsis Design Compiler and TSMC 65nm technology indicate that, embedding the proposed RA into a baseline router results in 9% area overhead, which is only 1% higher than that of SFRT and does not increase for bigger size NoCs.

- We introduce a strategy to differently treat partially faulty links that have different fault levels as follows: (i) links whose fault level is lower than a threshold are still utilized by means of the FS method, while (ii) Heavily Defected (HD) links whose fault levels exceed the threshold are de-

activated and dealt with the UPF-FTRA. Although utilizing HD links can preserve more NoC link bandwidth, they can actually cause high congestion in the upstream routers and significantly degrade the system performance. As the FS induced link flit transmission latency increases slowly when the link fault level is low but fast when the fault level is high, the optimal threshold can be easily determined by comparing the zero load packet transmission latency on the HD links and that on the shortest alternative path. Simulation results we obtained at various wire broken rate configurations indicate that we achieve the highest saturation throughput if 4- or 8-section links with a flit transmission latency longer than 4 cycles are deactivated.

- We propose a run-time task mapping algorithm, which takes both the path traffic load and link bandwidth variation into consideration and maps applications onto contiguous near convex NoC regions to reduce the internal and external congestion. We rely on a backtracking strategy to guaranty that the maximum link traffic load does not exceed a given limit determined by the link bandwidth and a loose factor. Note that the loose factor is employed to adjust the maximum percentage of link bandwidth that can be utilized. To evaluate our proposal we map synthetic and real video processing applications on partially defective 8×8 NoCs. The experiments indicate that our approach substantially outperforms equivalent state of the art task mapping heuristics when NoC defects are present, e.g., for 5% broken wires, we achieve at least 16% communication cost reduction and 45% shorter average packet transmission latency.

- We propose a Bus VC Allocation (BVA) mechanism to enable vertical Wormhole Switching (WS) in 3D NoC-Bus hybrid systems. We note that by implementing the vertical 3D NoC links as buses we can exploit the benefit of negligible TSV delay by running the bus at a higher frequency than its planar NoCs and reduce the amount low manufacturing yield TSVs by letting multiple routers resident on one tier to share the same bus. In such an NoC-Bus hybrid system, data are usually transmitted according to the Packet Switching strategy because enabling vertical Wormhole Switching (WS) in the conventional way requires a large amount of TSVs. The BVA mechanism address this issue by assigning in each layer to at most one cross layer packet a free input VC in its target router before injecting the packet into the bus. In this way, a routing path is reserved by the head flit, and the rest of the packet flits can be WS

transmitted through the vertical buses. Given that VC allocation is performed only once per packet per hop BVA can be implemented in such a way that it doesn't become a system bottleneck. We evaluate our proposal with both synthetic and PARSEC benchmarks. The experimental results indicate that when compared with conventional pipelined bus or Time Division Multiple Access (TDMA) bus based systems, implementing vertical WS can reduce the bus critical path length by at least 31%, diminish the average packet transmission latency by at least 22%, and save the area cost and power consumption of the output buffers incident to the bus by 47% and 43%, respectively.

1.4 Dissertation Organization

The reminder of this dissertation is organized as follows:

Chapter 2 introduces the essential NoC background knowledge by covering aspects as NoC topology, routing algorithm, switching policy, and router architectures. We also present the mixed language NoC simulation platform we utilize to evaluate and validate the contribution described in this thesis, the strategies to inject synthetic traffic and real application traces into the NoC, and the NoC performance evaluation metrics.

Chapter 3 presents our approach to tolerate soft errors occurring in the NoC router control plane, i.e., routing units, VC allocators, and Switch Allocators. The implementation details related to the error detection and correction are described.

Chapter 4 introduces the Flit Serialization (FS) method which can efficiently utilize remained bandwidth in partially defective NoC links. We also propose a low cost link augmentation method with one redundant link section to make up the FS drawback that the link bandwidth is reduced even if a link contains only one broken wire.

Chapter 5 presents our approach to tolerate Heavily Defected (HD) links. Specifically, we first find the optimal threshold to deactivate HD links by comparing the zero load packet transmission latency on the HD links and that on the shortest alternative path, and then propose a fault tolerate routing algorithm to tolerate deactivated links and to efficiently utilize unpaired functional links.

Chapter 6 introduces 4 new mapping quality evaluation metrics which take the link bandwidth variations into consideration and presents the link bandwidth aware backtracking based run time task mapping algorithm. This algorithm

consists of two sub-algorithms which are utilized to search for a near square free NoC region and to map the newly injected application into the selected region, respectively.

Chapter 7 introduces the Bus VC Allocation (BVA) mechanism to enable vertical wormhole switching data transmission in 3D NoC-Bus hybrid systems and discusses its utilization in pipelined and Time Division Multiple Access (TDMA) vertical buses. To deal with potential transient and permanent faults in 3D NoCs, we also discuss the application of our aforementioned fault tolerant techniques to detect and correct soft errors in bus VC allocators, to utilize partially faulty vertical buses, and to tolerate deactivated or totally broken vertical buses.

Finally, **Chapter 8** concludes our work and provides outlook on potential future work.

2

NoC Background Knowledge

In this chapter, we introduce essential NoC background knowledge. Specifically, we first present the basic NoC components with a simple example, and then introduce the NoC architecture by covering the aspects as network topology, routing algorithms, and switching policies. After that, a router architecture embedding wormhole switching technology is described. We also present the mixed language NoC simulation platform we utilize to evaluate and validate the contribution described in this thesis, the strategies to inject synthetic traffic and real application traces into the NoC, and the NoC performance evaluation metrics.

2.1 An NoC Example

A simple NoC example structured as a 4×4 2D mesh is presented in Fig. 2.1. An NoC consists of Network Interfaces (NIs), routers, and links.

- **Network interfaces** act as the interface between the cores and the NoCs to decouple computation from communication. In the NIs, data to be injected into the NoC are packed into packets and received packets from the NoC are unpacked.
- **Routers** are utilized to route packets to the destination according to the employed routing protocols. Based on the NoC topology, each router can be connected with multiple cores and neighboring routers. We note that the cores can be processing or memory units.
- **Links** connect adjacent routers. The interconnect between two adjacent routers are usually composed of two unidirectional links which in charge of the outgoing or incoming traffic, respectively. It is possible to replace

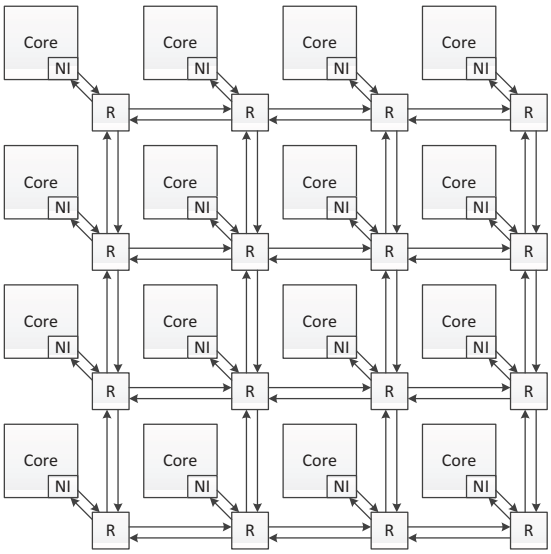


Figure 2.1: A NoC example with 4 × 4 2D mesh topology. R: Router. NI: Network Interface. Each router is connected with 1 cores and 4 neighbor routers.

the unidirectional links with bidirectional ones [97]. However, unidirectional links are still attractive as they provide better means to implement the control logic and to address timing error issues [95]. In this thesis, we focus on NoCs that utilize unidirectional links.

2.2 NoC Architecture

Based on the target application context, a NoC places the routers and connects them with links according to a certain topology and routes the packets according to the most suitable routing protocols. The routing protocols are implemented with corresponding router structures.

2.2.1 NoC Topology

Some basic NoC topologies are illustrated in Fig. 2.1, among which the most common one is 2D mesh. When compared with 2D mesh, 2D torus has more routing path diversity but also longer links between routers, fat tree and butterfly can better exploit the traffic locality but have fixed routing path and thus

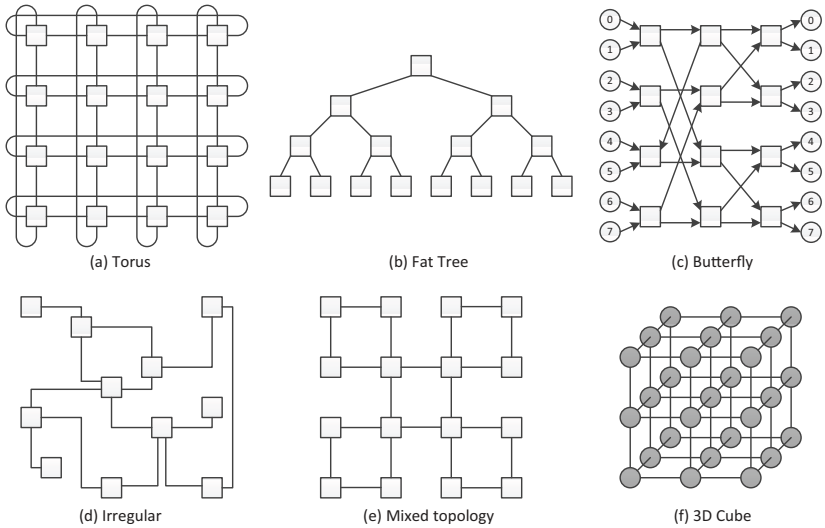


Figure 2.2: NoC topology examples. In (c), all interconnects are unidirectional. In (d), the interconnects can be bidirectional or unidirectional according to the system requirements. In other topologies, all interconnects are bidirectional.

lower reliability. The irregular topologies, e.g., Fig. 2.2(d), are usually application specific and thus can be optimized for the target applications. By mixing multiple topologies together, e.g., Fig. 2.2(e), one can balance the versatility and specificity of the NoC system. With the advent of 3D IC era, various 3D NoC topologies are proposed [82]. On each silicon layer, all the aforementioned 2D NoC topologies can be applied. The stacked layers are connected with Through Silicon Vias (TSVs). In Fig. 2.2(d), we present a 3D cube as an example, in which the vertical links can be implemented in the same way as planar links to provide point-to-point connection between vertically adjacent routers, or be implemented as buses to provide all-to-all connection among all routers in the same Z-pillar. In this dissertation we focus on the 2D mesh NoC topology in chapter 3 to 6, and expand the applications of 2D NoC technologies to 3D NoC systems in chapter 7.

2.2.2 Routing Algorithm

Routing Algorithms (RAs) are utilized to determine the routing path of each packet from the source node to the destination node. A well designed RA is able to find the shortest routing path, balance the traffic load, and tolerate broken routers and links to enable graceful NoC performance degradation [24].

Generally, the RAs must be carefully designed to avoid deadlocks and live-locks. As an exception, some RAs allow deadlocks but provide methods to detect and recover from them.

The RA can be deterministic or adaptive. Deterministic RAs always route packets along the same path between a given source/destination pair, while adaptive RAs are able to select paths according to the network status to route packets around congested or faulty regions. Based on whether minimal routing paths are always provided, adaptive RAs can be referred as minimal or non-minimal. The most typical deterministic RAs are XY (for 2D mesh) and e-cube (for hypercubes), in which packets sequentially traverse every dimension. Opt-Y [88] is an minimal adaptive RA example as it offers two admissible output ports to each packet when the destination node is not in the same row or column with the current router. Non-minimal adaptive RAs are usually utilized as Fault Tolerant Routing Algorithms (FTRAs), e.g., [17], as they can find alternative paths when the minimal paths are broken.

2.2.3 Switching Policy

The switching policy determines how the data are transmitted along the routing path. The most commonly utilized ones are Circuit Switching (CS), Packet Switching (PS), Virtual Cut-Through (VCT), and Wormhole Switching (WS).

With the CS technology, a path from the source to the destination is formed before the data transmission by reserving the routers and links with a probe message. It is suitable to the case where the message transmission time is much longer than the path set up time. It has the advantage of low buffering needs at the routers, but has the drawback that the reserved routers and links cannot be utilized by other users.

To save the path set up time and enable flexible resource utilization, the long messages can be divided into multiple packets and be transmitted with the PS technology. The routing and control information is stored in each packet head. At each hop, i.e., intermediate router, the entire packet is buffered and then the head information is extracted to determine the downstream router over which the packet should be forwarded to.

In fact, once the packet head is received, the output port can be determined without waiting until the entire packet is received. This means that the already received part can be transmitted to the downstream router while the reminder part is still being received. Such switching technique is referred as VCT. With VCT, an entire packet is only buffered in a router when the packet head is

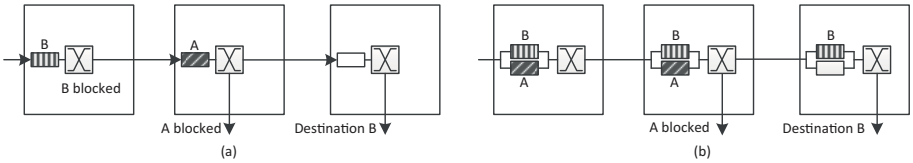


Figure 2.3: Use VCs to solve blocking issues. (a) Both packets A and B are blocked. (b) Packet A is blocked, but packet B can still be transmitted.

blocked. When compared with PS, VCT has reduced packet transmission latency as the time to wait for integral packets is removed.

PS and VCT require large buffers in the routers to buffer integral packets. With WS, the buffer size can be reduced. A packet is split into a certain number of flits, i.e., a head flit, several body flits, and a tail flit, which have the same width as the links. The head flit carries the routing and control information to reserve the routing path at each hop for the body and tail flits. The buffer in each route port is only required to be able to store a few flits. When the head flit is blocked, a packet may occupy buffers in multiple routers. To avoid blocking the transmission of packets with different destinations, a physical channel is usually shared by a number of Virtual Channels (VCs). Accordingly, the buffer in each router physical port is divided into the same number of VC buffers. When one packet is blocked, other packets in the same physical port but using different VCs can still be transmitted as illustrated in Fig. 2.3. In the rest part of this dissertation, we assume that VC based WS technique is always utilized in the NoC unless otherwise stated.

When WS is implemented, an upstream router can only transmit flits when free buffer slots exist in the downstream router. The buffer availability can be managed with the credit-based mechanism illustrated in Fig. 2.4. The upstream router uses a counter to actively maintain the number of buffer slots, i.e., credits, in each downstream VC. Once a flit is transmitted, a credit is consumed and the counter decrements by 1. In case the counter value is zero, all downstream buffer slots are occupied and no flits can be transmitted until a buffer slot becomes available again. In the downstream router, once a flit is forwarded and the associated buffer is freed, a credit is returned to the upstream router, causing the counter increments by 1.

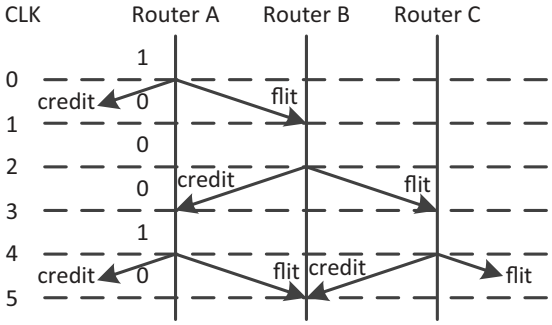


Figure 2.4: The credit-based buffer management mechanism. The numbers in the second column, below "Router A", indicate the number of available buffer slots in the downstream VC.

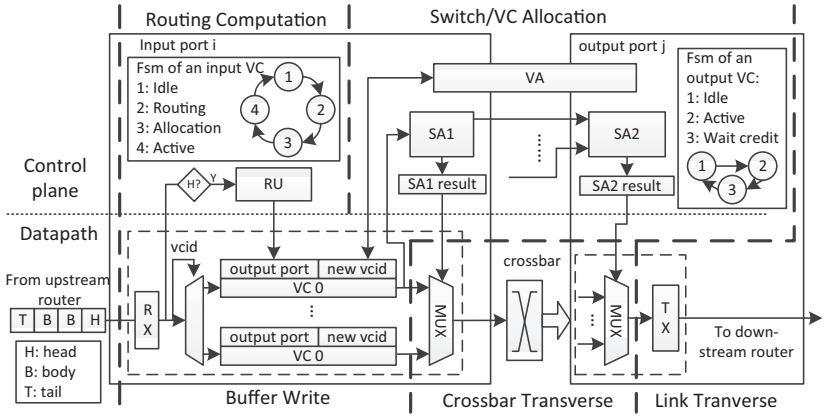


Figure 2.5: A typical VC based router architecture. RU: routing unit; VA: VC allocation; SA: switch allocation.

2.3 Router Architecture

The block diagram of a typical VC based router architecture is depicted in Fig. 2.5. The router functional units can be partitioned into the datapath group, which handles the storage and movement of the packets, and the control plane group, which is responsible for coordinating the movement of packets through the resources of the datapath [24]. The datapath consists of input buffers, multiplexors, the crossbar, and output buffers (if implemented). The control path consists of Routing Units (RUs), VC allocators, switch allocators, and other control logic.

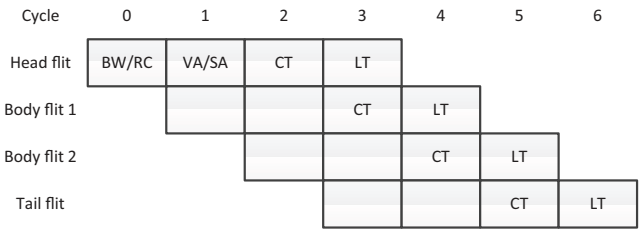


Figure 2.6: The pipeline stages to transmit a packet.

2.3.1 Router Pipeline

As illustrated in Fig. 2.6, the pipeline to transmit a flit consists of the following steps:

- **Buffer Write (BW)**: A flit arrives at a router input port and is written into the VC buffer indicated by the VC index (VCID) transmitted along with the flit.
- **Routing Computation (RC)**: If it is a head flit, the destination information is extracted and the output port is computed. BW and RC usually happen in the same clock cycle.
- **VC Allocation (VA)**: According to the output port, the VC allocator assigns the packet a free output VC, i.e., an input VC in the downstream router. RC and VA do not exist for body and tail flits.
- **Switch Allocation (SA)**: Each flit requests for a time slot on the crossbar and the output port from the switch allocator when VC credits exist. In modern routers, VA and SA are usually speculatively implemented in the same cycle to reduce the number of router pipeline stages.
- **Crossbar Transverse (CT)**: The flit is transmitted from the input VC to the output port through the crossbar.
- **Link Transverse (LT)**: The flit is transmitted to the downstream router.

2.3.2 Virtual Channel States

As illustrated in Fig. 2.7, the VCs in each input port can be organized in distributed or centralized ways, which are suitable to be implemented in Application Specific Integrated Circuits (ASICs) and Field Programmable Gate Arrays

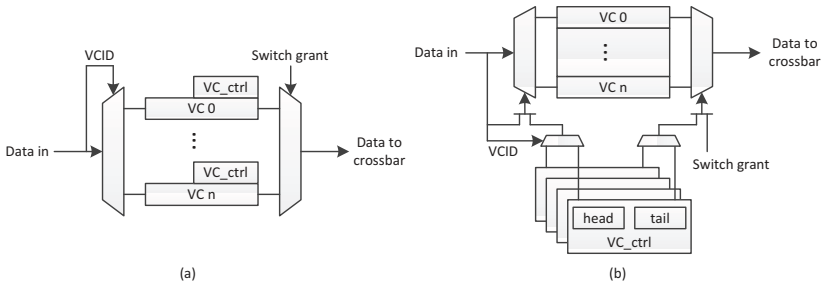


Figure 2.7: Two ways to organize VC buffers. (a) Each VC is independently implemented; (b) All VC buffers are implemented in one block memory.

(FPGAs), respectively. The states of input and output VCs are maintained by Finite State Machines (FSMs) at the input and output side, respectively.

An input VC can be in one of the four states: *idle*, *routing*, *allocation*, and *active*. An input VC is initially *idle*. When it receives a head flit, it enters the *routing* state where the routing information is computed. After an output VC is allocated to the packet, the input VC stays *active* until the entire packet is successfully transmitted and then returns to *idle*.

The state of an output VC can be *idle*, *active*, or *wait credits*. An output VC is initially *idle* and becomes *active* once it is assigned to an input VC. After tail flit of the packet which hold the output VC is transmitted, it waits until all credits are returned and then becomes *idle* again.

2.3.3 Speculative Virtual Channel and Switch Allocation

We can implement VA and SA in the same pipeline stage by speculating that they can both succeed. A head flit asserts the VA/SA request only when free output VC(s) is/are available in the target output port, thus if it won the SA arbitration, it can be granted a free output VC. Fig. 2.8 illustrates an implementation of the speculative VA/SA mechanism proposed by Lu et al. [65].

The SA is usually divided into two stages, i.e., SA1 and SA2, which are implemented in the input port and the output port, respectively. The SA1 arbiter in each input port selects one of the requests from the VCs that have pending flits in their buffers ①. The selected request is then forwarded to the SA2 arbiter in the target output port along with its type i.e., VA or SA ②. The SA2 arbiter performs arbitration among the requests from different input ports ③. The SA2 results are then routed back to the relative input ports ④. A request is granted if it won both SA1 and SA2 ⑤. The allocation process finishes or continues

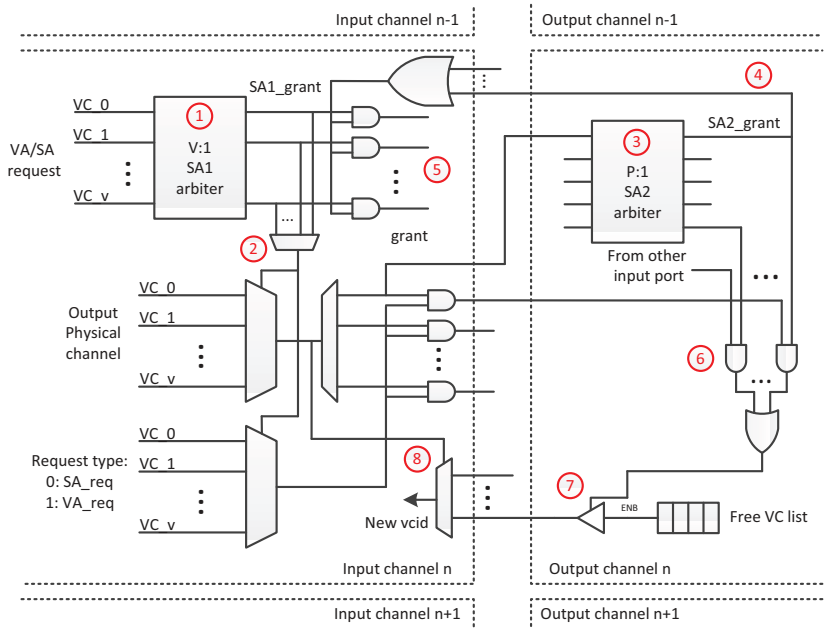


Figure 2.8: An implementation of speculative VA/SA. Each router has p physical ports and each port is shared by v VCs.

based on the request is a SA request or a VA request, respectively ⑥. In the latter case, a free output VC is picked out from the free VC list and is broadcast to the input ports ⑦. At the input port side, the correct new VCID is selected according to the destination of the granted request ⑧. Since the requests which do not have free VCs in their target output ports have been removed by the request regulating logic and at most one request can be granted by each output port, the request initiator which won SA is guarantied a free output VC.

2.4 Simulation Platform

The proposals in this dissertation are evaluated in our mixed language simulation platform with both synthetic traffics and real application traces. The platform is developed based on the one designed by Lu et al. [65]. The synthetic traffic generator and the NoC infrastructures, i.e., NIs, routers, and links, are implemented with Verilog Hardware Description Language (HDL). The real application traces are read from the records and then be injected into the NoC platform with C language. The simulation platform is illustrated in Fig. 2.9.

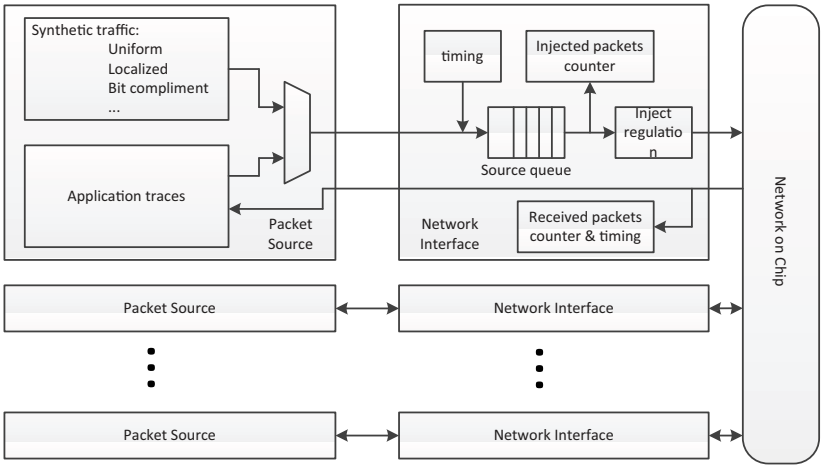


Figure 2.9: Structure of the simulation platform.

2.4.1 Synthetic Traffic

The NoC behavior differ considerably from one architecture to another and from one application to another. As there has been no standard traces to evaluate the NoC performance, most researchers and designers refer to synthetic workloads with different characteristics. A synthetic traffic pattern can be defined by the destination distribution and the traffic load which is indicated by Flit Injection Rate (FIR) and packet length [27].

The most frequently utilized destination distribution is the uniform one. In this distribution, a node sends packets to any other nodes with the same probability. In our simulation platform, we use Linear Feedback Shift Registers (LFSRs) to generate pseudo-random numbers which are then translated to destination node coordinates. The case of nodes sending packets to themselves is excluded as the network is not utilized. The NoC performance with uniform traffic can be treated as the upper bound on the mean communication distance.

In practice, most probably the application mapping is optimized and each node just communicates with nodes in short distance, case in which the localized traffic pattern can better reflect the NoC performance. The destination distribution in localized traffic pattern can be sphere of locality or decreasing probability distribution [85]. In the former case, a node communicate with nodes inside a sphere with the same high probability ϕ and with the nodes outside of the sphere with probability $1 - \phi$. In the latter case, the probability of a node sending packets to another one decrease as their distance increases. In

our simulation platform, the former distribution is utilized.

Another synthetic traffic patten employed in this dissertation is bit compliment, in which the node with binary coordinates $(a_0, a_1, ..., a_{n-2}, a_{n-1})$ communicates with the node $(\overline{a_0}, \overline{a_1}, ..., \overline{a_{n-2}}, \overline{a_{n-1}})$.

2.4.2 Real Application Traces

To be more accurate, the NoC performance should be evaluated with application-driven workloads. To this target, the simulation can be performed in a "full-system" platform consists of both NoC and processors, e.g., M5 [10], with the intended applications running on the processors. This approach has the drawback that the traffic generated by the applications can be influenced by the NoC feedback, which makes it difficult to locate the NoC bottleneck [24].

Alternatively, we can record the message from an application in advance and then replay the traffic trace in the simulation. Since the NoC feedback cannot change the recorded traffic trace, the simulation results can better reflect the NoC performance. Although this approach may decrease the simulation accuracy, it can significantly reduce the simulation platform's complexity and the simulation time. In this dissertation, we evaluate our proposals with traffic traces of PARSEC benchmarks [9] which are recorded with the Netrace [43] tool on the M5 full system simulator. The traces are replayed according to each packet time flag while maintaining the packets dependencies.

2.4.3 Task Mapping Benchmarks

One target of our research is to optimize the application mapping in the MPSoCs in awareness of the link bandwidth variation. The mapping algorithm is evaluated with TGFF [26] generated applications and four video processing applications [8]. Each application is split into a certain number of tasks. The communication between two tasks is characterized by the traffic volume and FIR. Note that we assume that the FIR does not change during the application's execution. In our experiments, the application mapping algorithms are implemented with C language. After every task is allocated to a processor, the tasks information, in terms of communication volumes and FIRs of the communication edges, are sent to the utilized processors via the control network [21]. The processors then generate communication messages according to the received task parameters.

2.4.4 Evaluation Metrics

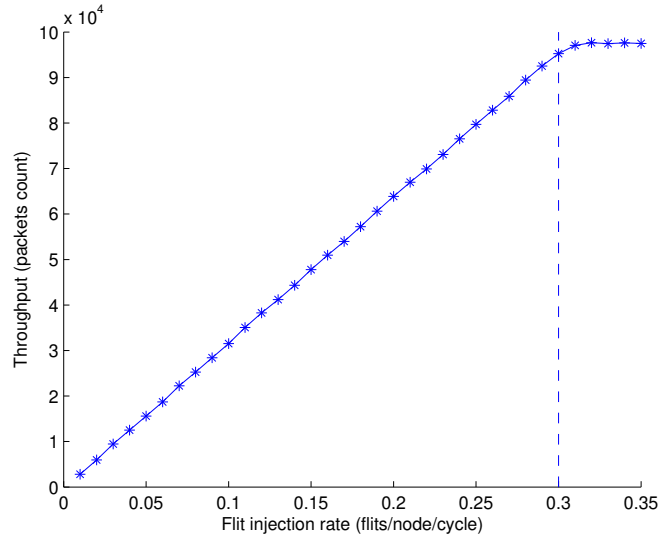
Among the various metrics to evaluate the NoC performance, the most frequently utilized three in our research are latency, throughput, and fault tolerance.

Latency is the time, i.e., the number of clock cycles, required to transmit a packet from the source to the destination since the packet head flit is generated in the source node till the last flit is received by the destination node, i.e., the source queuing time is included. In the early stage of the NoC development, the performance can be estimated with the zero-load latency, i.e., the average packet transmission latency when there is only one packet traverses the network at any time. Although it ignores the influence of contention among the packets, the zero-load latency provides a fast way to characterize a new structure's effect on the NoC performance. More generally, the contention should be considered and the network performance should be evaluated at different FIRs.

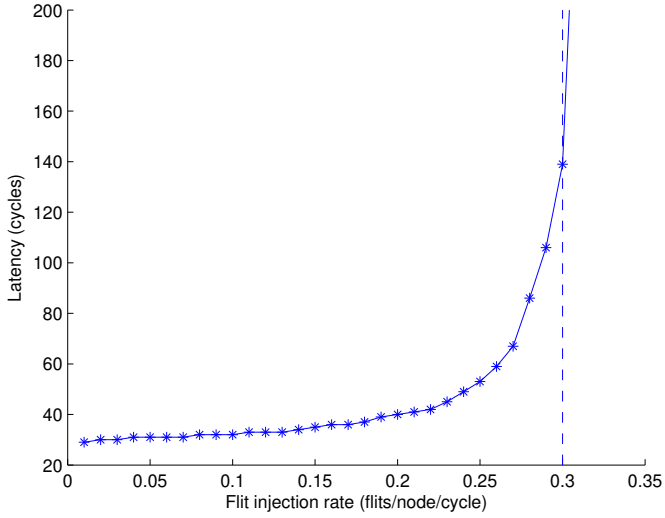
Throughput is the maximum amount of information delivered per time unit [27]. Strictly speaking, throughput is the amount, or portion, of the offered traffic that be accepted by the NoC. Note that the offered traffic can be measured with FIR, i.e., the rate of flits be generated by the packet source.

Fig. 2.10 plots the changes of throughput and average packet transmission latency at different FIRs in a baseline 8×8 2D mesh NoC. We can observe that as the FIR increases, the NoC throughput increases linearly until the NoC be saturated when the FIR is 0.3 flits/node/cycle, and the packet latency starts at the horizontal asymptote of zero-load latency and slopes upward to the vertical asymptote of saturation throughput. Now that the throughput has linear relationship with FIRs before the NoC is saturated and the packet transmission latency approaches infinity at the saturation throughput, we just depict the latency vs. FIR graphs in the rest part of this dissertation.

Fault tolerance is the NoC's capability to perform its functionality in the occurrence of faults. The faults can exist in the NIs, routers, or links. Fault tolerance is of vital importance to the modern NoC based MPSoCs as it is almost impossible to manufacture fault free chips due to the aggressive scaling of semiconductor technology. In principle, we expect the NoC performance degrades gracefully as the number of faults increases. As fault tolerance is hard to be quantified, it can be reflected by the packet transmission latency and saturation throughput when faults exist, and the number of faults that can be tolerated by the NoC architecture.



(a) NoC throughput.



(b) Average packet latency;

Figure 2.10: Performance of a 2D 8 $\times$ 8 mesh NoC under uniform traffic using XY routing algorithm. Each physical port is shared by 4 VCs. The buffer depth of each VC is 4-flits. The packet length is 4-flits.

2.5 Conclusions

In this chapter, we introduced the essential NoC background knowledge by covering the aspects as NoC topology, routing algorithm, switching policy, and router architectures. We also presented the mixed language NoC simulation platform we utilize to evaluate and validate the contribution described in this thesis, the strategies to inject synthetic traffic and real application traces into the NoC, and the NoC performance evaluation metrics.

In the following chapters, we present our contributions to improve the NoC dependability, evaluate their performance on our mixed language NoC platform, and compare the merits with that of state of the art counterparts.

3

Soft Error Tolerance in Router Control Plane

Soft errors in the Network-on-Chip (NoC) links and router datapath flip flits data bits but can be easily detected and corrected through the use of Error Correcting Codes (ECCs). Conversely, soft errors in the router control plane do not corrupt flits but cause packets or flits be transmitted to wrong output ports and are hard to detect. In this chapter, we solve this issue with a low cost method. The idea behind our proposal is to detect Routing Computation (RC) errors by comparing RC results from the local Routing Unit (RU) and idle RUs available at neighboring input ports. The RC results are recalculated in case errors are detected or neighboring RUs are not available. We detect errors in the VC Allocation (VA) and Switch Allocation (SA) results by checking if they are consistent with the correct RC results, each NoC resource is exclusively assigned to one request initiator, and each request initiator is allocated only one NoC resource. The VA/SA errors are corrected by redoing the failed procedures and retransmitting the flits. Experimental results on an 8×8 2D NoC indicate that: (i) in the routing units, the proposed method requires 38% more silicon real estate than the Σ & Branch method when the XY routing algorithm is utilized, but it is more general and can be utilized in conjunction with other routing algorithms; and (ii) in the combined VA/SA units, the proposed method is simpler and more effective than state of the art counterparts. When compared with the Triple Modular Redundancy strategy, for similar error detection and correction capabilities, the proposed method can reduce the area and power overhead in routing units by 53% and 38%, respectively, and in combined VA/SA units by 45% and 46%, respectively. The average packet transmission latency is less than 5% higher than the one of the baseline router with no soft error detection/correction mechanisms even if the soft error rate is as high as 0.1 errors/router/cycle.

3.1 Introduction

Due to miniaturization, permanent and soft errors occurring in the cores and Network-on-Chips (NoCs) are becoming more frequent and in order to prevent application misbehavior one should detect and correct them. In this line of reasoning, we focus on tolerating soft errors in NoCs in this chapter.

NoCs are composed of routers and links. As illustrated in Fig. 2.5, the components in a router can be partitioned into the datapath group, which mainly consists of memory elements, and the control plane group, which mainly consists of combinational logic circuits [24]. Previous research has been focusing on tolerating soft errors in links [74, 113] and router datapath [68], which are supposed to be more susceptible to soft errors than the router control plane. However, the Soft Error Rate (SER) in nowadays combinational logic circuits is already comparable with that encountered in unprotected memory elements [109]. Consequently, we further concentrate our attention to soft errors in the router control plane.

The main functional units in the router control plane are Routing Units (RUs), Virtual Channel (VC) allocators, and switch allocators. For each received packet, the RU compute the output port, the eligible output VCs at that output port when adaptive routing algorithms, e.g., Opt-Y [88], are employed, and the necessary misrouting information when fault tolerant routing algorithms, e.g., [19], are utilized. The VC allocator chooses one free eligible output VC and assigns it to the packet. Switch allocators decide which flits can be transmitted in the next clock cycle. Soft errors in these functional units can lead to deadlock or flits loss during packets transmission.

A soft error occurs only if a Single Event Transient (SET) is propagated to an output and latched into a memory element [52]. During the propagation, SETs can be diminished by logical masking, electrical masking, and temporal masking mechanisms [91]. Circuit design methods, e.g., gate resizing [25], circuit rewiring [109], and output multiple sampling [5], have been proposed to exploit these masking mechanisms. However, these methods have the drawbacks of high physical level design effort and high chip area overhead.

In NoC routers, it is inherently required that one output resource can only be assigned to at most one request at a given time. This feature is exploited in [55, 74,115] to detect soft errors with low silicon overhead. However, such schemes either have restricted application scope [115], or are still too complicated to allow for practical implementations [74].

In this chapter, we propose a low cost method to tolerate soft errors in the main

functional units in the router control plane.

By noticing that the RU is usually replicated at each input ports, especially if it is logic based, and it is only utilized when a head flit arrives at that port, we propose to detect Routing Computation (RC) errors by comparing RC results from the local RU and idle RUs available at neighboring input ports. The RC results are recalculated in case errors are detected or neighboring RUs are not available. The proposed method requires 38% more area overhead than the Σ & Branch method proposed in [115] when XY routing algorithm is utilized on a 2D mesh NoC, but is applicable in conjunction with many more routing algorithms. When compared with the Triple Modular Redundancy (TMR) strategy, for similar error detection and correction efficacy, our method can reduce the area and power overhead in RU by 53% and 38%, respectively, when Opt_Y routing algorithm [88] is utilized.

One common requirement for the VC Allocation (VA) and Switch Allocation (SA) results is that they should be consistent with the RC results. Moreover, they must correspond to legal states, i.e., one output resource can maximally be assigned to one request. In view of these, we propose to detect the illegal VA results at the output port side by checking if multiple head flits were transmitted to the same output VC. The method to detect illegal results of arbiters in [115] is expanded in this chapter to detect erroneous SA results and is implemented with simple logic. The VA/SA errors are corrected by redoing the failed procedures and retransmitting the flits. When compared with the Allocation Comparator (AC) unit in [74], our method is easier to implement and more reliable. If utilized in the implementation of the combined VA/SA units, which are embedding matrix arbiters, the proposed method can reduce the TMR area and power overhead strategy by 40% and 46%, respectively.

In the rest part of this chapter, Section 3.2 briefly introduces the methods to tolerate soft errors in the links and router datapath for the integrity of the contents. Section 3.3 presents a brief survey of state of the art strategies to tolerate soft errors in the router control plane. Section 3.4 introduces the proposed soft error tolerant method. Section 3.5 introduces the strategy to recover from the soft errors. Section 3.6 evaluates the proposed method and compares it with tightly related work. Section 3.7 concludes the presentation.

3.2 Soft Errors in Links and Router Datapath

Soft errors in NoC links and router datapath, i.e., input and output buffers, multiplexers, and crossbars, can be detected and even corrected using an ECC.

Among numerous ECC proposals, the Single Error Correction and Double Error Dectetion (SEC/DED) codes are the most commonly utilized ones.

The error detection and correction can be performed Hop-By-Hop (HBH) or End-to-End (E2E). In the HBH strategy, the correctness of each flit is checked at the input port of each router. If the error cannot be corrected, a Negative-Acknowledgment (NACK) signal is sent back to the upstream router and the flit is retransmitted. In the E2E strategy, the correctness of each packet is checked at the destination router. If there are errors in the packet, a NACK message is sent back to the source router and the whole packet is retransmitted. Both approaches require dedicated buffers to keep a clean copy of the data. The HBH approach is more popular than E2E as it has shorter round trip to recover from the errors and thus requires smaller retransmission buffers.

Soft errors happen in the memory units do not disappear until be corrected. Thus to maintain the clean data copies in the router buffers, it is also important to periodically sweep the buffers by reading each location and correcting any single-bit errors, in case the second error occurs and then the flit cannot be recovered.

3.3 Soft Errors in The Control Plane – Related Work

The most intuitive way to tolerate soft errors in the functional units is TMR, with the obvious drawback of high area and power consumption overhead.

A Σ & Branch method is proposed by Yu et al. [115] to detect soft errors in RC results by examining the appearance of forbidden signal patterns in RUs. This method is proved to be efficient to detect erroneous output ports, because each packet can only be transmitted to one output port unless multicast is needed. However, it cannot detect errors in other routing results, e.g., eligible output VCs, which can include multiple existing output VCs.

Park et al. [74] proposed an Allocation Comparator (AC) unit to detect errors in VA results. The output VC of each input VC is compared with the routing results to check if it is an eligible one, and with that of other input VCs to check if the output VC is occupied already. Considering that an output VC can be assigned to any input VC, such a comparison cannot be implemented with trivial effort.

To detect erroneous SA results, Kim et al. [55] proposed to add a message ID to each flit and check if the flits received by one input VC have the same ID. The drawback of this method is that the link width is increased and the

errors are detected in downstream routers. Differently, Park et al. [74] and Yu et al. [115] check if the SA results are consistent with the RC results and have legal states after they are registered. In this chapter, we apply this checking principle to VC based NoC routers and implement it with simple logic.

3.4 Soft Errors Detection

As illustrated in Fig. 2.5, the transmission of a head flit in the router must sequentially go through 3 stages: Routing Computation (RC), speculative VC Allocation (VA) and Switch Allocation (SA), and Crossbar Transverse (CT). An extra Link Transverse (LT) stage is usually required between two neighboring routers. Body and tail flits just need to go through the SA and CT stages. Each input VC has four states: *idle*, *routing*, *allocation*, and *active*. An input VC waits for routing results in the *routing* state and VA results in the *allocation* state. After that, it stays in the *active* state until the entire packet is successfully transmitted.

In this section, we assume that the requests to do relative computations, i.e., RC, VA, and SA, are asserted correctly. In fact, the logic to generate the requests can be protected by TMR method with negligible cost due the their simplicity.

3.4.1 Errors in Routing Units

RUs decide how packets should be transmitted according to the employed routing algorithm. Erroneous RC results can misdirect packets and lead to deadlock or packet loss. Correct RC results are the precondition of correct VA and SA results.

In a typical NoC router design, the RU is replicated at each input port to increase throughput. Each RU deals with local routing requests only, and is only utilized when head flits arrive at that port. Assuming that the packet length is P, the average RU utilization rate is less than $1/P$, because flits do not arrive in every cycle. This means that RUs are most of the time idle and we propose to utilize idle neighboring RUs, when available, to do redundant routing computation for local routing requests.

The proposed RU Sharing (RUS) method is illustrated in Fig. 3.1. When a head flit is received at an input port, the routing request is sent to the local RU as well as the RUs in the two neighboring ports. At each port, the highest

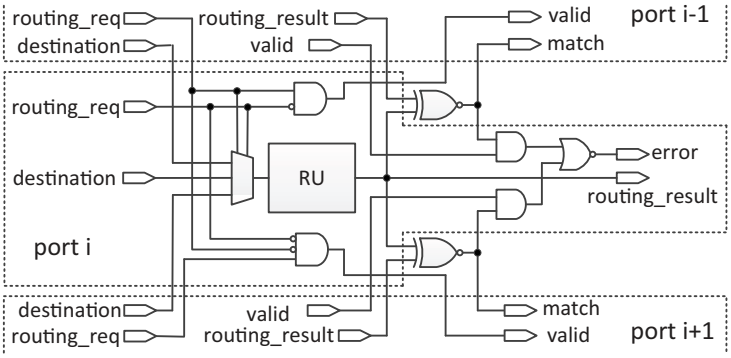
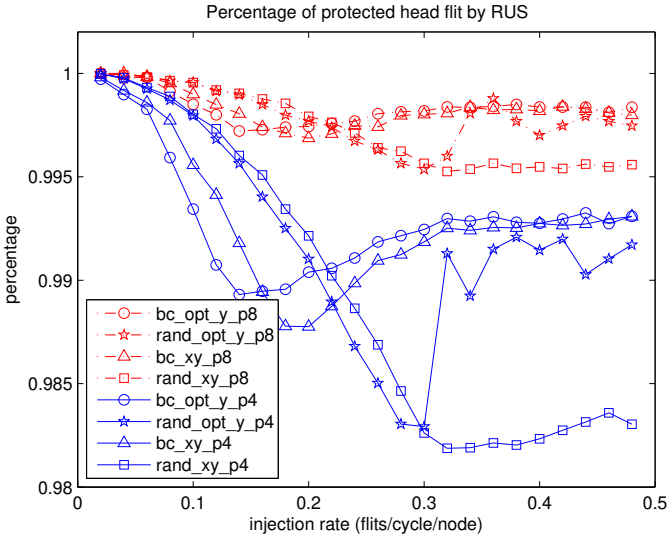


Figure 3.1: Routing unit sharing among neighboring input ports.

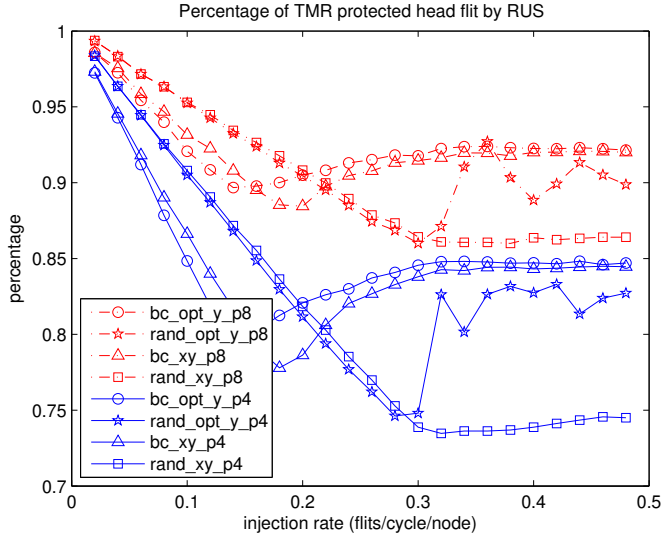
priority to utilize the RU is given to the local routing request. When the local routing request is not asserted, the priority is given to a neighboring port, e.g., port $i - 1$ in Fig. 3.1. RC results from two neighboring RUs are compared. The comparison result along with a *valid* signal are sent back to the routing request initiator input port. The *valid* signal indicates whether the RC results are computed for that port.

To save silicon cost and limit the critical path length increase in the RC stage, the RC results from neighboring ports are only utilized to check the correctness of local results. When erroneous RC results are detected, the RC is re-executed for the packet. We note here that with the overhead of a voter at each port, the correct RC result can be determined when two idle neighboring RUs are available for a routing request.

In practice, depending on the executed application and traffic, situations may occur when both neighboring RUs are occupied and only the local RC results are available at an input port, i.e., such RC results are not protected from soft errors. In Fig. 3.2 and Fig. 3.3, we illustrate the probability that RC results are checked for correctness in various situations in an 8×8 2D mesh NoC. We can observe in Fig. 3.2(a) that for all the evaluated synthetic traffic patterns, more than 98% of the RC results are protected in each router, even if the NoC is saturated. When the traffic load is low, almost every head flit can 'borrow' RUs from neighboring ports. For the real applications from the PARSEC benchmarks [9], our evaluations indicate (see Fig. 3.3(c)) that more than 99.9% of their packets are protected. Moreover, more than 72% of the packets in synthetic traffics (Fig. 3.2(b)) and more than 97% of the packets in the real application traffic traces (Fig. 3.3(d)) are TMR protected. The statistic results also reveal that the longer the packet length, the higher the probability that the

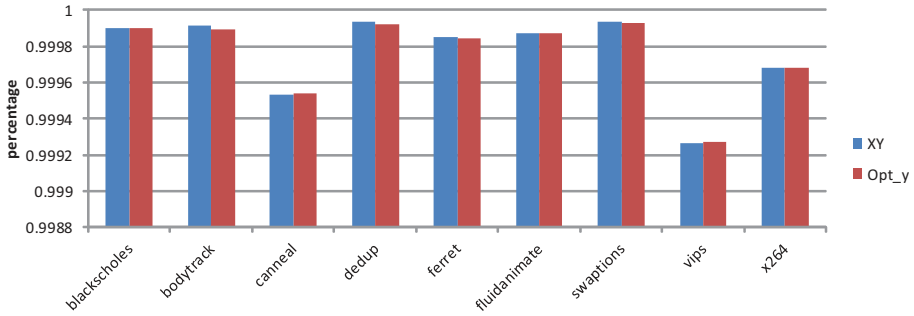


(a) Percentage of protected packets in synthetic traffics;

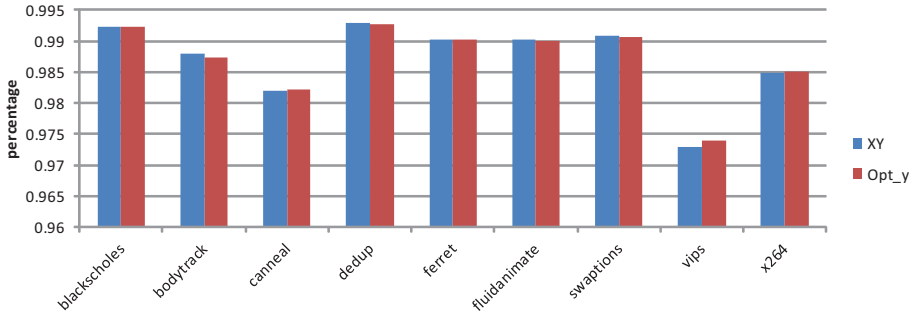


(b) Percentage of TMR protected packets in synthetic traffics;

Figure 3.2: Percentage of packets protected by RUS in synthetic traffics. Bit compliment (bc) and random (rand) traffic patterns, XY and Opt_Y routing algorithms, packets with a length of 8-flits (p8) and 4-flits (p4) are evaluated.



(a) Percentage of protected packets in real applications.



(b) Percentage of TMR protected packets in real applications.

Figure 3.3: Percentage of packets protected by RUS in real applications. XY and Opt_y routing algorithms are evaluated.

RC results are protected.

In case the unprotected RC results are contaminated by soft errors, we let such packets wait until at least one neighboring RU is available to make sure that the RC results can be checked for correctness. Due to the fact that only a small percentage of packets, e.g., less than 0.1% for PARSEC benchmarks, need to wait for idle neighboring RUs and reassert the routing request, the induced packet transmission latency overhead is negligible.

However, a deadlock situation can happen in this case. When a head flit is received at every input port simultaneously, every RU is occupied by the local routing request. In the next cycle, the local routing request at every port is asserted again by these head flits or newly received ones. Thus the deadlock lasts and eventually all the input VCs are occupied and no packet can go further.

The deadlock can be solved by temporarily prohibiting the routing request of one input port, such that one input port has an idle neighboring RU to utilize.

For the sake of simplicity, we always prohibit the routing request from the port connected with the local processing unit in such cases. Most probably the routing results from the two RUs match with each other and the port will not assert routing request until a new head flit arrives. In this way after several cycles, every packet gets the correct RC results and enter the next stage.

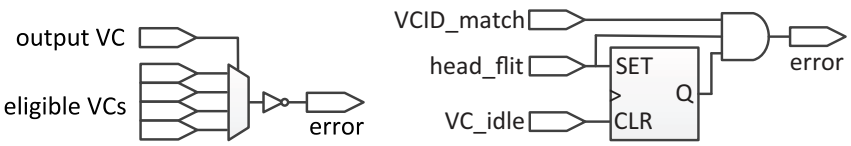
Routing results are stored in registers and must be maintained unchanged until the entire packet is successfully transmitted. Otherwise, the next flits will deviate from the path reserved by the head flit. We assume that these registers are protected against soft errors with proper technology, e.g., [90].

3.4.2 Errors in VC Allocators

After the output port of a packet is derived, the VC allocator assigns a free output VC at that output port to the packet. The VC's index (VCID) can be changed when errors happen in the VA unit. Then the output VC may become: (1) a wrong output VC, which does not exist or is not an eligible VC according to the RC results; or (2) an eligible output VC which is already assigned to another packet.

Type (1) errors can be easily detected by checking if the granted output VC is consistent with the RC results [74]. Assuming that eligible output VCs are indicated by setting the relative bits to '1' in the RC results, the error can be detected by the logic in Fig. 3.4(a). We name this method as Input Side Invalid VC Allocation Detection (I-IVAD).

In a router which has p physical ports and v VCs at each port, an output VC can be assigned to any of the pv input VCs. It is expensive to check if any two input VCs are granted with the same output VC by doing pairwise comparisons [74]. As it is guaranteed by I-IVAD that each input VC is assigned with an eligible output VC, we propose an Output Side Duplicated VC Allocation Detection (O-DVAD) method (see Fig. 3.4(b)) to detect type (2) errors.



(a) I_IVAD; **(b)** O_DVAD;

Figure 3.4: Detect soft errors in VA result.

In O-DVAD, a 1-bit latch is added to each output VC with the initial state '0'. Once a head flit is transmitted to the downstream router, the latch of the relative output VC turns to '1' to indicate that the output VC is occupied. The latch stays in '1' until the output VC returns to the *idle* state. If another head flit is transmitted to the same output VC when the latch is '1', it means that an attempt is made to assign the output VC to multiple packets and the error is detected. The input VC that is related with the error can be determined based on the SA results. The output VC remains under the utilization of the first packet it was assigned to.

I-IVAD and D-DVAD are implemented in the CT stage. The strategy to recover VA errors is presented in detail in Section IV. Similar with the RC results, we assume that VA results are stored in registers protected against soft errors.

3.4.3 Errors in Switch Allocators

Switch allocation is usually divided into the local stage (SA1) at the input side and the global stage (SA2) at the output side (see Fig. 2.5). An input VC can transmit flits only when it won both SA stages. The main components in switch allocators are arbiters.

The 4 symptoms of soft errors in SA results are explicitly discussed in [74]: (1) no asserted request is granted in SA1 or SA2, (2) an input VC is allocated with a wrong output port, (3) in the same cycle, multiple input VCs at one input port are granted in SA1, or multiple input ports are granted by SA2 at one output port, or an input port is granted by multiple output ports, and (4) an input VC which does not assert request wins both SA1 and SA2.

An efficient method to detect erroneous results of arbiters is presented in [115]. We apply this method in VC based routers to detect erroneous SA results. Note that the error detection logic has to be applied to SA1 results, SA2 results, and SA2 grants of each input port.

The SA results correctness is checked in the CT stage after they are registered. Thus soft errors in the registers can also be detected. In case the errors are caused by flipped priority registers in arbiters, the relative arbiters are reset whenever an SA error is detected [115] such that correct results can be obtained by redoing the allocation.

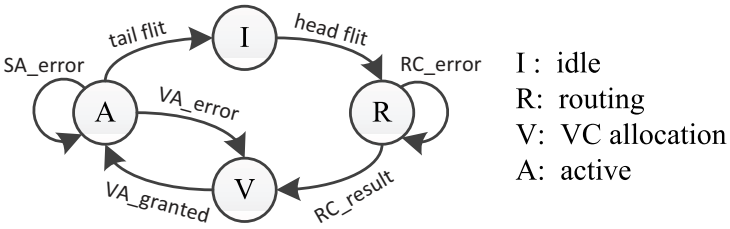


Figure 3.5: Changes of input VC states.

3.5 Soft Error Correction

To recover from errors in the control plane, the failed procedures must be repeated to generate correct results, and flits have to be retransmitted inside the router when necessary. During the recovery, the states of input VCs change as illustrated in Fig. 3.5.

Errors in RC results (*RC_error*) are detected in the RC stage. When an *RC_error* occurs, the input VC stays in the *routing* state instead of entering the *allocation* state. The routing request is reasserted when no new head flit arrives at the input port. Most probably the correct routing results will be derived with one clock cycle overhead. The head flit is not transmitted yet in this stage. So there is no need to do any retransmission.

Both errors in VA results and SA results are detected in the cycle after they are registered, i.e., in the CT stage, such that the critical path in the VA/SA stage is not changed. If an error is detected in the VA result, the input VC returns to the *allocation* state from the *active* state. Both VA request and SA request have to be reasserted to compete for a new output VC and flit transmission. The mistakenly allocated output VC needs to be released by the input VC. If an error is detected in SA results, only the SA requests need to be reasserted for flit retransmission. When a VA or SA error is detected, the VA/SA results derived in the same cycle are abandoned. Thus at least two extra clock cycles will be introduced.

In the CT stage, flits are read out from input buffers and are sent to output ports. If a VA or SA error is detected, the relative flit is ignored by the link registers at the output port, and then retransmitted when correct VA and SA results are derived. Because the original flit is still kept in the input buffers at this stage, retransmission buffers are not required.

It is also possible that there is no soft error in the control plane but an error signal is asserted by the error detection logic. Then the error recovery mech-

anism is triggered, which does not introduce new errors but only increase the flit transmission latency with several cycles.

3.6 Evaluation

The different soft error tolerant approaches are evaluated in the context of a wormhole switched 8×8 2D mesh NoC system. The routers are VC based and implemented according to the architecture in Fig. 2.5. Each router has 5 physical ports, and each PC has 4 VCs. Note that the proposed method is also applicable in 3D symmetric NoC system, as the only difference is that the routers have 7 PCs in such a system.

As soft errors happen only when SETs propagate to an output and are captured by registers [52], instead of injecting soft errors into the gates in the functional units, we simulate the symptoms of soft errors by injecting soft errors into the final RC, VA, and SA results. We assume that: (i) only one error can happen in one router in a cycle, and (ii) the router datapath is error free.

3.6.1 Reliability

Simulations with different soft error rates illustrate that the proposed method can detect and recover all kinds of errors in RC, VA, and SA results, when assumption (i) is satisfied. Thus the NoC system can operate normally unless errors happen in both the functional units under protection and the error detection logic circuits.

RUS provides TMR protection of the RC results in most of the time (see Fig. 3.2 and Fig. 3.3). Thus the RUS reliability for each packet is similar with that of TMR. Although the Σ & Branch method [115] is claimed to be more reliable than TMR in the detection of erroneous output ports, it cannot detect errors in other RC results, e.g., eligible output VCs, which are required in sophisticated routing algorithms, e.g., Opt_Y. From this aspect, RUS has much larger application scope than Σ & Branch.

To detect the erroneous VA results when one output VC is assigned to multiple input VCs simultaneously, the proposed method requires one D-latch and one AND gate for each output VC. In contrast, the AC unit in [74] compares the output VC of each input VC at the input side. Given that in a VC based router, an input VC can be assigned with any output VC at any output port, the AC unit is quite complicate to implement. In the evaluated router architecture, at least

Table 3.1: Area and power of soft error tolerant methods for RU

	Area (μm^2)		Dyn. Power (μW)		Leak. Power (μW)	
RA	XY	Opt_Y	XY	Opt_Y	XY	Opt_Y
Router*	64812.6	65231.3	24690.7	24720.1	452.7	454.1
baseline	219.6 100%	638.3 100%	18.0 100%	47.4 100%	0.758 100%	2.174 100%
RUS	465.8 212%	1018.4 160%	52.8 294%	95.6 200%	1.768 233%	4.027 185%
TMR	774.0 352%	2157.5 338%	60.1 335%	153.1 321%	3.004 396%	8.191 376%
Σ & Branch	336.6 153%	– –	25.6 143%	– –	1.290 170%	– –

* The router makes use of baseline matrix arbiters and RUs.

950 XOR gates are required to do the comparison in the AC unit. Assuming that the SER of a D-latch is 10 times higher than that of a gate [23], the SER ratio of O-DVAD to AC unit is $220/(950+X)$, where X is the number of other gates besides XOR gates in an AC unit. So the O-DVAD logic is more reliable than the AC unit.

3.6.2 Area and Power Overhead

Silicon area is an important issue that affects the chip reliability as large area overhead implies a higher chance to be hit by high energy particles, hence a higher SER [52].

RUs and combined VC/switch allocators, equipped with different soft error tolerant methods, are implemented at RTL level by using Verilog HDL, and synthesized using the Synopsys Design Compiler with TSMC 65-nm standard cell technology. The area costs and power consumption of different methods are illustrated in Table 3.1 and Table 3.2, respectively. The area and power costs of a router embedding baseline RUs and arbiters are also illustrated as a reference.

We note that RUS does not increase the number of RUs in the baseline router, and only requires half of the number of the comparators required by the TMR strategy. Thus the implementation cost of RUS is much lower than that of TMR, especially when complicated routing algorithms are employed. To be

Table 3.2: Area and power of soft error tolerant methods for VA/SA. R-R: Round-Robin.

	Area (μm^2)		Dyn. Power (mW)		Leak. Power (μW)	
Arbiter type	Matrix	R-R	Matrix	R-R	Matrix	R-R
Router*	64813	61597	24.7	24.1	452.7	434.9
baseline	10010	6794	2.1	1.5	62.1	44.3
	100%	100%	100%	100%	100%	100%
proposed	12828	10127	2.5	1.8	81.0	66.7
	128%	149%	119%	120%	130%	151%
TMR	23351	13824	4.6	2.4	147.7	95.5
	233%	203%	219%	160%	238%	216%

* The router makes use of baseline arbiters and RUs. The underlying routing algorithm is XY.

specific, the RUS area overhead is 53% less than that of TMR when Opt_Y routing algorithm is utilized, while the reduction is 40% when XY routing algorithm is utilized. The RUS power consumption exhibits the same trend. Although the RUS area and power costs are all higher than those of the Σ & Branch method [115], it is more general can be applied in conjunction with many more routing algorithms.

The cost to tolerate soft errors in VC and switch allocators in our method is proportional with the numbers of output ports and VCs in a router, regardless of the allocators implementation details. Table 3.2 suggests that the proposed methods reduce the TMR area overhead by 45% and 27% when matrix arbiters and round-robin arbiters are utilized, respectively. The power consumption of the proposed method is also lower than that of TMR.

3.6.3 System Performance

The average packet transmission latencies at different SERs and different flit injection rates (FIRs) are illustrated in Fig. 3.6. The error recovery mechanism is triggered when an soft error is detected in RC, VA, or SA results, and will be finished in one or two clock cycles. The experimental results indicate that even if the SER is as high as 0.1/cycle in each router, the average packet delivery latency increase is less than 5% when compared with the error free case.

The numbers of detected errors in different functional units are illustrated in

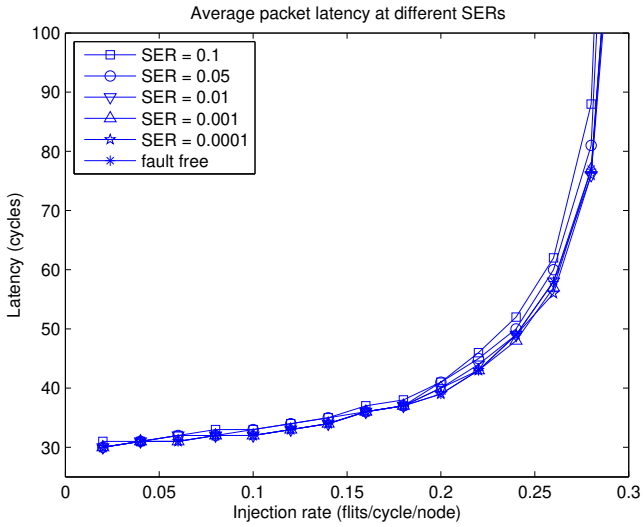


Figure 3.6: Average latency at different SERs and FIRs.

Table 3.3: Performance of RUS at different SERs

SERs	0.0001	0.001	0.01	0.05	0.1
Injected errors	31	432	4458	21966	43275
Idle RUs available	30	425	4380	21659	42626
unprotected	1	7	78	307	649

Fig. 3.7. As both RUs and VC allocators work at packet rate, they are supposed to have similar number of errors. However, in the proposed method, the situation that a packet has no available idle neighboring RU is treated in the same way as when an RC_errors happen. As a consequence, the error number in RC results is much higher than that in VA results, but they still have the same increasing trend as the SER increases, because the percentage of packets that has to wait for idle neighboring RUs depends on the traffic pattern. SA works at flit rate, thus the number of *SA_errors* is the biggest.

Table 3.3 demonstrates that when a head flit has no idle neighboring RUs available, it is necessary to treat such a situation as an RC_error happens. Otherwise, the absolute number of misdirected packets increases linearly as SER and execution time increase. Fortunately, the latency overhead induced by this strategy is marginal (see Fig. 3.6).

Figure 3.7: Detected error numbers at different SERs.

3.7 Conclusion

In this chapter, a low cost method is proposed to tolerate soft errors in the NoC router control plane. In essential we detect Routing Computation (RC) errors by comparing RC results from the local Routing Unit (RU) and idle RUs available at neighboring input ports. The RC results are recalculated in case errors are detected or neighboring RUs are not available. We detect errors in the VC Allocation (VA) and Switch Allocation (SA) results by checking if they are consistent with the correct RC results, each NoC resource is exclusively assigned to one request initiator, and each request initiator is allocated only one NoC resource. The VA/SA errors are corrected by redoing the failed procedures and retransmitting the flits. Simulations with different soft error rates on a wormhole switched 2D mesh NoC demonstrate that our method can efficiently detect and recover soft errors in RC, VA, and SA results. In the routing units, the proposed method requires 38% more silicon cost than the Σ & Branch method when XY routing algorithm is utilized, but is applicable to other routing algorithms; in the combined VA/SA units, the proposed method is simpler and more reliable than the state of the art methods. The average packet delivery latency increase is marginal when compared with the error free case.

With the method proposed in this chapter, we can ensure that packets and flits are correctly transmitted in the occurrence of soft errors in the router control plane. However, the routing path can be damaged by permanent faults and thus the packet transmission is blocked. In the next two chapters, we will propose a flit serialization method to utilize partially faulty links and a fault tolerant

routing algorithm to tolerate totally broken links as well as to efficiently utilize unpaired functional links.

Note. The contents of this chapter is based on the the following paper:

C. Chen, S. D. Cotofana, **A Low Cost Method to Tolerate Soft Errors in the NoC Router Control Plane**, Proceedings of International IEEE SoC (System-on-Chip) Conference (SOCC), 2013, pp. 374–379.

4

Effective Utilization of Partially Faulty Links

Aggressive MOS transistor size scaling substantially increase the probability of faults in NoC links due to manufacturing defects, process variations, and chip wire-out effects. Links with low fault level should be utilized rather than be discarded to avoid significant system performance degradation. However, state of the art partially faulty link utilization strategies either suffer from high area and power overheads, or significantly increase the average network latency. In this chapter, we propose a Flit Serialization (FS) method to efficiently utilize partially defected links. The FS approach divides the links into a number of equal width sections, and serializes sections of adjacent flits to transmit them on all fault-free link sections to mitigate the unbalance between the flit size and the actual link bandwidth. The proposed transmitter and receiver are transparent to the router such that their utilization is not constrained by the router architecture and implementation or network topology. Experimental results obtained on synthetic traffic and PARSEC benchmarks indicate that FS reduces the latency overhead significantly and enables graceful performance degradation when compared with related partially faulty link utilization proposals. It reduces area cost and power consumption by up to 29% and 43.1%, respectively, when compared with spare wire replacement methods, and can achieve lower area*power/saturation_throughput values than all state of the art link fault tolerant strategies. We also propose the link augmentation with one redundant section as a low cost mechanism to further increase the link dependability. Experimental results indicate that when 10% of the NoC wires are broken, adding a redundant section to each link can improve the NoC saturation throughput by 18% than just utilizing FS.

4.1 Introduction

While the aggressive transistor size shrinking improves the chip capability, it also makes the manufacturing yield and chip dependability increasingly serious concerns. Links in Networks-on-Chip (NoC) are becoming more prone to various kinds of failures caused by manufacturing defects [40], chip wear-out effects [14], or Process Parameter Variations (PPV) [98] [42]. As a single permanent fault in a link may degrade the system's performance dramatically or even render the chip useless, defective links need to be tolerated.

The most intuitive way to deal with defective links is making use of fault-tolerant adaptive routing protocols [19] [13]. Defective links are discarded and packets are forwarded along alternative fault-free paths. This ineffective utilization of link bandwidth increases packet delivery time if the detouring path is not minimal, and decreases network throughput due to the congestion around the faulty links.

For a given permanent wire fault rate, the fault level in a defective link is typically low, thus rather than completely discarding a defective link, a more effective approach is to isolate the faulty wires in the defective links and to keep on utilize the fault-free ones to transmit packets. While wires with small frequency deviation due to PPV can be dealt with by the methods described in [95] or [93], this chapter proposes a novel flit serialization method to tolerate permanent faulty wires and to utilize partially faulty links. We note that links with high fault level should be deactivated and tolerated by means of a fault tolerant routing algorithm. Our strategy to deal with heavily defected links are presented in details in Chapter 5.

In order to achieve the maximum utilization of the link bandwidth, in this chapter we propose a Flit Serialization (FS) method to efficiently utilize defective links with low fault level. The FS method divides the links and flits into a number of equal width sections, and place fault tolerant transmitters and receivers inside the output and input ports of NoC routers, respectively, to make use of all fault free link sections while mitigating the unbalance between the flit size and the actual link bandwidth. Due to this misalignment flit sections are serialized at the transmitter side to fit the narrowed link, and are deserialized at the receiver side to reconstruct integral flits. The proposed transmitter and receiver are transparent to the router such that their utilization is not constrained by router architecture and implementation or network topology. Moreover, we propose the link augmentation with one redundant section as a low cost mechanism to further increase the link dependability.

The proposed link fault-tolerant architecture is compared with equivalent state of the art solutions, i.e., the spare wire replacement method [61], the Partially Fault Link Recovery Mechanism (PFLRM) [100], and the Simple Flit Half Splitting (SFHS) method [72], in the context of a baseline NoC system. Experimental results indicate that our method reduces the latency overhead significantly and enables graceful performance degradation, when compared with related partially faulty link utilization proposals, saves the area and power overheads by up to 44% and 33%, respectively, when compared with the spare wire replacement methods, and achieves lower area*power/saturation_throughput value than all state of the art link fault tolerant strategies. Moreover, when the wire fault rate is as high as 0.1, the saturation throughput of NoC embedding the proposed strategy can be further improved by 18% when each link is augmented with an redundant link section.

The rest of this chapter is organized as follows. A brief survey of work related with partially faulty link utilization is presented in Section 4.2. Architecture and detailed implementation of the proposed transmitter and receiver are described in Section 4.3. Section 4.4 presents the simulation results and Section 4.5 concludes the presentation.

4.2 Related Work

Intuitively, we can prefabricate spare wires in the chips to replace faulty wires. Grecu et al. [40] use this method to enhance the NoC interconnect yield by mapping m interface signals to n link wires ($n \geq m$). However, their approach is only applied to the manufacturing process and cannot address runtime failures [60]. To address this drawback, Lehtonen et al. proposed a set of runtime faulty wire detection and replacing strategies. In [60], they divide the link into a certain number of sections and provides each section with one spare wire. However, this approach cannot tolerate the case in which more than one faulty wire exist in the same section. In [61], an improved method was proposed where the spare wires are shared rather than exclusively owned by each section. The spare wire replacement method can preserve the original link bandwidth, but the control logic is complicated and thus induces high silicon area cost.

A packet rebuilding/restoring algorithm is proposed by Yu et al. [114] to utilize links with reduced bandwidth. Each link is split into a big part with m bits and a small part with n bits ($m > n$). When a link is defected, the fault free wires in the small part are utilized to replace the faulty wires in the big part.

Accordingly, a packet first transmits the most significant m bits of each flit. The non-transmitted small parts of the flits are reassembled into one or more m-bit flits and then transmitted. When more than n wires are broken, the link is abandoned. This method can be seen as implementation of the spare wire replacement method without using prefabricated spare wires. However, because an integral flit can only be restored after all the reassembled flits have been received, this method cannot be utilized in low latency routers with wormhole or Virtual-Cut-Through (VCT) switching technology.

By noticing that the faults are rarely clustered when they randomly happen, Vitkovskiy et al. [100] introduced a Partially Faulty Link Recovery Mechanism (PFLRM), which is mainly comprised of a flit shifter, a de-shifter, and a flit re-assembler. Assuming the maximum fault cluster size is k in a link, it requires $k + 1$ cycles to successfully transmit a flit. In the first cycle, the flit is transmitted in the normal way. In each of the rest cycles, the flit is rotated by 1-bit before being transmitted, thus the data bits that were transmitted on faulty wires in the previous cycle can be transmitted on fault free ones. At the receiver side, the flits are de-rotated and the newly transmitted data bits are selected to reassemble the original flit. Although this approach can theoretically work for defective links with an arbitrary large number of faulty wires, the induced transmission latency overhead can be significantly high. We note that even only a single faulty wire exists in the link, two cycles are needed to transmit a flit successfully.

The aforementioned strategies rely on the exact knowledge of each link wire status, which may induce high burden to the Built-In-Self-Test (BIST) mechanism. Conversely, Palesi et al. [72] and Lehtonen et al. [60] proposed the method of using flit splitting to tolerate faulty wires. In this approach, a link is divided into four sections. The fault-free sections are utilized to transmit flits and the ones containing broken wires are abandoned. Thus the link status is diagnosed at section level, which reduces the BIST delay overhead and complexity. Note that the link can be divided into more sections to tolerate more broken wires. However, their approaches cannot utilize all fault free link sections. For example, when one of the four link sections is broken, only two functional sections are utilized to transmit flits half by half. In the remainder of this chapter, we name this method as Simple Flit Half Splitting (SFHS).

In summary, spare wires can preserve the NoC performance but introduce a high silicon overhead, while SFHS and PFLRM have low area overhead but induce high extra latency. By comparison, our FS method significantly reduces the latency, when compared with SFHS and PFLRM, while maintaining

a more reasonable silicon cost, when compared with the spare wire replacement methods.

4.3 Partially Faulty Link Utilization

The principle of the Flit Serialization (FS) strategy is to divide the links and flits into k equal width sections, and make use of all functional link sections to do data transmission. We assume that k is a power of 2 for the sake of control logic simplicity.

Fig. 4.1 depicts the proposed fault tolerant link architecture. For each unidirectional link, we use a Test Data Generator (TDG) at the Transmitter (TX) side and a Test Error Detector (TED) at the Receiver (RX) side to diagnose the link status and generate a k-bit fault vector to indicate the faulty link sections. If faulty wires exist in at least one link section, sections of adjacent flits are serialized by the flit serialization unit at the TX side and then transmitted on the fault free link sections. All flit sections are then deserialized at the RX side to reconstruct the original flits. On fault free links, the flits are transmitted according to the normal protocol, bypassing the proposed flit serialization and deserialization units. The FS mechanism is transparent to the rest of the router parts thus its utilization is not constrained by the router architecture and implementation, or by the network topology.

As the number of control signals, e.g., the data valid signal and the credit control signals, in each link is much smaller than the number of data lines, they are protected by Triple Modular Redundancy (TMR) method with a marginal silicon area overhead. If a Error Correcting Code (ECC) is utlized to protect data from transient errors, the error coding logic should be placed before the flit serialization unit and the error decoding logic should be placed after the flit deserialization unit, thus soft errors generated in the data link as well as inside the transmitter and/or the receiver can be detected and corrected.

4.3.1 Link Diagnosis

Unlike spare wire replacement and PFLRM, which need to know the precise status of each wire, our method just needs link fault vectors at the section level, i.e., a link section is broken if it contains broken wires. For example, if the third section of a link (with 4 sections in total) contains faulty wires, the fault vector of the link is marked as “1101”. Thus for an n-bit wide link

divided into k sections, we just need a k-bit wide register to store the section level fault vector.

In this chapter, we assume that the possible permanent faults are stuck-at, i.e., the wire value is stuck at '1' or '0', and crosstalk, i.e., the status of two adjacent wires interfere with each other and one is in the dominant position and determines the value of the other one. Under these fault models, to detect the faults in a 4-bit wide link section, the TDG sequentially injects 2 test vectors "0101" and "1010" to the link section under testing. At the RX side, received data are XORed with expected values and if the result is not always "0000", an error signal is asserted to indicate that faults exist in the link section. As a comparison, to achieve bit level fault vector, the link diagnosis method in [100] uses the same test vectors but can only detect stuck at faults, the method in [41] can detect crosstalk faults but requires 8 test vectors and thus 8 clock cycles to diagnose 1 wire, and the method in [60] can diagnose wires status without ceasing data transmission but requires complicated control logic. We note that to distinguish permanent errors from soft ones, the test is repeated 3 times and the link section is marked as broken only when the error signal is asserted at least twice.

Link diagnosis is triggered periodically and when the number of soft errors detected by the ECC logic exceeds a predefined threshold in a short time period [60]. Because intermittent errors may have the same syndrome as permanent errors when they happen, sections which are marked as faulty in the previous test are also tested, to prevent situations when vanished intermittent errors are still disabling sections. At the end of the diagnosis process the achieved fault vector is sent to the transmitter via a TMR protected serial wire.

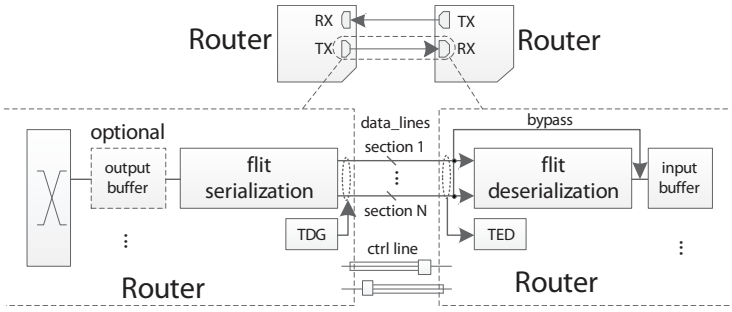


Figure 4.1: Proposed fault-tolerant link architecture.

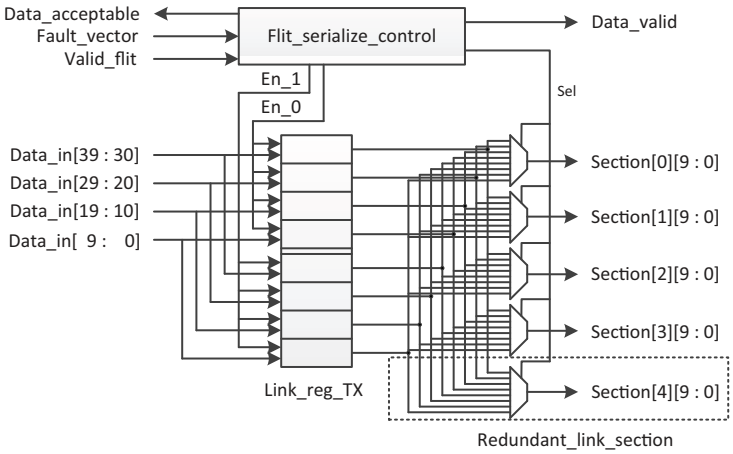


Figure 4.2: Flit serialization unit - TX.

4.3.2 Flit Serialization and Deserialization

In a typical NoC router, the head flit of each packet has to sequentially go through 3 pipeline stages: Routing Computation (RC), combined Virtual Channel (VC) Allocation (VA) and Switch Allocation (SA), and Crossbar Transversal (CT). An extra Link Transversal (LT) stage is usually required to send flits to the downstream router. The proposed flit serialization and deserialization units are implemented in the LT stage. For the sake of simplicity, we present our proposal for the case when both flits and links are divided into 4 sections, and the link width is 40-bit, i.e., each link section is 10-bit wide. We note that the proposed principle is more general and can be applied to wider links with more sections.

Fig. 4.2 illustrates the structure of the proposed flit serialization unit. In general, each output port embeds a 1-flit width link register to store flits before they are sent to the downstream router. To allow for flit serialization, we expand the link register width to 2-flits and divide it into 8 sections that can be read and write independently. The register is designed in such a way that a new flit can be registered in the Least Significant Half (LSH) of the register if there are flit sections in its Most Significant Half (MSH) still waiting to be transmitted, and vice versa. If the link is fault-free, only the register MSH is utilized, acting as a conventional link register. Otherwise, flits are serialized under the control of the *flit_serialize_ctrl* unit. The serialization process is presented in more detail in Section 4.3.3. The number of flit sections that can be transmitted in each cycle is the same as that of fault free link sections. Multiplexers are

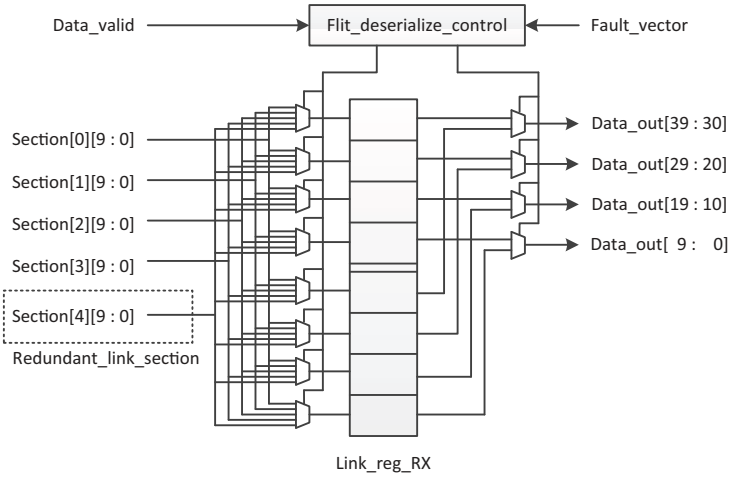


Figure 4.3: Flit deserialization unit - RX.

utilized to select the to be transmitted flit sections. The *Data_valid* signal indicates the downstream receiver whether valid data are transmitted on the link in each cycle. With a narrowed link, the flit is transmitted on the link at a lower rate than on the router crossbar and we rely on the *data_acceptable* signal to indicate if the next flit can be accepted by the serialization unit subject to the remained buffer space.

At the RX side, a flit deserialization unit (see Fig. 4.3) reconstructs the flit out of the serialized sections. Similar with the flit serialization unit, a 2-flit wide link register (*link_reg_RX*) is employed. Multiplexers are utilized to select the valid sections from the link under the control of the *flit_deserialize_ctrl* unit. The newly received flit sections are stored into the correct link register position to reassemble integral flits. When the link register MSH or LSH is full, one flit was assembled and can be read out by the router.

4.3.3 Flit Transmission Process

Fig. 4.4 graphically depicts the timing diagrams capturing the flit transmission process specific to our method. When the link is fault-free, a flit from the crossbar is loaded into the *link_reg_TX* MSH and then directly transmitted to the downstream router input buffers (see Fig. 4.4(a)). At the RX side, the flit deserialization unit is bypassed.

We introduce the FS method with an example situation when one of the 4 link

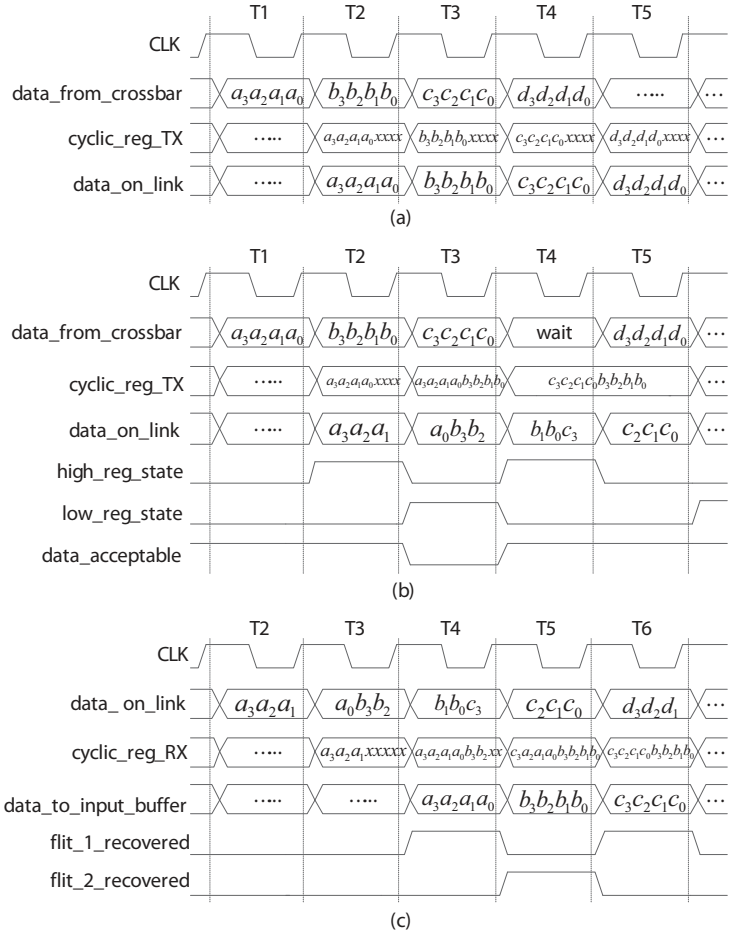


Figure 4.4: Timing diagram of proposed mechanism (a) Timing diagram for a fault-free link; (b) Transmitter side when one section contains faulty wires; (c) Receiver side when one section contains faulty wires.

sections is affected by faults. As illustrated in Fig. 4.4(b), at TX side, flit a floats at the output port of the crossbar at the rising edge of $T1$ and is written into the *link_reg_TX* MSH at the rising edge of $T2$. During the $T2$ cycle, the first three sections of the flit a (a_3, a_2, a_1) are transmitted to the downstream router via the three fault-free sections of the link. At the rising edge of $T3$, flit b is written into the LSH of *link_reg_TX*. Flit sections a_0, b_3, and b_2 are transmitted in the same cycle. The signal *data_acceptable* is set to '0' in $T3$ such that no new flit may appear at the crossbar output port in $T4$. A wait cycle is inserted to allow for the transmission of the last three sections of flit c

during $T5$. The signals *high_reg_state* and *low_reg_state* are utilized to indicate the status of *link_reg_TX* MSH and LSH, respectively. Each signal is asserted once a flit is written into the corresponding register part, and de-asserted in the clock cycle when all data belonging to the flit are read out.

At the receiver side (see Fig. 4.4(c)), flit sections are de-serialized and re-assembled into integral flits in *link_reg_RX*. Valid flit sections are selected by input side demultiplexers and written at the correct positions in *link_reg_RX*. Once the register MSH or LSH is full, an integral flit is recovered. The signals *flit_1_recovered* and *flit_2_recovered* indicate the availability of recovered flits and control the output side multiplexers to select the corresponding register sections.

4.3.4 Redundant Link Section

Even if we can efficiently utilize the remained link bandwidth, the flit transmission latency is increased when faults exist. As illustrated in Fig 4.2 and Fig. 4.3, by extending each link with one redundant section, we can combine the benefits of the spare wire replacement method and the FS method. If only one fault exists, or multiple faults exist but "luckily" they are all resident in the same link section, the link can still transmit one integral flit per cycle. We note that such scenario is a "disaster" for the PFLRM method as it may cause large fault cluster size and hence long flit transmission latency.

We do not rely on bit level spare wire replacement methods, e.g., [60, 61], because overlapping FS with these methods requires: (i) an extra multiplexing and demultiplexing step, which can significantly increase the link critical path length, and (ii) additional registers to store the selection bits, which results in significant area overhead and power consumption. Given that FS extra area is dominated by the 2-flit wide link registers, especially for wide links, and that adding one redundant link section does not require larger link registers the link augmentation with a redundant section is a cost effective solution. As illustrated in Section 4.4, by adding one redundant section per link we can achieve up to 18% saturation throughput improvement when the link fault rate is as high as 0.1, with only 4.7% area and 2.4% power overhead, respectively.

4.3.5 Link Latency and Reliability

The flit transmission latency when the FS method is utilized (l_{FS}) to continuously transmit a number of flits (*flit_number*) can be expressed as:

$$l_{FS} = \left\lceil \frac{section\_number \times flit\_number}{fault\_free\_section\_number} \right\rceil, \tag{4.1}$$

where *section_number* is the number of sections in the link. For example, transmitting 10 flits via a link which has one broken section requires 14 and 12 cycles when the link is divided into 4 and 8 sections, respectively.

For the sake of comparison, PFLRM (l_{PFLRM}) and SFHS (l_{SFHS}) flit transmission latencies are expressed in (4.2) and (4.3), respectively.

$$l_{PFLRM} = (cluster\_size + 1) \times flit\_number, \tag{4.2}$$

where *cluster_size* is the maximum fault cluster size.

$$l_{SFHS} = \frac{section\_number}{available\_section\_number} \times flit\_number. \tag{4.3}$$

Note that the *available_section_number* is not the number of fault free sections. It can be equal with or less than the number of sections in the link and can have the value of 2^i, $i = 0, 1, 2, ...$ For example, when the link is divided into 4 sections, it can be 1, 2, or 4.

Table 4.1 presents the average flit transmission latency (cycles/flit) when FS, PFLRM, and SFHS are utilized to continuously transmit flits via a defective link. The number of faults in the first table row indicates the fault cluster sizes for PFLRM, and the numbers of faulty sections for FS and SFHS. S4 and S8 mean that the link is divided into 4 and 8 sections, respectively. From the Table we observe that PFLRM and SFHS latencies double in the presence of one error while for FS this happens only after half of the link sections are broken.

In Fig. 4.5, we depict the average flit transmission latency on 40-bit wide links when fault wires are uniformly distributed and each link is divided into 8 sections. The results are obtained by doing Monte Carlo simulations when the following methods are applied: FS, FS with one redundant link section (FS+1), PFLRM, and SFHS. Note that the links with no fault free section are not considered. We can observe that from the statistic point of view, FS provides lower flit transmission latency than PFLRM when the link has less than 6 faulty wires. When more faulty wires exist, FS performs worse than PFLRM

but still better than SFHS. We can also observe that FS+1 achieves lowest flit transmission latency when there are less than 9 faulty wires.

When 9 or more faulty wires are uniformly distributed in the links, PFLRM outperforms all the other counterparts. However, in the cases corresponding to large physical defects multiple, e.g., k, adjacent wires may get faulty. In such a scenario, PFLRM requires $k+1$ cycles to successfully transmit a flit, which results in a large latency overhead, and a spare wire replacement method has to make use of k spare wires, which results in a large area overhead. Given that such a large defect will most likely affect only one or two link sections the proposed FS approach can handle such extreme cases with an efficiency corresponding to the case when one or two faulty wires are detected in the link.

In principle, we can split a link into more sections, e.g., 16 or more, to achieve more graceful performance degradation. However, this implies that more and larger multiplexers are required, which have a negative impact on area and power overheads. If we assume that each wire has the same probability (p_e) to be permanently faulty, the probability that an n-bit wide link has k faulty wires can be calculated using (4.4).

$$P_k = \binom{n}{k} {p_e}^k (1-p_e)^{n-k} \tag{4.4}$$

Thus even if p_e is as high as 0.001, the probabilities for a 40-bit wide link to have 4 and 8 faulty wires are only 8.8×10^{-8} and 7.4×10^{-17}, respectively. This means that by dividing the link into 8 sections, we can ensure that the link probability to have a flit transmission latency of 2 cycles is lower than 10^{-8} and the probability for a link to be totally broken is lower than 10^{-16}. Given that faulty wires are not always evenly distributed in different sections,

Table 4.1: Average flit transmission latency (cycles/flit) when flits are transmitted continuously

number of faults		0	1	2	3	4	5	6	7	8
FS	4 sections	1	1.33	2	4	–	–	–	–	–
	8 sections	1	1.14	1.33	1.60	2	2.67	4	8	–
PFLRM		1	2	3	4	5	6	7	8	9
SFHS	4 sections	1	2	2	4	–	–	–	–	–
	8 sections	1	2	2	2	2	4	4	8	–

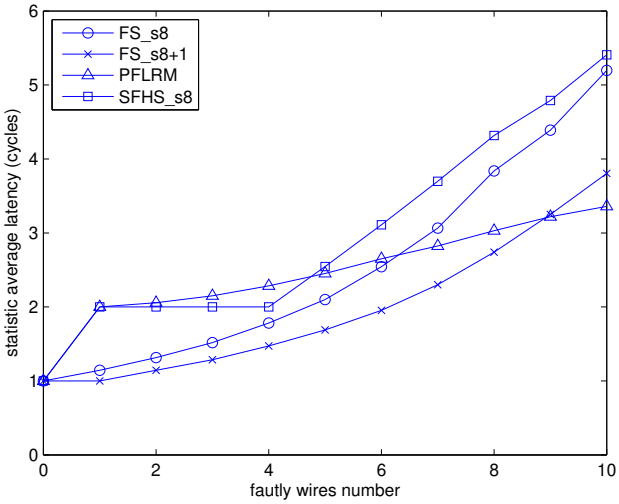


Figure 4.5: Average flit transmission latency of different partially faulty link utilization strategies. The link is divided into 8 sections for FS and SHFS.

the aforementioned two probabilities are much lower in practice. In view of this analysis, we conclude that dividing links into more than 8 sections has no practical relevance for most state of the art NoCs whose flit width is 32-bit [87], and the section number should be a power of 2, e.g., 4 or 8, to achieve simple control logic. The actual number of sections can be determined via a trade-off process which takes into consideration the targeted fault-tolerance capability and the available silicon real estate.

4.4 Evaluation

To put the implications of our link fault-tolerant architecture in a better practical prospective, we evaluate and compare it with other three tightly related proposals presented in [61], [100], and [72], namely spare wire replacement, PFLRM, and SFHS, respectively. To this end, we implemented all these four link fault-tolerant methods at RTL level by using Verilog HDL, and applied them in the context of an 8×8 2D mesh NoC.

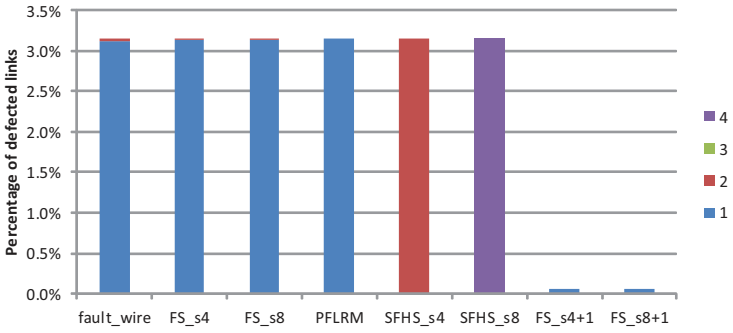
Each baseline router has 3 pipeline stages, i.e., RC, VA/SA, and CT, and 5 Physical Channels (PC). Each PC is shared by 4 VCs, and each VC buffer is 4-flit deep and 40-bit wide, as both of flit and link widths are 40-bit. The router

and the link fault-tolerant modules are synthesized using the Synopsys Design Compiler with TSMC 65-nm standard cell as target technology.

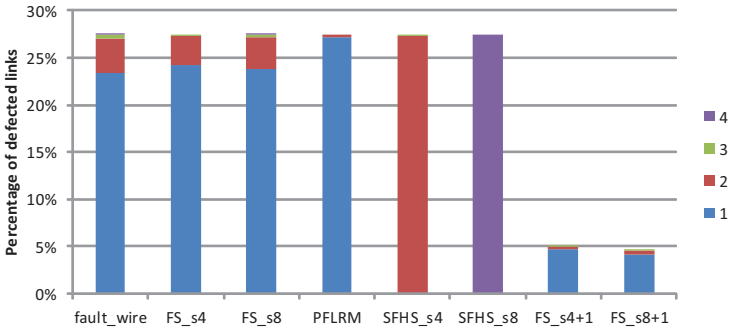
4.4.1 FS Performance on Synthetic Traffic

To evaluate the performance of the FS method, we first run synthetic uniform random traffic in the context of different fault link patterns for a wide range of permanent wire fault rates, i.e., 0.001, 0.01, 0.05, and 0.1. We assume that each wire has the same fault rate p_e and faults are uniformly distributed across the links. The three partially faulty link utilization strategies, i.e., FS, PFLRM, and SFHS, are applied to the NoC system and simulated for each fault pattern. Note that when spare wire replacement method is employed, the original NoC performance is preserved until all spare wires are utilized to replace the broken wires. For the section based strategies, i.e., FS and SFHS, we simulated two cases when each link is divided into 4 (FS_s4 and SFHS_s4) and 8 (FS_s8 and SFHS_s8) link sections, respectively. For the FS method we also simulated the case when each link is augmented with one redundant link section, i.e., FS_s4+1 and FS_s8+1. Each packet consists of 4 flits and is routed with the XY routing protocol.

We first run Mento Carlo Simulations to study the fault distribution at different wire fault rates. We randomly create faulty wires in the NoC and count the number of defected links. For each defected link, we count the number of faulty wires, the maximum fault cluster size, and the number of broken link sections. The fault distribution is illustrated in Fig. 4.6 and 4.7. In the figures, each column's height represents the percentage of defective links present in the NoC. In each column, different colors represent the percentage of defective links with different fault levels. For example, the columns' red parts denote the percentage of links with 2 faulty wires, or links with 2 unusable link sections in FS and SFHS, or links with a fault cluster size of 2 in PFLRM. Take Fig. 4.6(b) for example, it illustrates that when p_e is 0.01, 27.4% of the NoC links are defected, among which the percentage of links contain 1, 2, and 3 faulty wires are 23.4%, 3.7%, and 3%, respectively. This also means that the percentage of links contain 1, 2, and 3 broken sections are 24.2%, 3.0%, and 0.2%, respectively, for FS_s4 and SFHS_s4, and 23.7%, 3.4%, and 0.3%, respectively, for FS_s8 and SFHS_s8, and the percentage of links have a fault cluster size of 1 and 2 are 27.1% and 0.3%, respectively. For FS with one redundant section per link, the link bandwidth is reduced only when 2 or more sections are broken, thus for FS_s4+1 and FS_s8+1 the percentage of links with reduced bandwidth is much lower than in the other cases. Note that SFHS has



(a) fault pattern when $p_e = 0.001$;

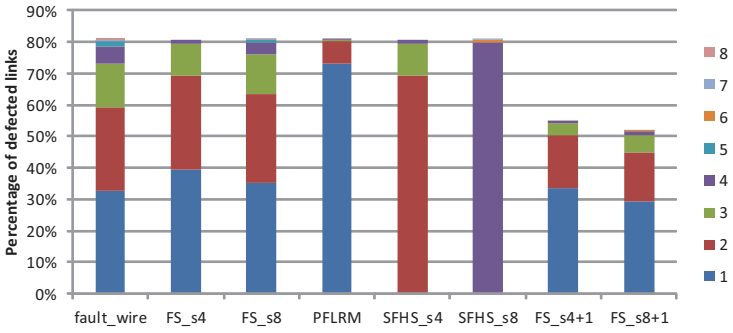


(b) fault pattern when $p_e = 0.01$

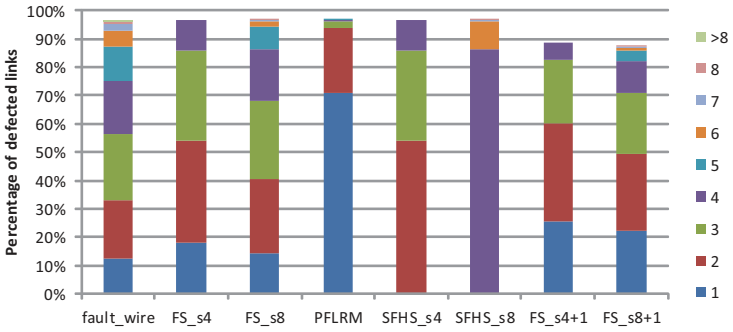
Figure 4.6: Fault link Patterns at different wire fault rate.

much lower link bandwidth utilization efficiency than FS, e.g., even if a link has only 1 broken section, SFHS_s4 and SFHS_s8 treat it equivalently as 2 and 4 sections are broken, respectively. Such SHFS property is reflected in Fig. 4.6 and 4.7 by rounding up the number of broken link sections to its equivalent case.

Fig. 4.8 and 4.9 depict the NoC performance measured in terms of average packet transmission latency obtained when different partially faulty link utilization strategies are applied. The packet transmission latency is counted since the packet is generated in the source node till the tail flit is received by the destination node, i.e., the queuing time in the source node is included. We gradually increase the Flit Injection Rate (FIR) at a step length of 0.01 flits/cycle/node to derive the near zero traffic load packet transmission latency and the saturation throughput, i.e., the FIR when the packet latency approaches infinity.



(a) fault pattern when $p_e = 0.05$



(b) fault pattern when $p_e = 0.1$

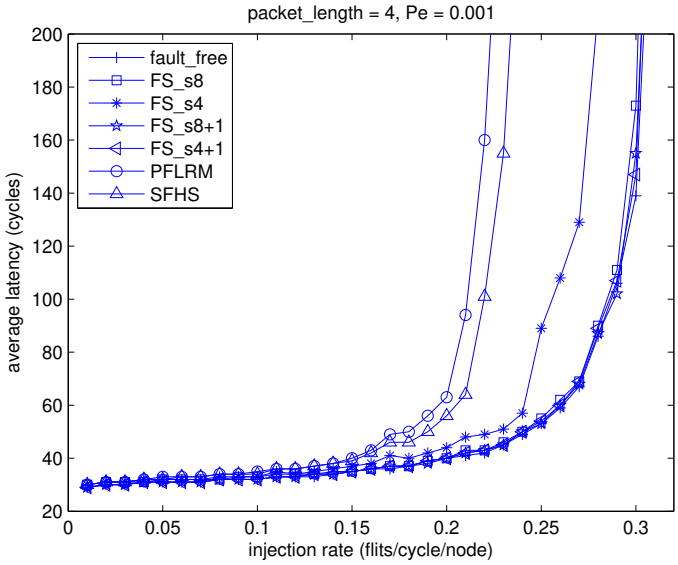
Figure 4.7: Continue – Fault link Patterns at different wire fault rate.

Without Redundant Link Section

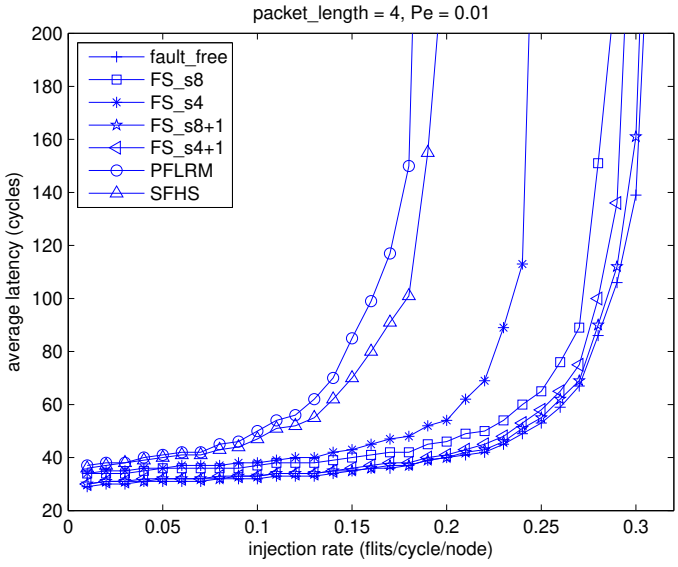
We first compare the performance of the three partially faulty link utilization strategies when the redundant link section is not provided.

As indicated in Table 4.1, FS_s8 induces the lowest flit transmission latency overhead on defective links with 1 broken section, and thus it achieves the best performance when p_e is 0.001, i.e., the average packet transmission latency is very close to the fault-free case (see Fig. 4.8(a)). At such fault rate, the performance of FS_s4 is lower than that of FS_s8 but still much better than that of PFLRM and SFHS. This can be explained by the fact that in links with one broken section, both PFLRM and SFHS will double the flit transmission latency at least, while FS_s4 can keep the latency overheads as low as 33.3%. Note that for both SFHS_s8 and SFHS_s4, the flit transmission latency on the defective links is doubled at this fault rate thus they have the same performance.

As p_e increases, more links contain faults and the average number of faulty

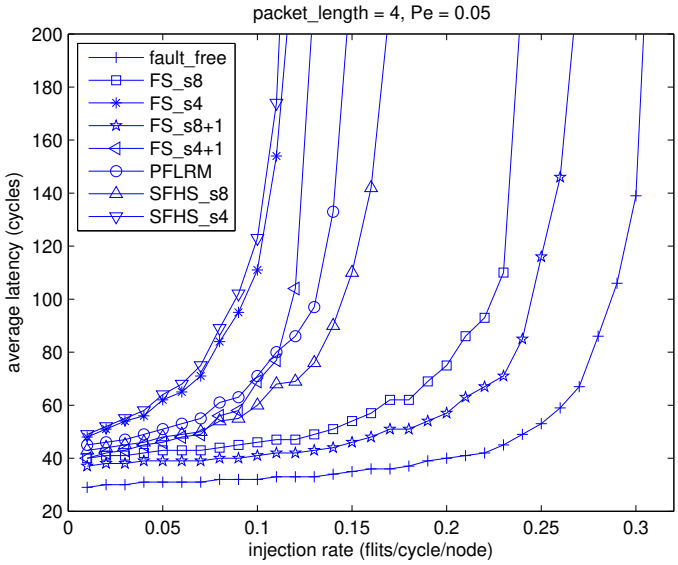


(a) performance when $p_e = 0.001$;

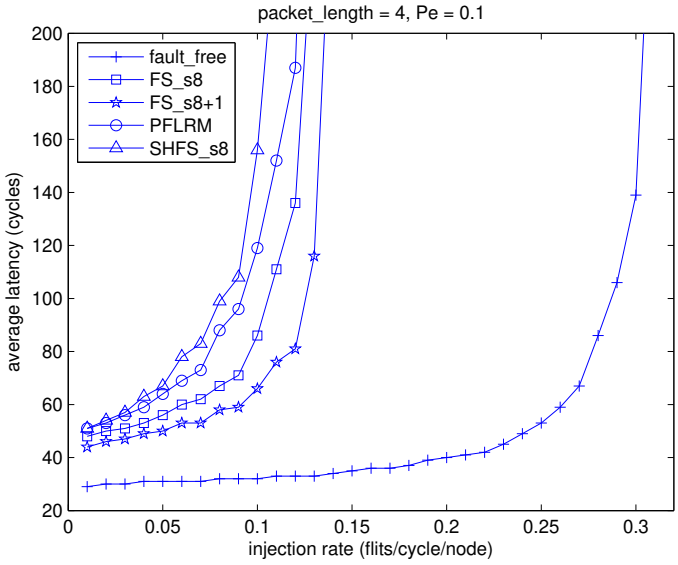


(b) performance when $p_e = 0.01$;

Figure 4.8: NoC Performance at different wire fault rate.



(a) performance when $p_e = 0.05$;



(b) performance when $p_e = 0.1$.

Figure 4.9: Continue – NoC Performance at different wire fault rate.

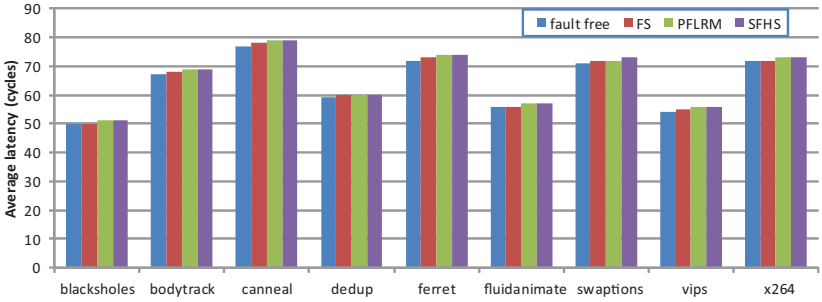
wires becomes larger, leading to more unusable sections and bigger fault cluster size in links. The average flit transmission latency increases for all partially faulty link utilization strategies. But when the fault rate is not very high, e.g., $p_e = 0.01$, FS still outperforms PFLRM and SFHS (see Fig. 4.8(b)).

When p_e further increases to 0.05, FS_s8 still achieves the best performance because the flit transmission latency on more than 99% of the defective links is less than 2 cycles. However, FS_s4 performs worse than SFHS_s8 and PFLRM. This is because the number of links that have a flit transmission latency higher than 2 cycles in FS_s4 is much larger than that in SFHF_s8 and PFLRM. These slow links cause severe congestion in their upstream routers and hence obvious system performance degradation. We note that at such p_e value, totally broken links can exist in FS_s4 and SFHS_s4. To avoid the implication of FTRAs to the performance of partially faulty link utilization methods, the fault patterns which contain totally broken links are not considered in this subsection. This cannot fundamentally affect the results because only 1 or 2 such links may exist in the NoC at this fault rate.

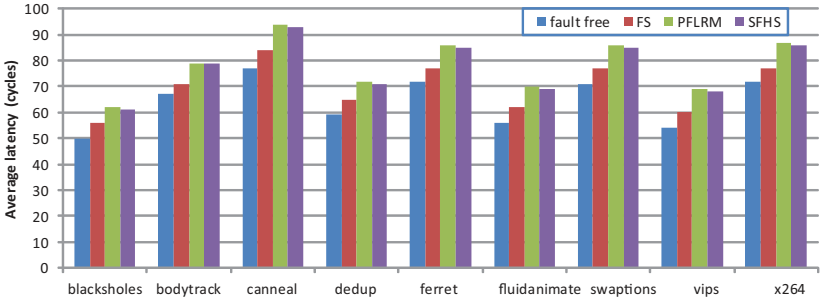
When the permanent wire fault rate is as high as 0.1, 96.6% of the links are defective and the average fault level is high, as depicted in Fig. 4.7(b). Under this extreme conditions, FS_s8 exhibits only slightly better performance than PFLRM (see Fig. 4.9(b)) where FS_s4 and SFHS_s4 have so many totally broken links that they are not considered. If the fault rate keeps on increasing, the average packet transmission latency in FS_s8 increase and eventually its performance gets worse than that of PFLRM.

With One Redundant Link Section

As illustrated in Fig. 4.6 and 4.7, when each link is augmented with one redundant link section, the numbers of defective links when p_e is 0.001, 0.01, 0.05, and 0.1 are reduced by 98%, 82%, 32%, and 8%, respectively. The system performance, in terms of average packet transmission latency and saturation throughput, is also obviously improved. For example, when p_e is up to 0.01, FS_s4+1 and FS_s8+1 can still provide similar performance with the fault free case, while FS_s4 and FS_s8 induce 21% and 3% saturation throughput degradation already. When $p_e = 0.05$, one redundant link section can improve the FS_s4 and FS_s8 saturation throughput by 20% and 8.7%, respectively. Although the number of defective links is only reduced by 8% when $p_e = 0.1$, the FS_s8 saturation throughput is improved by 18%. As the wire fault rate increase from 0 to 0.1, FS_s8+1 provides the most gracefully system performance degradation when compared with other counterpart partially faulty link



(a) Average latency when $p_e = 0.001$;



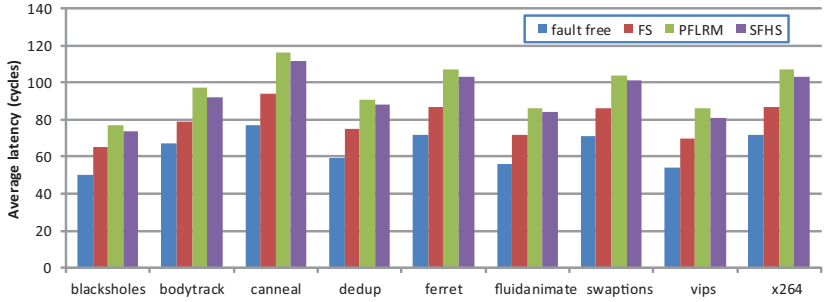
(b) Average latency when $p_e = 0.01$;

Figure 4.10: Average packet transmission latencies of PARSEC Benchmarks at different fault rates. Links are divided into 8 sections for FS and SFHS.

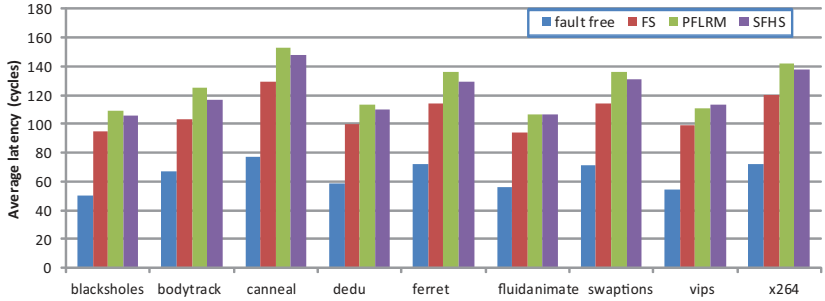
utilization strategies.

4.4.2 FS Performance on PARSEC Benchmarks

In this subsection, we evaluate our proposal with PARSEC benchmarks [9] traffic traces recorded with the Netrace [43] tool on the M5 full system simulator [10]. We replay the benchmark traces and inject the packets into our NoC platform according to the packet time flag while maintain the packets dependencies. When compared with the full system simulation, simulation with recorded traffic can better reflect the performance of the NoC system [24] as the performance fluctuations caused by the interaction between the cores and the NoC are removed. The packet length can be 4-flits and 20-flits according to the packet type. The transmission delay of each packet is counted since the packet is read out from the record in the source node till the tail flit is received by the destination node. The simulation results are illustrated in Fig. 4.10 and Fig. 4.11. We note that the links are divided into 8 sections for FS and SFHS.



(a) Average latency when $p_e = 0.05$;



(b) Average latency when $p_e = 0.1$.

Figure 4.11: Average packet transmission latencies of PARSEC Benchmarks at different fault rates. Links are divided into 8 sections for FS and SFHS.

We can observe that for all the three partially faulty link utilization strategies, the average packet transmission latency increases as the wire fault rate becomes higher. When the wire fault rate is quite low, i.e., $p_e = 0.001$, only several defected links with low fault level exist in the NoC. Given that benchmarks' FIRs are much lower than the saturation FIR there is no obvious difference between the 3 partially faulty link utilization approaches. As the p_e increases, the advantage of our proposal becomes obvious. For example, the FS packet transmission latency is on average 13% and 12% lower than that of PFLRM and SFHS, respectively, when $p_e = 0.01$, but the FS advantage increases to 28% and 22% latency reduction, respectively, when $p_e = 0.1$.

4.4.3 Area and Power

The area and power overheads of the four different link fault-tolerant methods, i.e., FS, PFLRM, SFHS, and spare wire replacement, are presented in Table. 4.2. Our proposal and SFHS are evaluated with two versions containing

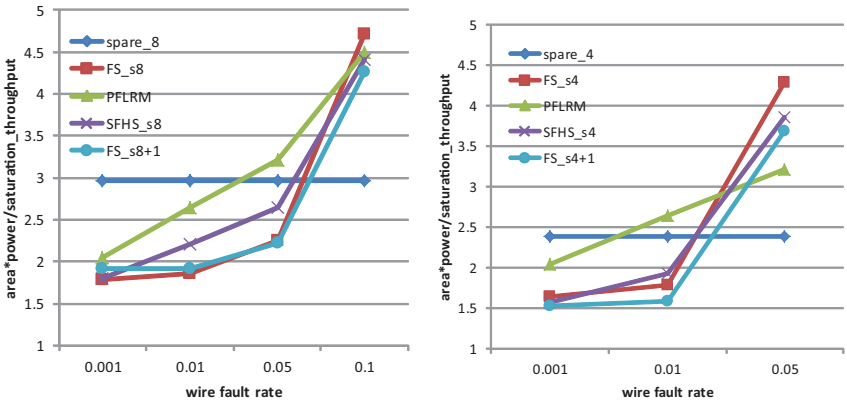
Table 4.2: Power and area overhead of different link fault-tolerant methods

		Area (μm^2)	Power (mW)
Basic router		64813 / 1.00	25.14 / 1.00
Spare wire	8 wire	55942 / 1.86	39727 / 1.61
	4 wire	14.85 / 1.59	12.06 / 1.48
FS	S8	27413 / 1.42	6.49 / 1.26
	S4	15812 / 1.24	4.76 / 1.19
PFLRM		15363 / 1.24	6.17 / 1.25
SFHS	S8	14407 / 1.22	3.02 / 1.12
	S4	7350 / 1.11	1.90 / 1.7.6
FS+1	S8	30827 / 1.48	7.23 / 1.29
	S4	17757 / 1.27	5.08 / 1.20

4 (s4) and 8 (s8) link sections, and the spare wire replacement method is evaluated with two versions containing 4, and 8 spare wires. From the Table we can observe that, the FS area and power overheads are lower than the ones of spare wire replacement, but higher than the ones of PFLRM and SFHS. For example, when compared with the baseline router, the FS_s8 area overhead is 42%, while those of 8 spare wires, PFLRM, and SFHS_s8 are 86%, 24%, and 22%, respectively; the FS_s8 power overhead is 26%, while those of 8 spare wires, PFLRM, and SFHS_s8 are 59%, 25%, and 12%, respectively. The FS_s4 area and power overheads also falls between those of 4 spare wires and SFHS_s4. It is worth to note that FS_s4 requires similar silicon area cost and less power consumption than PFLRM but provides better system performance when the wire broken rate is less than 0.01.

It is illustrated in Table 4.2 that adding one redundant link section to each link in the context of the FS method (FS+1) increases the area and power consumption by 6% and 3%, respectively, in the S8 case, and 3% and 1%, respectively, in the S4 case. We note that the number of wires is increased by 12.5% and 25% for the S8 and S4 cases, respectively.

To give a comprehensive overview on the implementation cost and performance of different link fault tolerant strategies, we compute their Area*Power/Saturation_throughput (AP/S) metric value and illustrate the normalized results, to that of the baseline router, in Fig. 4.12. A strategy that



(a) With 8 spare wires or sections; **(b)** With 4 spare wires or sections;

Figure 4.12: Normalized value of area*power/saturation_throughput metric of different link fault tolerant strategies. Lower is better.

can achieve a low AP/S value, i.e., high saturation throughput and low area and power cost, is preferred. We can observe that when the wire fault rate (p_e) is as low as 0.001, FS, FS+1, and SFHS achieve the similar AP/S value, which is lower than that of PFLRM and spare wire replacement. When p_e is 0.01 and 0.05, FS_s8 and FS_s8+1 always achieve lower AP/S value than SFHS, PFLRM, and the spare wire replacement method. When p_e is as high as 0.1, which is unlikely to happen in practice, all partially faulty link utilization strategies induce high saturation throughput reduction and the spare wire replacement method becomes the most effective link fault tolerant strategy. Thus in practice FS is an effective method to tolerate faulty link wires and to utilize remained link bandwidth.

4.5 Conclusion

In this chapter, we proposed a Flit Serialization (FS) method to efficiently utilize partially defected links. The FS approach divides the links into a number of equal width sections, and serializes sections of adjacent flits to transmit them on all fault-free link sections to mitigate the unbalance between the flit size and the actual link bandwidth. Experimental results obtained on synthetic traffic and PARSEC benchmarks indicate that FS reduces the latency overhead significantly and enables graceful performance degradation when compared with related partially faulty link utilization proposals. It reduces

area cost and power consumption by up to 29% and 43.1%, respectively, when compared with spare wire replacement methods, and can achieve lower area*power/saturation_throughput values than all state of the art link fault tolerant strategies. We also propose the link augmentation with one redundant section as a low cost mechanism to further increase the link dependability. Experimental results indicate that when 10% of the NoC wires are broken, adding a redundant section to each link can improve the NoC saturation throughput by 18% than just utilizing FS.

While utilizing partially faulty links with low fault levels by means of the FS method can preserve the NoC link bandwidth and thus reduce the NoC performance degradation speed, utilizing Heavily Defected (HD) links may cause severe congestion in their upstream routers. In the next chapter, we will discuss when HD links should be perceived as broken, and propose a fault tolerate routing algorithm to tolerate deactivated links while make use of the unpaired functional link in partially broken interconnects.

Note. The contents of this chapter is based on the the following papers:

C. Chen, Y. Lu, and S. D. Cotofana, **A Novel Flit Serialization Strategy to Utilize Partially Faulty Links in Networks-on-Chip**, Proc. IEEE/ACM International Symposium on Networks on Chip (NoCS), pp.124-131, May 2012.

C. Chen, Y. Fu, and S. D. Cotofana, **Toward Maximum Utilization of Remained Bandwidth in Defected NoC Links**, submitted.

5

Heavily Defected Link Deactivation and Fault Tolerant Routing

While defected links with low fault levels should be utilized to preserve the NoC link bandwidth, Heavily Defected (HD) links should be deactivated and dealt with by means of a Fault Tolerant Routing Algorithm (FTRA) to avoid severe congestion. In this chapter, we discuss the optimal threshold to deactivate HD links and propose a FTRA to tolerate the deactivated links as well as to efficiently utilize Unpaired Functional (UPF) links in partially broken bidirectional interconnects. The optimal link deactivation threshold is determined by comparing the zero load packet transmission latency on the HD links and that on the shortest alternative path. The basic fault pattern tolerated by the proposed UPF link aware FTRA (UPF-FTRA) is a fault wall, which is composed of adjacent broken links with the same outgoing direction. Messages are routed around the fault walls along the misrouting-contours of the broken links. Our proposal is evaluated with both synthetic traffic and PARSEC benchmarks. Experimental results indicated that UPF-FTRA can improve the NoC saturation throughput by up to 22% when compared with state of the art counterparts. Synthesis results with TSMC 65nm technology indicate that, embedding UPF-FTRA into a baseline router increases the area and power overhead by 9% and 2% respectively, which is similar with that of the conventional solid fault region based algorithms. Simulation results we obtained at various wire broken rate configurations indicate that we achieve the highest saturation throughput if 4- or 8-section links with a flit transmission latency longer than 4 cycles are deactivated.

5.1 Introduction

According to the portion of faulty wires to the link bandwidth, the defected links may have different fault levels. To achieve the maximum utilization of remained link bandwidth, the links with different fault levels should be treated with different strategies. For a partially defected link with low fault level, only a small number of wires in it are broken. Rather than treat such links as totally broken, it is more beneficial to keep on utilizing them with Partially Faulty Link Utilization Methods (PFLUM), e.g., the Flit Serialization (FS) method proposed in the previous chapter, to minimize the system performance degradation. For links with high fault levels or are already totally broken, we have to make use of Fault Tolerant Routing Algorithms (FTRAs) to route packets along alternate paths. We note that routers can also be partially defected and still be utilized [54]. However, in most cases, the defected routers or router ports can be considered as equivalent with the situation when the links incident to them are broken.

Whether to utilize a defective link can be decided (i) dynamically based on the local traffic load or (ii) statically by checking if its fault level has exceeded the link deactivation threshold. In case (i), an Routing Algorithm (RA) selects the best output port according to the factors like output link bandwidth, the number of free VCs in the downstream routers, and the paths latency to the destination which is achieved by means of the Q-learning method [31]. Conversely, the static solution, i.e., case (ii), just requires the calculation of the appropriate link deactivation threshold value, and the link deactivation decision is always made by the local router. The calculation can be performed off-line according to the link structure, traffic pattern, and the underlying RA. Considering that we rely on the FS method to make use of the partially defected links and the FS induced link latency increases slowly when the link fault level is low and fast when the fault level is high, it is easy to determine an optimal link deactivation threshold and thus we choose the static solution in this dissertation. We note that the link bandwidth and delay aware selection strategies, e.g., [4, 72, 100], can still be applied to select the best output port.

Deactivated links must be dealt with by means of a FTRA. When a Heavily Defected (HD) links is deactivated, the other link in the same interconnect is usually still functional and should be utilized if possible to preserve the link capabilities. Note that in this dissertation, we assume that each interconnect between two adjacent NoC routers consists of two unidirectional links, each link having its own control flow wires and handling either outgoing or incoming traffic. Although the unidirectional links can be replaced with bidirectional

ones as suggested by Tsai et al. [97], when an input or output port is broken, a bidirectional link also becomes unidirectional. Moreover, unidirectional links are still attractive as they provide better means to implement the control logic and to address timing error issues [95]. However, most state of the art FTRAs, e.g., [1,16,17,19,29,39,56,81,101,116], abandon the entire interconnect even when only one link is broken. Thus the UnPaired Functional (UPF) links in such partailly broken interconnects are wasted even though utilizing the UPF links can partially preserve the link capabilities and can result in graceful system performance degradation.

In this chapter, we first discuss the optimal link deactivation threshold by comparing the zero load packet transmission latency on the HD links and that on the shortest alternative path and then propose a distributed logic based FTRA to tolerate the deactivated link as well as efficiently utilize the UPF links. The basic fault pattern tolerated by the proposed UPF links aware FTRA (UPF-FTRA in short) is a fault wall, which is composed of adjacent broken links with the same outgoing direction. Messages are routed around the fault walls along the misrouting contours of the broken links. It requires a minimum number of 3 Virtual Channels (VCs) and dynamically reserves them to messages whose transmission is blocked to guaranty deadlock freeness. The UPF-FTRA and the link deactivation threshold are evaluated with both synthetic traffic and recorded real traffic traces. Experimental results indicated that UPF-FTRA can improve the NoC saturation throughput by up to 22% when compared with state of the art counterparts. Synthesis results with TSMC 65nm technology indicate that, embedding UPF-FTRA into a baseline router increases the area and power overhead by 9% and 2% respectively, which is similar with that of the conventional solid fault region based algorithms. Simulation results we obtained at various wire broken rate configurations indicate that we achieve the highest saturation throughput if 4- or 8-section links with a flit transmission latency longer than 4 cycles are deactivated.

The rest of the chapter is organized as follows. Section 5.2 presents a brief related work survey. Section 5.3 discusses the optimal threshold to deactivate HD links. Section 5.4 describes the proposed fault tolerant routing algorithm and proves its deadlock freeness property. Section 5.5 evaluates the performance of the proposed algorithm and validate the efficiency of the selected link deactivation threshold. Section 5.6 concludes the presentation.

5.2 Related Work

Targeting at different fault cases, numerous strategies are proposed to route packets in awareness of the link bandwidth variation and tolerate deactivated or totally broken links.

5.2.1 Link Bandwidth Aware Routing

Utilizing defective links with low fault level can preserve NoC bandwidth and avoid severe performance degradation. Moreover, if the underlying routing algorithm is adaptive, a proper path selection strategy can be applied to determine the path with the lowest data transmission delay. However, Heavily Defected (HD) links may cause severe congestion in the upstream routers, and thus should be discarded. Whether to utilize or abandon a defected link should be decided in such a way that the system performance lose is minimized.

In [72], Palesi et al. proposed an application specific routing function with a set of selection policies which are aware of the link fault distribution. At each routing hop, the best admissible output port is selected with a probability determined according to the link fault level and the traffic conditions. The probability for each link is off-line computed and stored in a routing table. This strategy provides the best system performance when executing a certain specific application. However, it requires complicated off-line computations and accurate traffic analysis, which makes it not suitable for dynamically changing systems.

In [100], Vitkovskiy et al. proposed a path selection strategy which always chooses the next progressive hop that has maximum available Virtual Channel (VC) amount and minimum Effective Link Utilization (ELU). The ELU of a link is the product of the number of flits that traverse this link and the flit transmission latency on the link. For example, a fault free link and a link with a flit transmission latency of 2 cycles have the same ELU if 50 and 25 flits are transmitted via each link, respectively, in the same time period. Vitkovskiy et al. also discussed the impact of discarding HD links on the system performance. However, they did not propose a method to decide if a defected link should be discarded or not.

When defective links are utilized, they exhibit longer data transmission latency than fault free links. Thus all the delay or bandwidth aware Routing Algorithms (RAs), e.g., [4, 30, 31, 71, 100], can be employed to select the optimal routing path. Such algorithms usually select the best output port from

multiple admissible ones according to factors like the output link bandwidth, the number of free VCs in the downstream routers, and the path latency to the destination achieved by means of the Q-learning method [31].

Nevertheless, for some minimal path adaptive RAs, e.g., Opt-Y [88], when the packets are already in the same column or row with the destination routers, there is only one admissible output path and which is utilized regardless of the fault level of path links, even if discarding the HD links and detouring the packets could be a better option. Thus it is necessary to determine in which conditions HD links should be deactivated.

5.2.2 Fault Tolerant Routing Algorithms

State of the art fault tolerant RAs can be roughly classified into three groups based on the number of tolerated faults and the way they are tolerated.

RAs in the first group, e.g., [29, 39, 116], can tolerate a bounded number of faults by modifying the baseline RA turn rules around the faults without causing deadlock. Usually the system performance degradation is minimal when faults occur as the modifications are turn based only and do not constraint the VC usage in each of the routers. However, their application scope is limited to NoC systems with low fault rates.

RAs in the second group, e.g., [17,19,56,101], can tolerate an unbounded number of faults, but the regions formed by the faults must satisfy certain requirements, e.g., the fault regions must have solid shapes like +, L, or T [17, 19]. Otherwise, some fault free routers have to be deactivated to make the shape solid. To avoid deadlock, RAs in this group usually constrain the VC usage on the fault region boundaries, which results in a substantial system performance degradation. The main advantage of such kind of Solid Fault Region Tolerant (SFRT) RAs is that VCs are utilized according to the underlying RA in the fault free area. This provides the best system performance to applications if their tasks are best effort based mapped on the NoC fault free area.

RAs in the third group, e.g., [1, 16, 81], can tolerate an unbounded number of faults and do not pose any restrictions on the faulty region. The routing path between source and destination routers is determined by searching the best path during message transmission or NoC reconfiguration, when a new fault is detected. These routing paths are stored in routing tables to indicate how the following messages have to be transmitted. RAs in this group have the advantage that they can utilize almost all functional resources as long as they are connected with the NoC and usually require a low number of VCs. However,

they also have drawbacks when utilized in wormhole switched NoCs. On one hand, a routing table is maintained in each router to indicate the output ports to requested destinations. The routing table sizes are proportional to the NoC size, which makes such approaches less scalable than distributed RAs and introduces a high silicon area overhead. On the other hand, when the VC usage is constrained, it is applied throughout the entire NoC, i.e., both in the fault free area and in the faulty region, which results in unjustified performance reduction as it is known that increasing the VC usage flexibility can substantially improve the system performance.

In view of the previous discussion, we propose a distributed logic FTRA which can tolerate an unbounded number of faults as well as efficiently utilize the UPF links, and thus provides more graceful system performance degradation when compared with the aforementioned RAs.

5.3 Heavily Defected Links Deactivation Threshold

For totally broken links, the only option is to discard them and make use of FTRAs to route packets along alternate paths. For partially defected links, utilizing the ones that have low fault level can partially preserve the NoC link bandwidth and reduce the transmission latency overhead caused by packet detouring and congestion, while utilizing Heavily Defected (HD) ones may cause server congestion in the upstream routers.

From (4.1) we can observe that for the FS strategy, the link flit transmission latency is inversely proportional with the number of fault free link sections. Thus for a k-section link, when the number of broken sections increases from b to $b+1$, the flit transmission latency on this link becomes $(k-b)/(k-b-1)$ times higher. For example, when each link is divided into 8 sections, the flit transmission latency is only increased by 14% when a new section of a fault free link becomes broken, while the latency is doubled when the number of broken link sections increases from 6 to 7. This FS property makes it easy to decide the optimal link deactivation threshold.

Take the case illustrated in Fig. 5.1 for example. Assuming a packet is waiting to be transmitted from router C to router N. By default, the packet should be transmitted via L_0. The question now is: If L_0 is defected, at which fault level should we abandon it and detour the packets to achieve minimum system performance degradation?

If L_0 is deactivated, most probably the packets will be misrouted along alterna-

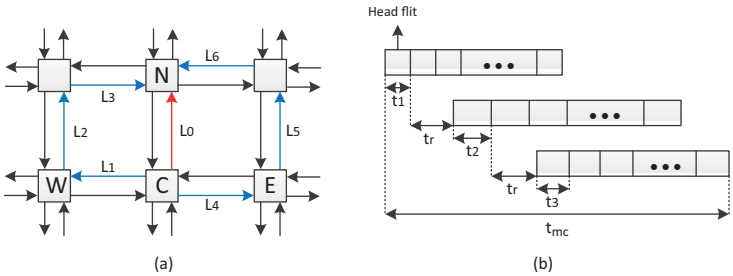


Figure 5.1: Detouring example. (a) The misrouting-contour of L_0. (b) Detouring delay.

tive paths formed by its adjacent links, i.e., $L_1 \rightarrow L_2 \rightarrow L_3$, or $L_4 \rightarrow L_5 \rightarrow L_6$. We refer to such paths with the concept of *misouring-contour*. According to the outgoing direction of L_0, we can divide its misrouting-contour into the left half, i.e., $L_1 \rightarrow L_2 \rightarrow L_3$, and the right half, i.e., $L_4 \rightarrow L_5 \rightarrow L_6$.

Let us assume that the packet length is P, the NoC operates according to the wormhole switching technique [24], and each router has 3 pipeline stages. At zero traffic load, the time required to transmit an entire packet to router N via L_0 (T_{L_0}) and its left half misrouting-contour (T_{mc}) can be expressed as (5.1) and (5.2), respectively. Here we assume that the packets are detoured along the left side misrouting-contour by default, but we note that the analysis to the right half misrouting-contour can be done in a similar way.

$$T_{L_0} = Pt_0, \tag{5.1}$$

$$T_{mc} \geq t_1 + t_r + t_2 + t_r + Pt_3, \tag{5.2}$$

where t_i ($\geq$ 1 cycle) is the flit transmission latency on link L_i, i=0,1,2,3, and t_r ($\geq$ 3 cycles) is the latency to traverse a 3-stages pipelined router. In (5.2), T_{mc} is larger than the right side polynomial when t_1 or t_2 is is large enough to create the situation when a flit arrives at a router input port after all precedent flits have already been transmitted to the next hop.

Obviously, we should deactivate L_0 and detour the packets on the misrouting-contour when

$$T_{L_0} > T_{mc} \geq 8 + P, \tag{5.3}$$

i.e.,

$$t_0 > (t_1 + t_2 + 6)/P + t_3 \geq 8/P + 1. \tag{5.4}$$

Thus the minimum link deviation threshold is inversely proportional to the packet length, e.g., the threshold is $t_0 > 3$ cycles when the packet length is

4 flits and decreases to $t_0 > 1.5$ cycles when the packet length is 16 flits. However, in practice, T_{mc} is much higher than $8 + P$ due to the fact that: (i) the links on the misrouting-contour are not always fault free, (ii) detouring the packets increases the congestion on the misrouting-contour especially at high traffic load, and (iii) extra flow control delay [24] should be considered in T_{mc}.

We note that at near zero traffic load, the possibility for a detoured packet and a normally transmitted packet to compete for the same NoC resource is low, case in which deactivating HD links at the minimum threshold can reduce the packet transmission latency. However, the congestion on the misrouting-contour increases as the traffic load gets higher, thus if L_0 is deactivated when T_{L_0} is only slightly higher than T_{mc}, the packet transmission latency on the misrouting-contour might surpass the one on L_0. This implies that in practice, to reduce the degradation speed of the saturation throughput, the link deactivation threshold should be set higher than the minimum one. In fact, even at low traffic load, moderate increase of the link deactivation threshold will only bring negligible increase of the average packet transmission latency because the FS induced flit transmission latency increases slowly when less that 75% of the sections are faulty and cannot be utilized. Thus L_0 should be deactivated only when $T_{L_0} \gg T_{mc}$. In view of such analysis, we adjust the minimum link deactivation threshold to $t_i > 4$ cycles for both short and long packets, i.e., an 8-section link is deactivated when it has 7 or more broken sections, and a 4-section link is deactivated when all link sections are broken. The effectiveness of the selected threshold is validated in Section 5.5.3.

5.4 Unpaired Functional Link Aware Fault Tolerant Routing Algorithm

The deactivated links must be tolerated by means of a FTRA to maintain the functionality of the NoC system. In this section, we propose a FTRA which can tolerate the broken links and efficiently utilize UPF links in partially defected interconnects. In the NoC fault free area, messages are transmitted according to the underlying RA. When a message is blocked by a broken link, the proposed RA applies. In this chapter, we employ the optimal fully adaptive RA Opt-Y [88] as the underlying RA, but note that other RAs can also be utilized.

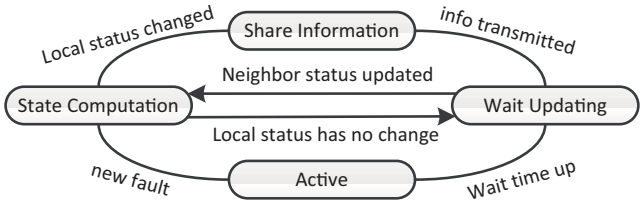


Figure 5.2: Flow chart of fault pattern validation FSM in each router.

5.4.1 Fault Pattern Validation

When a link is deactivated or detected to be totally broken, the NoC fault pattern must be validated before messages can be forwarded. The fault pattern validation is controlled by a Finite State Machine (FSM), illustrated in Fig. 5.2, implemented in each NoC router. Similar with the fault pattern validation process proposed in [56], we assume that each active router has a self-test mechanism to periodically diagnose itself and the neighbor routers. In the rest part of this chapter, we interchangeably use broken and deactivated to refer to the out of service links.

The normal router state is *Active*. If a new broken link is detected, the router enters the *State Computation* stage and computes its status and the ones of its incident links. A router is deactivated if it has 3 or more broken TX or RX links. If a router has 2 broken TX links in different dimensions, i.e., X and Y, the router is marked as a TX Concave (TC) router. Similarly, an RX Concave (RC) router has 2 broken RX links in different dimensions. If the status of a router, i.e., active or deactivated, and TC or RC, or status of any link, i.e., broken or functional, incident to it did change, the router broadcasts the new status information to the 4 neighboring routers in the East (E), West (W), South (S), and North (N), in the *Share Information* stage via a Triple Modular Redundancy (TMR) protected serial wire. Otherwise, the router enters the *Wait Updating* stage.

Each TC (RC) router sends a TC (RC) flag to its neighbor if the TX (RX) link to (from) that neighbor is still functional. Upon receiving a TC (RC) flag from one direction, e.g., W, the router checks if it has a faulty TX (RX) link in an orthogonal direction, i.e., N or S in this case, and if the neighbor where the flag comes from also has a faulty TX (RX) link in the same direction. If this holds true, the router forwards the TC (RC) flag to the next neighbor, i.e., the neighbor in the W in this case. If a router has received TC (RC) flags from two opposite directions in the same dimension, e.g., W and E, and has a broken TX

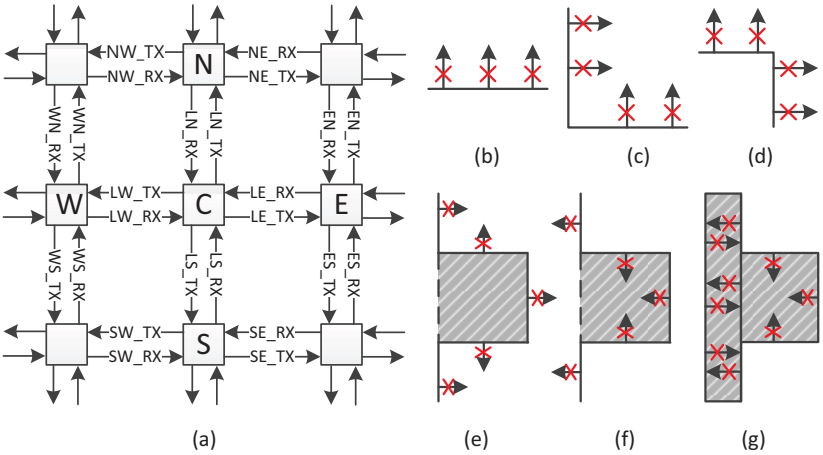


Figure 5.3: Validated fault pattern. (a) Routers and links seen by router C; (b-g), Fault Patterns can be tolerated by the proposed RA.

(RX) link in another dimension, i.e., N or S, the router and all links incident to it are deactivated.

If status information from a neighboring router is received while being in the *Wait Updating* stage before the waiting time up, the router computes the local status again. Otherwise, the router returns to the *Active* stage. The waiting time is set to be equal with the longest time required to transmit the status information from one NoC edge, e.g., the W edge, to another edge, i.e., the E edge. We note that after the fault pattern is validated, each router is aware of the statuses of 4 routers and 24 links adjacent to it, as illustrated in Fig. 5.3(a). Data transmission pausing when a new fault is detected and resuming after the fault pattern is validated can be managed with the strategy proposed in [51].

The most basic fault pattern that can be tolerated by the proposed RA is depicted in Fig. 5.3(b), where adjacent links in the same direction (N in this case) are broken. We note that the number of faulty links and their directions are not restricted in any fault pattern. When such *Fault Wall* exists, only the data transmission in that direction is blocked. The direction of a fault wall is the direction of the broken links that form the fault wall. More complex fault patterns, e.g., the ones illustrated in Fig. 5.3(c-g), can be formed by several fault walls, which have different fault directions. Some routers may be deactivated to make the fault patterns solid as in Fig. 5.3(e-g). We note that all fault patterns that can be tolerated by SFRT RAs are tolerable by the proposed RA.

5.4.2 Turn Rules

In a 2D mesh NoC the direction of each link is WE, EW, SN, or NS according to the position of its source and destination routers. A message which is transmitted from node n_s=(x_s, y_s) to node n_d=(x_d, y_d) is labeled as WE if $x_d > x_s$, or EW if $x_d < x_s$. When the message reaches its destination column, the message type changes to NS if $y_d > y_s$ or SN otherwise and cannot change its type again. WE and EW messages are row messages, while NS and SN messages are column messages [17].

Among the admissible output ports provided by Opt-Y, one is determined by e-cube RAs [17], e.g., XY. We name the hop via that port as *e-cube hop*, and the others as *adaptive hops*. The e-cube hops for WE, EW, NS, and SN messages are E, W, S, and N ports, respectively.

A message's status is normal if it is a row message and the e-cube hop is not blocked; or 2) it is a column message, the e-cube hop is not blocked, and the current router is in the destination column. Otherwise, the message status is misrouting. If the e-cube hop of a normal message is on the misrouting-contour of a broken link, only the e-cube hop can be utilized. If an adaptive hop is on the misrouting-contour of a broken link, the adaptive hop cannot be utilized.

When a message is blocked by a fault wall, it is misrouted around the fault wall along the misrouting-contours of the broken links. The default misrouting direction is clockwise. However, the misrouting direction should be counter-clockwise if with the default misrouting direction row messages are forced to return to the previous column or column messages are forced to return to the previous row by another fault wall or an NoC edge. This can be anticipated by checking if a TC flag has been received from the port in the clockwise direction. For example, in Fig. 5.4, messages M1, M4, M5, and M8 are misrouted in clockwise direction, while messages M2, M3, M6, and M7 are misrouted in counter-clockwise direction as otherwise they will be forced to return to the previous column or row as illustrated by the dashed arrows.

For a misrouted row message, its status becomes normal when the e-cube hop, i.e., E or W, is functional. Thus we can simply misroute row messages to S or N until a functional e-cube output port is found. With this routing method, EW links are never utilized by WE messages, and WE links are never utilized by EW messages.

To avoid livelock as illustrated in Fig. 5.5(a), we use the localized routing scheme proposed in [101] to misroute column messages. When a column message is blocked, it anticipates the shortest path to the destination router of the

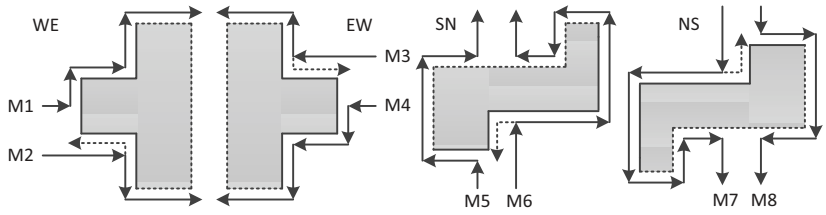


Figure 5.4: Misrouting direction of different messages. The dashed boarder of the shadows may not be fault walls.

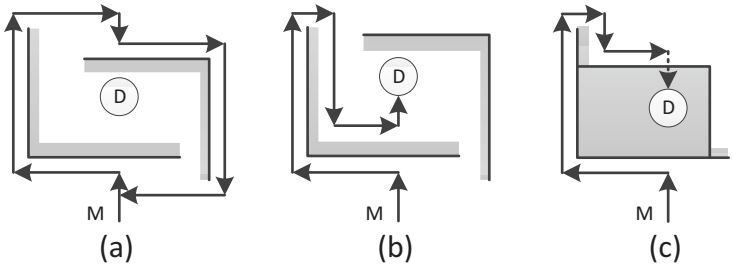


Figure 5.5: Misrouting of column messages. The shadows indicate the directions of fault walls. (a) livelock occurs; (b) destination is reached; (c) destination is not reachable

broken link. The misrouting path is stored in the head flit and is dynamically adjusted in each router in case links in the anticipated misrouting path are also broken. For example, when an SN message is blocked by a broken SN link, the shortest path to the next router in the same column is to take the W, N, and E (WNE) hops. If the N hop is blocked after the W hop, the remained misrouting path changes to WNEE. When the message is in the destination column again, its status returns to normal if the next e-cube hop is not blocked (Fig. 5.5(b)). If the e-cube hop is blocked again and the destination router is still in the North, the message starts another misrouting process. If the destination router is already in the South but the NS link is broken, the destination router must be broken or deactivated and thus unreachable (Fig. 5.5(c)). In this case, the local Process Unit (PU) absorbs the message and sends an message to the source router to report the error.

With this routing method, NS links are utilized by SN messages only when their source routers have received RC flags from the S ports, e.g., when M6 is routed to the south in Fig. 5.4. Similarly, SN links are utilized by NS messages only when their source routers have received RC flags from the N ports, e.g., when M7 is routed to the north in Fig. 5.4.

SN Messages never make E-S-W or W-S-E turns. According to the turn rules, a message turns to the S after one E (W) hop only because the destination column is reached or is still in the E (W). Thus it will not turn to the W (E) again. Similarly, NS messages never make E-N-W or W-N-E turns.

5.4.3 VC utilization Constraints

When messages are misrouted, some forbidden turns in Opt-Y are utilized. To avoid deadlock, extra constraints to the VC usage besides the Opt-Y ones must be applied.

Opt-Y requires two VC classes, Y_1 and Y_2, in the N and S directions. Among all the turns, two 90^o turns, N-W and S-W, using Y_1 are prohibited, and the 0^o turns from Y_2 to Y_1 are allowed only when the message does not need to route further west. In our proposal, 3 VCs, labed as C_0, C_1, and C_2, are required. In the fault free region, C_2 is utilized as Y_1, C_0 and C_1 are utilized as Y_2.

When messages are transmitted on misrouting contours of broken links, whether the message status being normal or misrouting, extra VC usage constraints are in place: C_0 is reserved for row messages on the misrouting-contours of broken NS or SN links. C_1 is reserved for NS messages on the misrouting-contours of broken NS links and in SN links whose source routers have received RC flags from their N ports. C_2 is reserved for SN messages on the misrouting-contours of broken SN links and in NS links whose source routers have received RC flags from their S ports. With this VC usage strategy, VCs are only reserved when necessary. For example, in a link which is only on the misrouting contour of a broken EW link, C_0 is reserved to row messages, while C_1 and C_2 are freely utilized by all column messages. We note here that if extra VCs are available in the NoC, they can be freely utilized by all types of messages.

5.4.4 Deadlock Freeness

To prove that the proposed RA is deadlock free, we just need to check if deadlock can happen when turns forbidden by Opt-Y are utilized, as it is known that Opt-Y is deadlock free [88]. The clockwise Channel Dependency Graph (CDG) of Opt-Y is depicted in Fig. 5.6(a). In the figure, $C_{0,*}$, $C_{1,*}$, $C_{2,*}$, and $C_{3,*}$ are VCs in SN, WE, NS, and EW links, respectively. The turns that are always allowed are illustrated with solid arrows and the turns that are allowed in certain conditions are illustrated with dashed arrows.

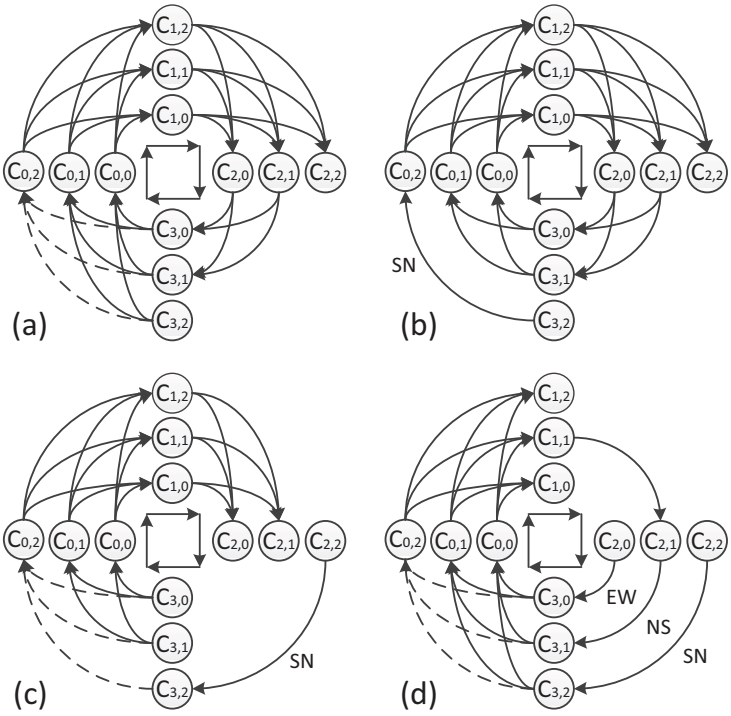


Figure 5.6: Channel dependency graphs.

Dependency $C_{3,2} \rightarrow C_{0,2}$ only occurs when SN messages are forced by broken SN links to make W-N turns (see Fig. 5.6(b)). On the misrouting contours, i.e., in the EW and SN links involved in the W-N turns, $C_{*,2}$ is reserved to SN messages. While other channel dependency remain the same as Opt-Y. Thus the CDG in Fig. 5.6(b) is still deadlock free.

S-W turns from $C_{2,2}$ to $C_{3,2}$ only occur to SN messages. The NS and EW links involved in the turn are on the misrouting-contour of broken EW and SN links, respectively. Thus $C_{*,2}$ is reserved to SN messages in these links. According to the turn rules, the other types of messages do not make S-W turns if their statuses are normal, as illustrated in Fig.5.6(c). The dependency $C_{1,2} \rightarrow C_{2,2} \rightarrow C_{3,2}$ does not exist because SN messages never make E-S-W turns. We can observe that the CDG in the figure is acyclic and thus is deadlock free. If EW messages are forced by broken EW links to make S-W turns, they can only use $C_{*,0}$ in the SN and EW links involved in the turns. Because EW messages never use WE links, the turns from $C_{1,*}$ to $C_{2,0}$ do not exist (see Fig. 5.6(d)). When NS messages are misrouted, they use $C_{*,1}$ only, thus the

turns from $C_{1,0}$ or $C_{1,2}$ to $C_{2,1}$ do not exist. Thus the CDG in Fig.5.6(d) is also deadlock free.

By means of a similar analysis, we can also prove that the counter-clockwise CDG is deadlock free, thus we can conclude that the proposed routing algorithm is deadlock free.

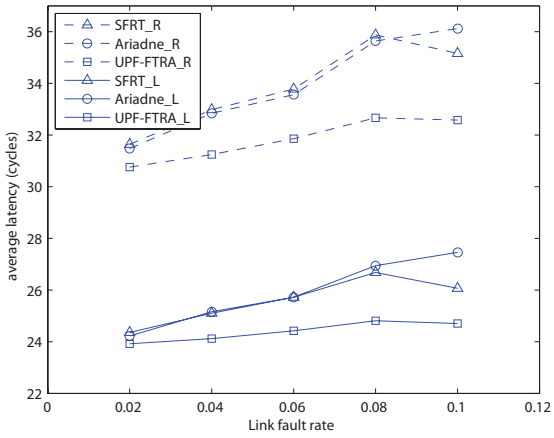
5.5 Evaluation

To put the implication of our proposal in a better practical prospective, we evaluate the proposed UPF-FTRA and compare its figure of merit with the one of tightly related FTRAs, i.e., the SFRT algorithm [19] and the routing table based algorithm Ariadne [1], and examine the effectiveness of the link deactivation threshold in different fault patterns. For a fair comparison, the underlying RA is Opt-Y in all cases and every RA is implemented in the context of an 8×8 2D mesh NoC platform at RTL level by using Verilog HDL. The baseline router has 3 pipeline stages and each VC buffer is 4-flit deep.

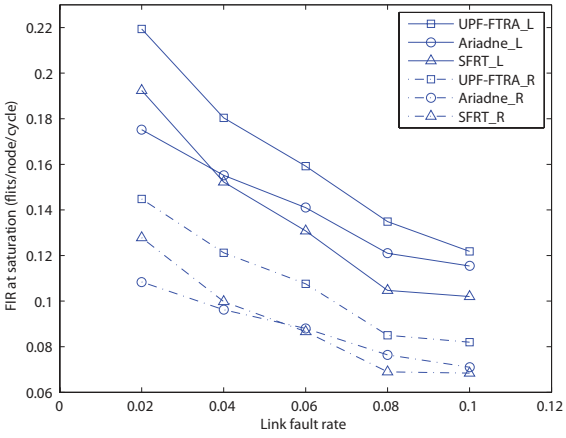
We note that SFRT minimally requires 3 VCs which are statically reserved to row, NS, and SN messages around each fault region, while Ariadne allows all VCs be freely utilized by any message type. In practice, it is beneficial to utilize more VCs than the minimum, e.g., 3 VCs for our UPF-FTRA and SFRT, to eliminate the Head of Line (HOL) blocking issues in the NoC system. In our experiments, 4 VCs are utilized by each RA, with one of them being freely utilized by all message types.

5.5.1 UPF-FTRA Performance on Synthetic Traffic

The three RAs' performance is evaluated in NoCs with different fault rates under different synthetic traffic patterns. The average packet transmission latency at light traffic load, i.e., 0.02 flits per node per cycle, and the saturation throughput, i.e., the Flit Injection Rate (FIR) for which the average packet transmission latency approaches infinity, of the considered RAs are illustrated in Fig. 5.7. Each data point is derived by averaging the results of 100 different fault patterns with the same fault rate and traffic pattern. In the random traffic, the message destinations are uniformly distributed throughout the NoC, while in the localized traffic, 50% of the messages are destined to the 8 nodes adjacent to the source node, which is the case for optimized task mappings, thus can better reflect the system performance in practice.



(a) Average latency;



(b) Saturation points.

Figure 5.7: NoC performance at different fault rates and traffic patterns. In the legend, R means random traffic pattern, and L means localized traffic pattern.

Although UPF-FTRA can also be classified into the second RA group (see Section II), it can utilize more resources and has more VC usage flexibility than SFRT, thus has better performance. As we can observe in Fig. 5.7, when compared with SFRT, UPF-FTRA provides a packet transmission latency reduction of 6% and 5%, and a saturation point increase of 20%, for random and localized traffic, respectively. Note that some routers are deactivated when the

link fault rate increases from 0.08 to 0.1 to achieve valid fault patterns when UPF-FTRA and SFRT are applied. In such scenario, the total number of packets injected into the NoC per cycle is decreased and the traffic load becomes lighter, thus the average transmission latency is reduced.

Ariadne explores the routing paths between any two routers during the NoC reconfiguration which is triggered by the detection of a new fault. The routing paths, although with limited adaptive capability, are fixed for any traffic pattern. Although UPF-FTRA has less VC usage freedom than Ariadne around the faults, it transmits messages according to the underlying RA, i.e., Opt-Y, in the fault free area, and thus has more balanced traffic distribution. When the link fault rate is low (0.02), the fault free area in the NoC is large, thus our proposal has 34% and 25% higher saturation points than Ariadne for random and localized traffic, respectively. As the fault rate increases, the fault free area shrinks. When the fault rate is 0.10, our proposal still has 15% and 6% higher saturation points than Ariadne for random and localized traffic, respectively. In all the evaluated fault patterns, UPF-FTRA has slightly lower ($< 5\%$) packet transmission latency and on average 22% and 14% higher saturation points than Ariadne for random and localized traffic, respectively.

5.5.2 UPF-FTRA performance on PARSEC Benchmarks

The RAs are also evaluated with recorded traffic traces of PARSEC [9] benchmarks. The average packet transmission latency of the applications in different fault circumstance is illustrated in Fig. 5.8.

Although the execution time of an application can be affected by multiple issues, including traffic load, fault pattern, and so on, UPF-FTRA out performs SFRT and Ariadne for all evaluated applications. It is worth to mention that the average packet latency of both UPF-FTRA and SFRT increase monotonically as the percentage of broken links in the NoC increases, while that of Ariadne experiences ups and downs. This is because the performance of Ariadne highly depends on the structure of the connecting tree built by the router that first detects the new broken link.

5.5.3 The Effectiveness of the Link Deactivation Threshold

In this section, we divide the links into 8 sections and examine the performance of the proposed defective link utilization strategy at different link deactivation threshold. The considered thresholds are T5, T6, T7, and T8, i.e., a link is

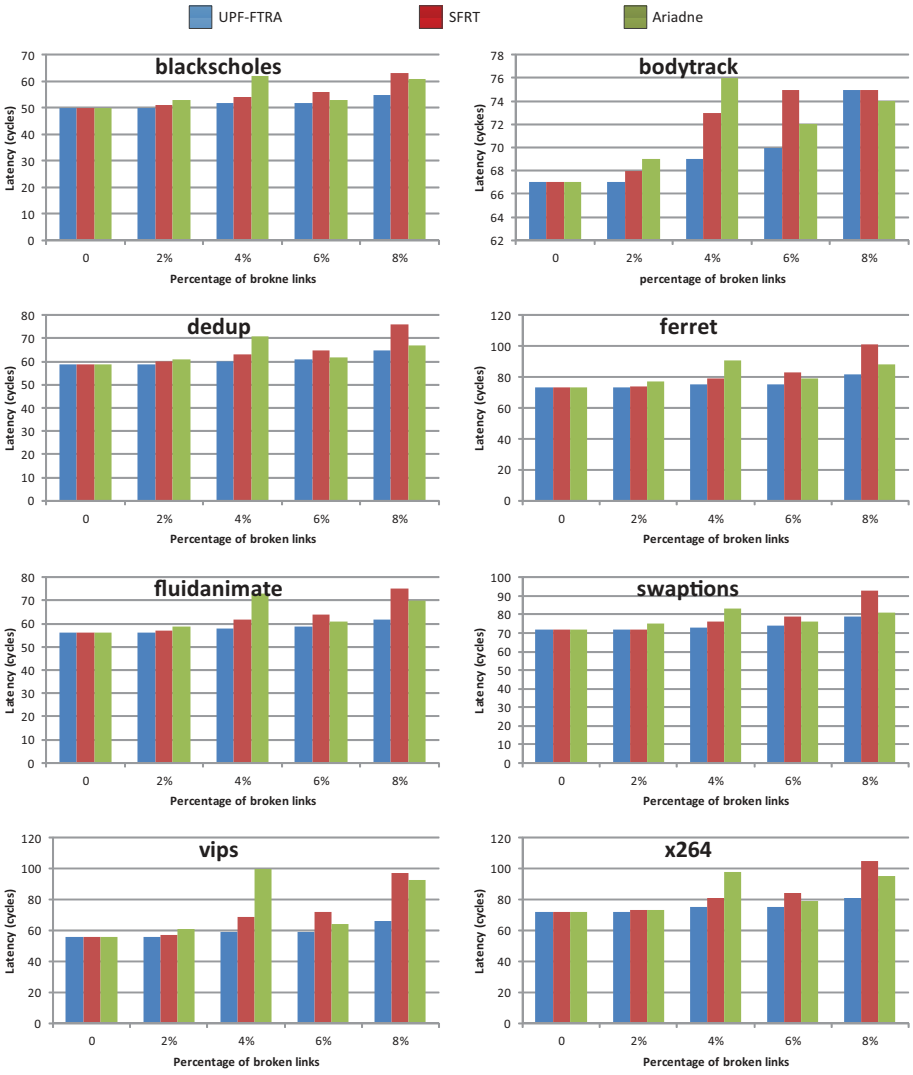


Figure 5.8: Average packet transmission latency (cycles) of different benchmarks when the NoC has different percentage of broken links.

deactivated when the number of broken sections is equal with or larger than 5, 6, 7, or 8, respectively. In the experiments, we randomly select 1%, 5%, 10%, and 15% of the NoC links and inject 20% to 40% broken wires into them to create Heavily Defected (HD) links, while the wire fault rate in the other links is 5%. The percentage of links with different number of broken sections at different wire fault rates are illustrated in Fig. 5.9 which indicates

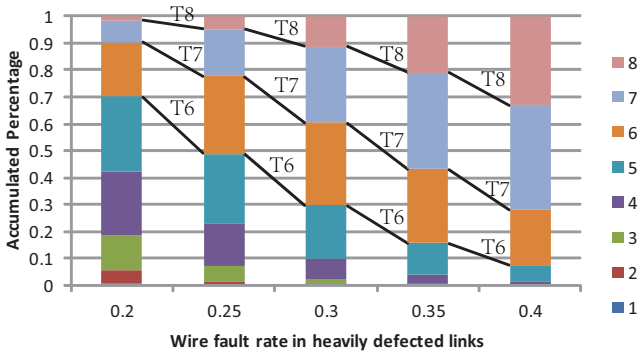
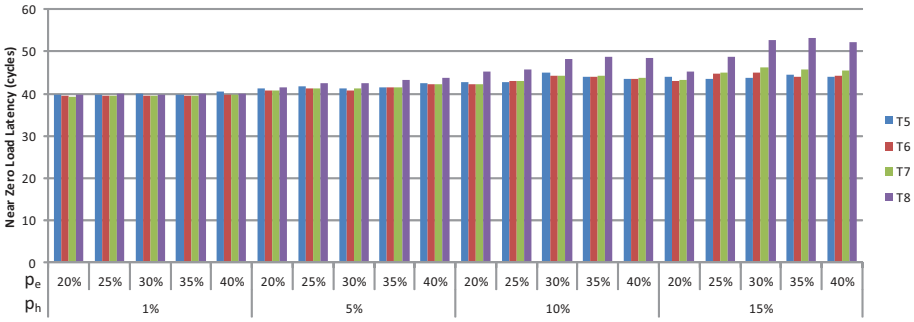


Figure 5.9: The link fault level change trend at different wire fault rate. The legend is the number of broken link sections in a heavily defected link.

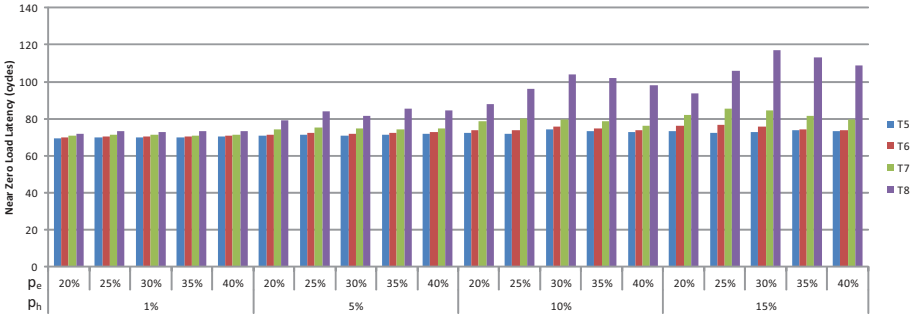
that most links have 5 or more broken sections when $p_e \geq 20\%$. As the link deactivation threshold increases from T5 to T7, an decreasing number of links are deactivated. However, as p_e grows, the number of broken sections in the HD links increases and thus more links are deactivated at the same link deactivation threshold.

The NoC system near zero load (FIR = 0.01 flits/cycle/node) packet transmission latency and saturation throughput at different wire fault rate configurations are illustrated in Fig. 5.10 and Fig. 5.11, respectively. The x-coordinates in the figures indicate the percentage of HD links (p_h) in the NoC and the percentage of faulty wires (p_e) in the HD links. We simulate the cases of short (4-flits) and long (20-flits) packets. Each experimental result is derived by averaging the results of 20 fault patterns with the same fault rate configuration. We note that when each link is divided into 8 sections, the average number of cycles required to successfully transmit a flit via a link with 5, 6, 7, and 8 broken link sections are 2.67, 4, 8, and ∞, respectively.

The results in Fig. 5.10 indicate that the average packet transmission latency at near zero traffic load in the T5, T6, and T7 cases has only small variation. Specifically, when compared with T6, the latency in T5 is slightly higher when the packet length is 4-flits but slightly lower when the packet length is 20-flits. This indicates that the link deactivation threshold for short packets should be set higher than that for long packets, which validates our analysis in Section 5.3. The difference between T6 and T5 is that the links that contain 5 broken sections are utilized in T6 but are deactivated in T5. According to the analysis in Section 5.3, the packet transmission latency via links at such fault level is similar with that via their misrouting-contour, thus T5 and T6 achieve similar



(a) Average packet transmission latency when packets length is 4-flits;

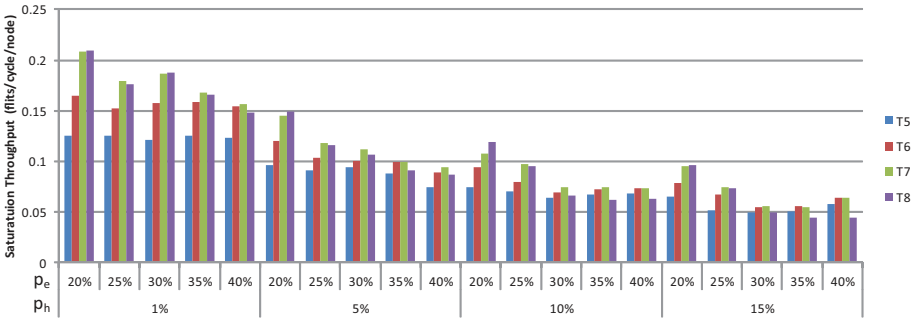


(b) Average packet transmission latency when packets length is 20-flits;

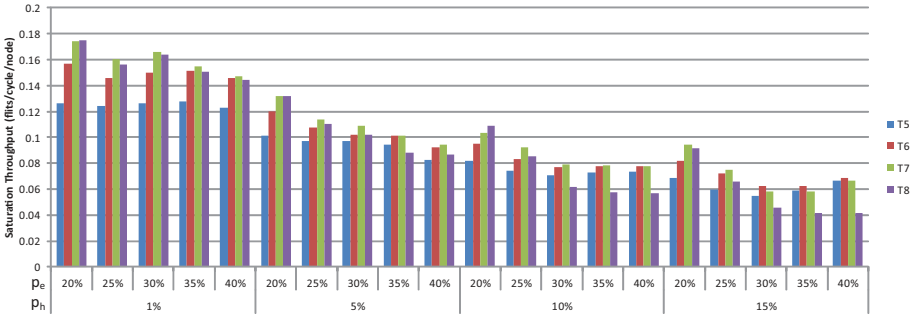
Figure 5.10: The system average packet transmission latency when deactivate links with high fault level with different threshold.

near zero load performance. Due to similar reason, although slower links, i.e., the links with 6 broken sections, are still utilized in the T7 case, its near zero traffic load packet transmission latency is only slightly higher than that of T6, e.g., less than 9% even when $p_h = 15\%$. By comparison, links are utilized until all link sections are broken in the T8 case, case in which the links with a flit transmission latency of 8 cycles induce severe congestion in their upstream routers. Consequently the packet transmission latency in T8 is obviously higher than that of the cases with lower thresholds when $p_h \geq 5\%$.

The results in Fig. 5.11 indicate that T7 can achieve the highest saturation throughput for most of the fault rate configurations. We can also observe that as the link deactivation threshold increase from T5 to T7, although the near zero load packet transmission latency increases slowly, the saturation throughput is quickly improved. This is caused by the fact that when the NoC traffic load is high, deactivating HD links that contain 5 and 6 broken sections cause high congestion on their misrouting-contours, and thus it is more beneficial to



(a) Saturation throughput when packets length is 4-flits;

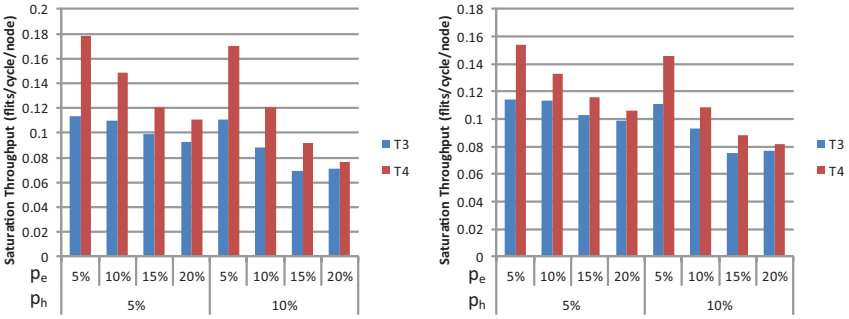


(b) Saturation throughput when packets length is 20-flits.

Figure 5.11: The system saturation throughput when deactivate links with high fault level with different threshold.

directly transmit packets along these HD links. In the T8 case, the links that have 7 broken sections are so slow that the congestion in their upstream routers when they are utilized is much severer than the congestion in the misrouting-contours when they are deactivated. In the extreme case all VCs in an input port can be occupied by packets that are transmitted via such a slow TX link, case in which all subsequent packets are blocked even if they will be routed to other output ports. Consequently the saturation throughput at T8 is lower than that at T7.

It is worth to mention that when $p_h > 0.10$ in the NoC and $p_e > 30\%$ in the HD links, too many links are deactivated and some routers are also deactivated by UPF-FTRA to avoid deadlock. The number of deactivated routers increases as p_e grows. Consequently fewer packets are injected into the NoC and the near zero load packet transmission latency decreases and the saturation FIR for each node becomes higher.



(a) Packets length is 4-flits; **(b)** Packets length is 20-flits.

Figure 5.12: The system saturation throughput at different link deactivation threshold when each link is split into 4 sections. A link is deactivated if 3 and 4 sections are broken in the T3 and T4 cases, respectively.

In conclusion, when the NoC links are divided into 8 sections and partially broken links are utilized by means of the FS method, deactivating links that have 7 or more broken sections can efficiently balance the requirements for low near zero load packet transmission latency and high saturation throughput. In other words, the links should be utilized when their flit transmission latency is 4 cycles or less.

When the links are divided into less, e.g., 4, sections the average link fault level is higher at the same wire fault rate, thus if a link is deactivated the packet transmission latency on its misrouting-contour also becomes longer. This means that 4-section links should only be deactivated when the flit transmission latency on them is longer than 4 cycles, i.e., when all link sections are broken. This is also proved by the results illustrated in Fig. 5.12 that the saturation throughput for T4 is on average 33% and 18% higher than that at T3 for 4-flit and 20-flit packets, respectively.

5.5.4 Area and Power

Routers equipped with different RAs are synthesized by using the Synopsys Design Compiler and the TSMC 65nm technology. The silicon area and power consumption overhead corresponding to different RAs are presented in Table 5.1. From the table we can observe that embedding SFRT, UPF-FTRA, and Ariadne into a baseline router increases the silicon area cost by 8%, 9%, and 10%, respectively, and increases the power consumption by 1%, 2% and 5%,

Table 5.1: Area and power overhead of different RAs. The NoC size is 10×10 for Ariadne* and 8×8 in other cases

	Baseline	SRFT	UPF-FTRA	Ariadne	Ariadne*
Area	67098	72384	73231	73544	75635
(μm^2)	100%	108%	109%	110%	113%
Power	26.01	26.332	26.58	27.21	27.98
(mW)	100%	101%	102%	105%	108%

respectively, when the NoC size is 8×8. UPF-FTRA requires more area than SFRT because the two unidirectional links in each interconnection are separately considered thus more registers are required to record the status of routers and links. It is worth to mention that the area cost of UPF-FTRA and SFRT is independent on the NoC size, while the routing table size in Ariadne increases proportionally with the number of NoC routers. For example, for a 10×10 NoC, the area and power overhead induced by Ariadne increases to 13% and 8%, respectively.

5.6 Conclusion

In this chapter, we discussed the optimal threshold to deactivate HD links and propose a FTRA to tolerate the deactivated links as well as to utilize Unpaired Functional (UPF) links in partially broken bidirectional interconnects. The optimal link deactivation threshold is determined by comparing the zero load packet transmission latency on the HD links and that on the shortest alternative path. The basic fault pattern tolerated by the proposed UPF link aware FTRA (UPF-FTRA) is a fault wall, which is composed of adjacent broken links with the same outgoing direction. Messages are routed around the fault walls along the misrouting-contours of the broken links. Our proposal is evaluated with both synthetic traffic and PARSEC benchmarks. Experimental results indicated that UPF-FTRA can improve the NoC saturation throughput by up to 22% when compared with state of the art counterparts. Synthesis results with TSMC 65nm technology indicate that, embedding UPF-FTRA into a baseline router increases the area and power overhead by 9% and 2% respectively, which is similar with that of the conventional solid fault region based algorithms. Analysis suggests that the links with a flit transmission latency longer than 4 cycles should be deactivated and dealt with UPF-FTRA if they are divided into 4 or 8 sections. Simulation results on synthetic traffic patterns

indicate that we achieve the highest saturation throughput at this threshold in various wire broken rate configurations.

When partially faulty links are efficiently utilized and deactivated links are tolerated, the identification of novel mapping heuristics able to take advantage of link bandwidth variations can be viewed as the natural continuation of our research at a higher abstraction level. Our link bandwidth aware task mapping quality metrics and backtrack based run time task mapping heuristic are presented in the next chapter.

Note. The contents of this chapter is based on the the following papers:

C. Chen, S. D. Cotofana, **An Effective Routing Algorithm to Avoid Unnecessary Link Abandon in 2D Mesh NoCs**, Proc. Euromicro Conference on Digital System Design (DSD), pp. 311–318, Sep. 2013.

C. Chen, S. D. Cotofana, **Towards an Effective Utilization of Partially Defected Interconnections in 2D Mesh NoCs**, Proc. IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pp. 492-497, Jul. 2014.

C. Chen, Y. Fu, and S. D. Cotofana, **Toward Maximum Utilization of Remained Bandwidth in Defected NoC Links**, submitted.

6

Link Bandwidth Aware Task Mapping

With the Flit Serialization (FS) strategy and the UnPaired Functional link aware Fault Tolerant Routing Algorithm (UPF-FTRA), the Network-on-Chip (NoC) performance is gracefully degraded in the occurrence of permanent faults. When new applications are injected into the NoC based Multi-Processor Systems-on-Chip (MPSoCs), it is a nature idea that the faults must be properly considered during the application mapping process to avoid substantial performance penalties. In this chapter, we propose a run-time task mapping algorithm, which takes both the path traffic load and link bandwidth into the consideration and maps applications onto contiguous near convex NoC regions to reduce the internal and external congestion. We rely on a backtracking strategy to guaranty that the maximum link traffic load does not exceed a given limit determined by the link bandwidth and a loose factor. The loose factor is employed to adjust the maximum percentage of link bandwidth that can be utilized. To evaluate our proposal we map synthetic (TGFF tool generated) and real video processing applications on partially defective 8×8 NoCs. The experiments indicate that our approach substantially outperforms equivalent state of the art task mapping heuristics when NoC defects are present, e.g., for 5% broken wires, we achieve at least 16% communication cost reduction and 45% shorter average packet transmission latency.

6.1 Introduction

On NoC based MPSoCs, applications are usually split into a set of concurrent tasks, which are mapped onto different processor nodes to enable their parallel execution. If the MPSoCs running applications are static, task mapping is performed at design time with sophistic strategies, e.g., Branch and Bound [44], to achieve optimal performance and energy consumption. However, in most

practical situations applications enter/leave the MPSoCs dynamically, change their execution scenario, and the MPSoC region available for their execution is unpredictable. For such cases the task mapping has to be done dynamically, at run time, and this is the focus of this chapter.

Up to date, targeting different optimization goals, numerous task mapping heuristics have been proposed [92]. In mesh NoC based homogeneous system, the mapping quality is usually evaluated with two popular metrics, i.e., Manhattan Distance (MD) and Path Load (PL), as for identical node capabilities the only mapping quality factor affecting is the communication cost between connected tasks. Small MD means low data transmission delay and energy cost, and low PL means that the routing path is not congested. However, MD and PL cannot capture NoC link bandwidth variations induced by, e.g., manufacturing defects, process parameter variation, and chip wear-out effects. Note that in most task mapping heuristics links are supposed to be either fully functional or totally broken while, in fact, partially defective NoC links with narrowed bandwidth can still be utilized and if their diminished bandwidth is carefully considered, better mapping quality could be achieved.

In this chapter, we propose a link bandwidth aware task mapping algorithm for 2D mesh homogeneous MPSoCs, targeted at mapping each application onto a contiguous near square region with balanced traffic load on each link. We introduce the Congested extended MD (CeMD) metric, which takes both link traffic load and bandwidth into consideration, and utilize it to select the best processor node for the execution of each task. We rely on backtracking to guaranty that the maximum link traffic load does not exceed a certain limit determined by the link bandwidth and a loose factor. The loose factor is employed to adjust the maximum percentage of link bandwidth that can be utilized. We evaluate our proposal on synthetic (TGFF generated) and real video processing applications on NoCs with various defective degrees. Experimental results indicate that when 5% NoC link wires are broken, at least 16% communication cost and 45% average packet latency reductions are achieved over state of the art counterpart heuristics.

The rest of the chapter is organized as follows. Section 6.2 presents a brief related work survey. Section 6.3 introduces the task mapping problem and evaluation metrics. Section 6.4 describes our task mapping algorithm and Section 6.5 evaluates its performance and compares it with that of closely related work. Section 6.6 concludes the presentation.

6.2 Related Work

Mapping an application into NoC based MPSoCs at run time encompasses the selection of a set of idle processor nodes and the identification of a task per node placement that meets specific requirements.

Carvalho et al. [15] assign each task to the First Free (FF) node or the Nearest free Neighbor (NN). To compensate for disregarding the communication cost when choosing FF or NN, they proposed to minimize the channel or path load and to limit the search region to the neighbor nodes of the master task in the Best Neighbor (BN) heuristic. Applications mapped with BN have shorter average MD between tasks and thus have less internal congestion, but BN may lead to high external congestion as the mapping of subsequent applications is not considered.

To minimize external congestion, Chou et al. [21] propose the incremental mapping heuristic (INC) which first find a near convex region by selecting nodes with minimum dispersion factor and centrifugal factor in the Neighbor-aware Frontier (NF) of the mapped regions, and then map tasks onto the selected region following the descending sequence of the tasks' total communication volume. For applications that have long execution time, the internal congestion is optimized with priority by forming the optimal region of the application first, and then find such a region in the MPSoCs [20]. According to the observed application behavior, proper heuristic is utilized to optimize the internal or external congestion.

Fattah et al. point out that mapping an application into a contiguous near square region with low Normalized Mapped Region Dispersion (NMRD) helps to minimize both internal and external congestion probability. In the CoNA heuristic [36], the first task that have the largest number of edges is mapped on the first node that have the largest number of neighbors. The subsequent tasks are mapped to the nodes that fit in the smallest square centered in the first node. However, CoNA cannot guarantee that the first node is always surrounded by enough free nodes. Thus they propose a Smart Hill Climbing (SHiC) method [33] to find a contiguous near square region having a number of free nodes equal or slightly larger than the number of tasks. The application is then mapped into the region with the Contiguity Adjustable Square Allocation (CASqA) method [34] which aims at minimizing the Internal Congestion and Energy per Bit (ICEB).

The aforementioned heuristics assume that all links are fully functional and have the same bandwidth. However, in practice, the links can be defected and

thus have narrowed bandwidth. The generic heuristic proposed by Nollet et al. [70] takes the link bandwidth into consideration and backtracking is utilized to ensure that the link traffic load never exceed the bandwidth. However, the link bandwidth is not included in the cost metric utilized for the evaluation of admissible nodes. Note that for the same traffic load on two links with different bandwidth, the congestion on the narrow link is much severer than that on the wide one. Moreover, when the traffic load is low and the bandwidth limitation is never exceeded, tasks are mapped in the similar way the non-backtracking based schemes do.

Most heuristics require a Central Manager (CM) to run the mapping algorithm and monitor the state of all processor nodes. Such a centralized approach can cause communication hot spots around the CM and has the disadvantage of limited scalability and reliability. To solve this issue, agent based distributed task mapping approach, e.g., [32, 35, 57], are proposed. In practice, most CM based heuristics can be easily adjusted to work with distributed managers.

We note that many other run time task mapping heuristic exist with different optimization goals [92]. However, the impact of link bandwidth on the mapping results is not deeply studied in most proposals. The strategy proposed in this chapter thoroughly takes the link bandwidth, path load, and path congestion into consideration and thus can make better matches between tasks and processor nodes.

6.3 Problem Description

An NoC based MPSoCs can be represented by the Architecture Graph $AG(N, L)$, containing a set of processor nodes N interconnected by a set of links L. Data are transmitted in the form of packets composed of a certain number of flits. The link bandwidth ($bw_{u,v} \leq 1$) is the number of flits (≤ 1) that can be transmitted in each cycle thus the link latency is $1/bw_{u,v}$. In this chapter, we assume that partially defected links are utilized by means of the FS method and the underlying routing algorithm is XY. Note that links and routers can also be totally broken case in which fault tolerant routing algorithms, e.g, the UPF-FTRA proposed in Chapter 5, should be utilized, but this has no fundamental consequences on the proposed heuristic.

We assume that each application is already divided into a certain number of tasks that can be mapped onto different nodes and executed in parallel. One task is exclusively mapped onto one processor node, and vice verse. An application (AP) is represented by a directed Task Graph $AP = TG(T, E)$. Each

vertex $t_i \in T$ represents a task. The directed communication between tasks t_i and t_j is represented by an edge $e_{i,j} \in E$. The communication volume of edge $e_{i,j}$, i.e., the total number of flits, is $w_{i,j}$. If T is the execution time of the application with optimally mapped tasks, the Flit Injection Rate (FIR) of each edge is $r_{i,j} = w_{i,j}/T$.

The problem now is to map each injected application into a contiguous near square region in the MPSoC at run time with most appreciated matches between the tasks and processor nodes in the region. The mapping results can be evaluated with one or more of the following metrics.

(1) Extended Manhattan Distance (eMD)

The communication cost of each AP edge is related with the communication volume and the path latency between the source (N_s) and destination (N_d) nodes in AG. When all links are fault free each link can transmit a flit in 1 cycle, thus the path latency is proportional with the Manhattan Distance $MD(N_s, N_d)$ and the application's communication cost can be evaluated with the Average Weighted Manhattan Distance (AWMD) [36]. When the links are partially defected, the path latency should be evaluated with the extended Manhattan Distance (eMD):

$$eMD(N_s, N_d) = \sum_{N_s}^{N_d} 1/bw_{u,v}. \tag{6.1}$$

Where u and v are the routers on the path between N_s and N_d. Consequently, AWMD should be replaced with the Average Weighted extended Manhattan Distance (AWeMD):

$$AWeMD = \frac{\sum \left(w_{i,j} \times eMD(N_{t_i}, N_{t_j})\right)}{\sum w_{i,j}}. \tag{6.2}$$

(2) Average Link Load

When a link $l \in L$ is shared by multiple AP edges, its overall traffic load is $\sum_{l \in e_{i,j}} r_{i,j}$. To reflect the link congestion degree for different bandwidth values, we define the link load as the ratio of the link traffic load to the link bandwidth. The Average Link Load (ALL) is computed with (6.3):

$$ALL = \frac{\sum_{\forall l \in L_u} \sum_{l \in e_{i,j}} r_{i,j}/bw_l}{|L_u|}. \tag{6.3}$$

Where L_u is the set of links utilized by an AP, and $|L_u|$ is the links number.

(3) Standard Link Load Deviation

Balanced load can prevent the NoC communication hot spots occurrence and thus can reduce data transmission latency and energy consumption. We use Standard Link Load Deviation (SLLD) to evaluate how even the traffic is distributed. Only utilized links are considered in SLLD.

$$SLLD = \sqrt{\frac{1}{|L_u|}\left(\sum_{l \in e_{i,j}} r_{i,j}/bw_l - ALL\right)^2}. \tag{6.4}$$

(4) Congested eMD.

Given that multiple acceptable node candidates may exist for one task, the selected node should have minimum eMD with the mapped task neighbors, and the transmission path should be minimally congested. To evaluate the two issues with one metric, we define the Congested eMD (CeMD) between any two nodes in (6.5). Here we assume that when a link is already involved in other communication edges, only the remained link bandwidth can be utilized by the task to map. Note that in practice, a link is exclusively utilized by one edge in each cycle.

$$CeMD = \sum 1/\left(\lambda bw_{u,v} - \sum r_{i,j}\right), \lambda \in \Re^+. \tag{6.5}$$

The λ parameter in (6.5) is the loose factor with a default value 1.0, but in practice, λ can be set slightly higher than the maximum edge FIR of the application, e.g., $\lambda = 0.4$ when $r_{max} = 0.36$. We require the CeMD is non-negative, i.e., the link load should not exceed $\lambda bw_{u,v}$. This helps to balance the traffic load on the links. In case the requirement is too tight that non mapping solution exist, λ can be set higher to loose the constraint. Thus a mapping solution can be reached with the penalty of a higher congestion.

6.4 The Mapping Algorithm

Given that it has been demonstrated that mapping an application into a contiguous near square region helps to reduce the internal and external congestion [21, 33, 36], we first find such a region able to accommodate the number of application tasks, then map the application into the selected region. For the sake of simplicity, the proposed heuristic runs on a CM node which locates at $(0, 0)$ in the mesh NoC. We note that CM can run on any other NoC location and the heuristic can also be easily adjusted to agent based strategies.

6.4.1 Region Selection

In [21], Chou et al. search for available nodes in the NF to minimize the total MD of the occupied and unoccupied region. However, NF has the drawback of not being aware of the shape of the unoccupied region. For example, an application with 9 tasks may select a 2×5 region but not a 3×3 one if the former is closer to the CM. Fattah et al. use the SHiC [33] method to find the first node surrounded by enough free nodes. The task with most communication edges is mapped to the first node. The available nodes around the first node are then gradually discovered by expanding the radius, and assigned to unmapped tasks [34]. However, the first node might not be the best one for the first task, and the best nodes for the next to be mapped tasks may not exist in the current radius. In our proposal, we first find for each application a contiguous near square region with a node cardinality equal with the number of application tasks, and then pick the best node from the region for each task.

The proposed region selection pseudo-code is presented in Algorithm 1 and 2. We explain the algorithm with the example task mapping scenario illustrated in Fig. 6.1, in which App 0, 1, and 2 are already running in the MPSoCs, and APP 3, which has 10 tasks, is waiting to be mapped. The edge FIR and link bandwidth values are only utilized as an example.

Immediately after an application is mapped, we discover the boundary nodes of the unoccupied region, e.g., nodes marked with asterisks in Fig. 6.2, sort them according the descending sequence of their (i) distance to the CM node and (ii) idle neighbors number, and update the maximum free square size attached to each boundary node with the assumption that the node locates in a square corner. When a new application is injected, we search for the maximum square which has equal or slightly smaller number of nodes than the number of application tasks along the boundary nodes. Nodes in the front of the sorted boundary node list are checked first. For APP 3 in Fig. 6.2, the target square side length is 3.

After a square is found, e.g., the one surrounded with dashed lines in Fig. 1, its available nodes are counted and added to the *available_node_list*. If the *available_nodes_num* is smaller than the number of application tasks, the nodes in the square frontier (dark gray nodes in Fig. 1) are checked. The nodes with: (i) less idle neighbors and (ii) smaller eMD from the nodes in the *available_nodes_list* are preferred. In the example in Fig. 6.2, node $(2, 3)$ is selected first. If all nodes in the current frontier are selected but more nodes are needed, the nodes in the next frontier (light gray nodes in Fig. 1) will be checked.

Algorithm 1 Map region selection.

```
1: max_square_SL ←sort_boundary_nodes();          ▷ square_SL: square side length.
2: function SEARCH_MAPPING_REGION()
3:     square_SL ← max(sqrt(task_num), max_square_SL);
4:     region_found ← 0;
5:     while (region_found ≠ 1)and(square_SL > 0) do
6:         for Each unchecked boundary_node do
7:             if square_SL ≤ square_SL(boundary_node) then
8:                 Count available_nodes_num in the square;
9:                 Add available nodes in the square to the available_node_list;
10:                if available_nodes_num == task_num then
11:                    region_found ← 1; break;
12:                else if available_nodes_num < task_num then
13:                    region_found ← pick_nodes_in_frontier();
14:                    if region_found == 1 then return success;
15:                    else
16:                        Clear available_node_list;
17:                        available_nodes_num ← 0;
18:                    end if
19:                end if
20:            end if
21:            Mark checked boundary nodes;
22:        end for
23:        square_SL − −;
24:    end while
25:    if region_found ≠ 1 then return failed;
26:    end if
27: end function
```

Algorithm 2 Pick nodes in the frontier.

```
1:  function PICK_NODES_IN_FRONTIER()
2:      while available_nodes_num < task_num do
3:          nodes_num_to_find ← task_num − available_nodes_num;
4:          discover available nodes in frontier;
5:          Count available_nodes_in_frontier;
6:          if available_nodes_in_frontier > 0 then
7:              if available_nodes_in_frontier ≤ nodes_num_to_find then
8:                  available_nodes_num ← +available_nodes_in_frontier;
9:                  Add available nodes to the available_node_list;
10:             else
11:                 Pick nodes_num_to_find available nodes from the frontier
12:             end if
13:         else
14:             return 0;
15:         end if
16:     end while
17:     return 1;
18: end function
```

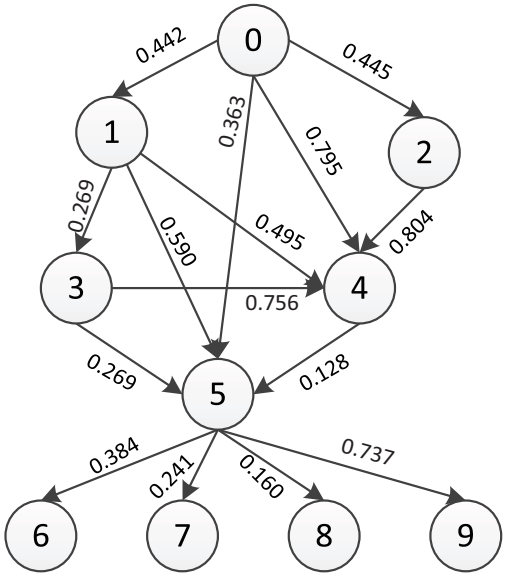


Figure 6.1: An application to map example.

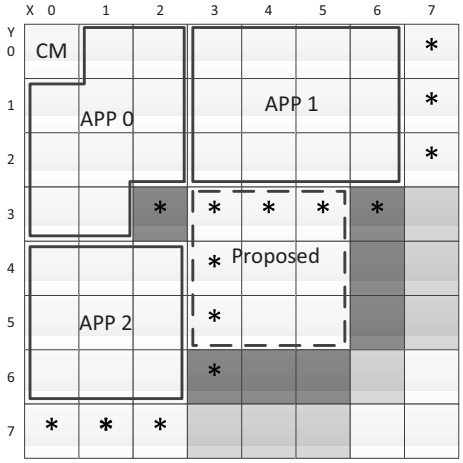


Figure 6.2: A NoC architecture example.

If there are not enough available nodes around the squares with a side length equal or larger than the target one, we reduce the target square side length by 1 and search again (line 23 in Algorithm 1). Note that the already checked nodes will not be checked. The searching steps are repeated until a contiguous near square region is found. In case the algorithm is failed, the CM may choose to map the application later after other application(s) is/are finished, or to map it into multiple unconnected regions. In this chapter, we choose for the first option.

The mapping region for the task example in Fig. 1(a) is illustrated in Fig. 6.3(a). The remained link bandwidth is mentioned on each link. Note that the link from $(2, 5)$ to $(2, 4)$ is already utilized by APP 2.

6.4.2 Task Mapping

After the mapping region is found, the tasks are mapped onto the nodes with Algorithm 3 and 4. The algorithm is backtracking based to identify the best mapping solution.

The first task to map is the one with the maximum total traffic load, e.g., t_4 in APP 3. Such tasks have the maximum number of edges and/or the edges have high traffic load. The first task is followed by its neighbor tasks which are sorted in the descending sequence of their total traffic load, and so on with the rest. In this way, we guarantee that the tasks having tight connection, i.e., small hop counts and high communication volume, are mapped with priority.

Algorithm 3 Map application to the selected region.

```
 1: function MAP_APPLICATION()
 2:     λ ← initial_value;
 3:     sort_tasks();
 4:     while map_success ≠ 1 do
 5:         mapped_task_num ← 0;
 6:         backtrack[task_num] ← all 0;
 7:         map_success ← 1;
 8:         previous_map_success ← 1;
 9:         while mapped_task_num < task_num do
10:             task_to_map = sorted_task_list[mapped_task_num];
11:             if previous_map_success == 1 then
12:                 backtrack[mapped_task_num] ← 0;
13:             else
14:                 backtrack[mapped_task_num] + +;
15:             end if
16:             backtrack_num ← backtrack[mapped_task_num];
17:             task_map_success ← map_task(task_to_map,backtrack_num);
18:             if task_map_success == 1 then
19:                 mapped_task_num + +;
20:                 previous_map_success ← 1;
21:             else
22:                 if mapped_task_num == 0 or max_backtrack_count
    then
23:                     map_success ← 0;
24:                     break;
25:                 else
26:                     mapped_task_num − −;
27:                     previous_map_success ← 0;
28:                 end if
29:             end if
30:         end while
31:         if map_success == 1 then return success;
32:         else
33:             λ = λ + 0.1;
34:         end if
35:     end while
36: end function
```

Algorithm 4 Map a task to the best node.

```
1: function MAP_TASK(task_to_map, backtrack_num)
2:     if bactrack_num == 0 then
3:         Find admissible nodes from unmapped_nodes_list;;
4:         Sort_admissible_nodes(admissible_nodes_list);
5:     else
6:         Remove previous map result of task_to_map;
7:         Move the released node to unmapped_nodes_list;
8:     end if
9:     if admissible_nodes_num > backtrack_num then
10:        best_node ← admissible_nodes_list[backtrack_num];
11:        Map task_to_map to best_node;
12:        Add the best_node to mapped_nodes_list;
13:        return success;
14:    end if
15:    return failed;
16: end function
```

The sequence to map the APP 3 tasks is $\{4, 5, 0, 1, 3, 2, 9, 6, 7, 8\}$.

For each task to map, we search for candidate nodes in *unmapped_nodes_list*. For a task t_i, an admissible node N_a, i.e., a node where t_i can potentially run, should meet the following requirements:

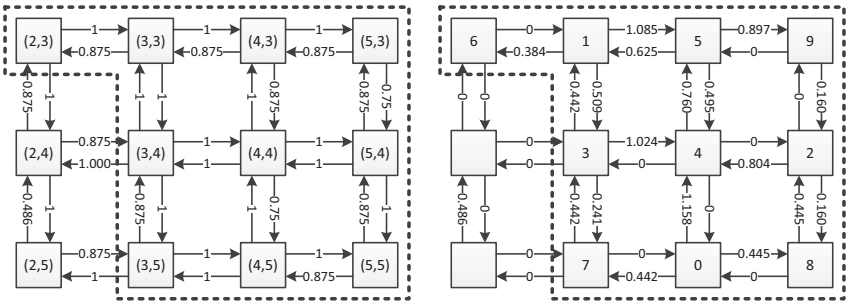
$$\sum r_{i,j} \leq \lambda \sum bw_{a,v},\ \forall t_j \in T,\ MD\{N_a, N_v\} = 1; \tag{6.6}$$

$$\sum r_{j,i} \leq \lambda \sum bw_{v,a},\ \forall t_j \in T,\ MD\{N_a, N_v\} = 1; \tag{6.7}$$

$$r_{i,j} \leq \lambda bw_{u,v} - \sum_{l_{u,v} \in e_{k,h}} r_{k,h},\ \forall t_j \in T_m,\ \forall u, v \in N. \tag{6.8}$$

Where $bw_{a,v}$ and $bw_{v,a}$ are the bandwidth of an output and input link of N_a, respectively. The links that will not be utilized by any task, e.g, the north link of node $(3, 3)$, are not considered in (6.6) and (6.7). In (6.8), T_m are already mapped tasks, and $e_{k,h}$ is any communication edge that make use of $l_{u,v}$ with a FIR of $r_{k,h}$. Note that λ can be larger than 1.

For the first task, its admissible nodes are sorted in the descending sequence of their total link bandwidth. For each subsequent task, the admissible nodes are sorted in the ascending sequence of the their CeMD to the mapped neighbor tasks. If two nodes have the same CeMD value, the one with more free neighbor nodes is preferred. According to the backtrack count, the task select one



(a) Mapping region, **(b)** Mapping result.

Figure 6.3: Map tasks into the target region with loose factor $\lambda = 1.20$. Link bandwidth is illustrated in (a), link traffic load is illustrated in (b).

node from the sorted *admissible_nodes_list* (line 46) and move the node to the *mapped_nodes_list*.

If a task t_i cannot find any admissible node, its mapping fails and the previous successfully mapped task will be reallocated to the next admissible node (line 14) and return the current one to the *unmapped_nodes_list*; Such mapping steps are repeated until all tasks are successfully mapped into the target region. In case all admissible nodes are tried for each task but no solution was found, the loose factor λ is increased and the mapping procedure restarted. We note here that the λ can also be decreased for applications with low traffic load to achieve more balanced traffic distribution. The step length to increase or decrease λ is also adjustable. The APP 3 mapping result illustrated in Fig. 6.3(b) was reached for a loose factor $\lambda = 1.20$.

Given that we rely on a backtracking strategy, the complexity of our proposal is obviously higher than the one of other state of the art heuristics. We note that it can be reduced by setting a higher initial λ value or by limiting the maximum backtrack count (line 22), e.g., $10 \times task\_num$.

6.5 Evaluation

In this section, we utilize the metrics introduced in Section 2 to evaluate our proposal and compare it with the incremental (INC) approach in [21] and the CASqA method in [34]. Note that CASqA searches for the first node with the SHiC [33] scheme.

The experiments are performed on our mixed language simulation platform

(processor nodes are implemented in C and the NoC in Verilog HDL). The NoC size is 8×8, each NoC router is divided into 3 stages, the NoC frequency is 1GHz, and the fault-free link data width is 32 bits. Partially defected links are utilized with the FS strategy. For the sake of simplicity, totally broken links and routers are not considered and packets are routed with the XY routing algorithm.

All heuristics are evaluated on both synthetic and real applications and for the sake of fair comparison, the applications enter and leave the MPSoCs in the same sequence in in all experiments. As long as enough idle nodes exist in the MPSoC, an application is injected, with the requirement that applications are always mapped into contiguous regions.

6.5.1 Mapping Quality

We first evaluate the heuristics on 100 TGFF [26] generated applications. Each application has 5 to 15 tasks and each task has up to 4 input and output communication edges. The FIR of each edge is randomly distributed in the range of 0.02 to 0.2 flits/node/cycle, and the initial λ is 0.3. The task mapping results of different heuristics, normalized to that of our proposal, are presented in Table 6.1. The NoC contains 5% randomly distributed broken wires and thus 80% links have narrowed bandwidth.

To evaluate the effectiveness of the CeMD metric, we also simulated the case when CeMD is replaced with Path Load (PL) [15] in Algorithm 1 and 3. In Table 6.1, prop.CeMD and prop.PL means that the CeMD and the PL metric is utilized in our proposed algorithm, respectively, and prop.CeMD* and prop.PL* correspond to $\lambda = 0.6$.

We can observe that prop.CeMD achieves better mapping results than all the

Table 6.1: Mapping quality for synthetic benchmarks.

	ALL	SLLD	AWeMD	NMRD	latency
prop.CeMD	1.00	1.00	1.00	1.00	28.64 / 1.00
prop.PL	1.05	1.07	1.04	1.00	29.46 / 1.03
prop.CeMD*	1.10	1.32	1.03	1.00	34.31 / 1.20
prop.PL*	1.19	1.54	1.06	1.00	36.44 / 1.27
CASqA	1.22	1.62	1.16	1.02	41.46 / 1.45
INC	1.25	1.82	1.19	1.12	46.30 / 1.62

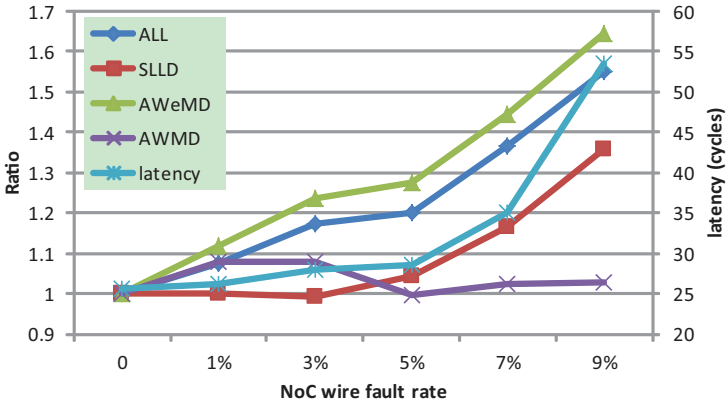


Figure 6.4: prop.CeMD mapping quality at different NoC wire fault rate. Results are normalized against the fault free case.

other counterparts. When compared with non-backtracking based heuristics, i.e., CASqA and INC, prop.CeMD achieves at least 22% lower ALL, 62% lower SLLD, and 45% shorter average packet transmission latency. This is because by considering the link bandwidth limitation and backtracking, the tasks and nodes are better matched and thus the traffic load is more evenly distributed. Balanced traffic load helps to reduce the congestion and thus shorten the average packet transmission latency. It is worth to note that because both our proposal and CASqA map applications into contiguous near square regions, they achieve similar NMRD values.

The only difference between CeMD and PL is that CeMD takes the amount of remained link bandwidth into account while PL does not. The SLLD of prop.PL is only 7% higher than that of prop.CeMD when $\lambda = 0.3$, but becomes 17% higher when $\lambda = 0.6$. Indeed, tight bandwidth constraint, i.e., low λ value, can efficiently reduce the max link traffic load. However, when the bandwidth constraint is loose, i.e., high λ, a mapping solution can be reached with few or even no backtracking steps, without exceeding the bandwidth limitation. In such case PL just selects the node have low path traffic load, while CeMD can also distinguish the path with more available bandwidth which means low ALL.

We further run prop.CeMD and CASqA on NoC platforms with different wire fault rates and illustrate the mapping results in Fig. 6.4 and Fig. 6.5. As the NoC wire fault rate increases, the average link bandwidth decreases and has higher variation, and thus it is getting harder to evenly distribute the NoC traffic load. Consequently, the mapping quality decreases and the packet transmis-

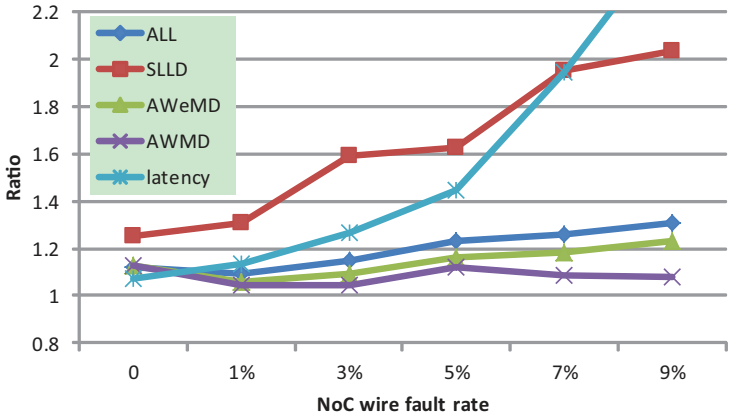


Figure 6.5: CASqA to prop.CeMD mapping quality ratio at different NoC wire fault rate.

sion latency becomes longer. The ascending lines in Fig. 6.5 indicate that the mapping quality of CASqA decreases faster than that of our proposal. When the NoC wire fault rate is high CASqA cannot properly balance the traffic load, consequently some narrow links are easily saturated and thus the packet transmission latency increases quickly.

6.5.2 Loose Factor

In this section, we study the loose factor impact on the mapping quality. To do so, we increase the loose factor from 0.2 to 1.0 when mapping the applications. The mapping results are illustrated in Fig. 6.6 where the CASqA mapping quality is also illustrated as a reference.

We can observe that almost the same mapping quality is achieved when λ is 0.2 and 0.3. This is because the link bandwidth constraint is too tight, and λ has to increase for every application to achieve a mapping solution. As λ keeps on increasing, the NoC traffic load becomes uneven and the mapping quality drops, which leads to higher packet transmission latency. When $\lambda \geq 0.6$, all applications can be mapped onto the selected region without backtracking. Although there are still differences in the mapping results due to the change of λ, the ALL, AWeMD, and packet latency do not exhibit obvious increase any more. However, larger λ decrease the CeMD difference of the admissible nodes, the influence of CeMD to the mapping quality is weakened and thus the link load deviation still has moderate increase. It is worth to note that

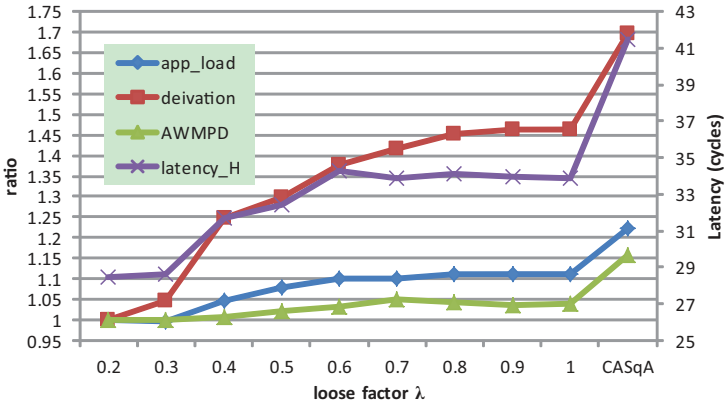


Figure 6.6: Network latency for different λ values. Results are normalized against the $\lambda = 0.2$ case.

because both the traffic load and the connection relationship among the tasks are considered in the task mapping sequence, tasks that have tight connection, i.e., low hop counts and high traffic load, are always mapped close to each other, the increase of AWeMD is less than 5% as λ changes.

6.5.3 Real Applications

We also evaluated our task mapping heuristic with four video processing applications [8]: Video Object Plane Decoder (12 tasks), MPEG-4 decoder(12 tasks), Picture-In-Picture (8 tasks), and Multi-Window Display (12 tasks). The 4 applications are repeated 5 times with random sequence, thus they may be mapped to different NoC regions. The loose factor of each application is decided according to the maximum edge FIR in prop.CeMD and prop.PL. The mapping results in Table 6.2 indicate that our proposal outperforms the state of the art counterparts in a similar way it did for synthetic applications.

Table 6.2: Mapping quality for video applications.

	ALL	SLLD	AWeMD	NMRD	latency
prop.CeMD	0.063	0.047	1.85	1.05	27.46
prop.PL	0.066	0.520	1.90	1.05	29.67
CASqA	0.081	0.060	2.22	1.09	33.18

6.6 Conclusion

In this chapter, we proposed a link bandwidth aware dynamic task mapping algorithms for 2D homogenous Multi-Processor Systems-on-Chip. The backtracking strategy is employed to ensure the maximum link traffic load does not exceed a limit determined by the link bandwidth and a loose factor. The Congested extended Manhattan Distance can further balance the traffic load by selecting admissible nodes with not just low traffic load but also more available bandwidth. Experimental results on synthetic and real video processing applications indicate that when 5% NoC link wires are broken, our proposal achieves at least 16% communication cost and 45% average packet latency reduction than that of counterpart task mapping heuristics.

In 2D NoCs based MPSoCs, all nodes resident on the same planar and there are high probabilities that communication edges of different tasks are overlapped that the NoC congestion increases. The emerging of 3D ICs opens a new path to 3D NoC based MPSoCs which can provide more task mapping solutions. However, novel 3D NoC infrastructures that can better exploit the Through Silicon Vias (TSVs) low latency benefits while improve their manufacturing yields are still expected. In the next chapter, we present our solutions to address these issues.

Note. The contents of this chapter is based on the the following paper:

C. Chen, S. D. Cotofana, **Link Bandwidth Aware Backtracking Based Dynamic Task Mapping in NoC based MPSoCs**, International Workshop on Network on Chip Architectures (NoCArc), pp. 5-10, 2014.

7

Enabling Wormhole Switching and Tolerating Faults in 3D NoC Vertical Links

With the emerging of 3 dimensional (3D) IC stacking technology, various 3D NoC architectures have been proposed to improve the performance of 2D NoCs. As most 2D NoC principles can be applied to each silicon layer, the main challenge in 3D NoCs relates to the vertical links implementation and utilization. Among state of the art 3D NoC proposals, the 3D NoC-Bus hybrid structure can better exploit the benefit of negligible Through Silicon Vias (TSVs) delay by running the buses at a higher speed than the routers while reduce the amout of low manufacturing yield TSVs by letting multiple routers resident on one tier to share the same bus. However, data are vertically transmitted with the Packet Switching (PS) technology but not Wormhole Switching (WS) because implementing vertical WS in the conventional way requires a large amount of TSVs. While WS enables lower packet transmission latency and requires less silicon area cost than PS. In this chapter, we propose a a Bus Virtual channel Allocation (BVA) mechanism that enables vertical WS in 3D NoC-Bus hybrid systems. In each cycle, BVA assigns to at most one cross layer packet a free input Virtual Channel (VC) in its target router before the packet flits are injected into the vertical bus and WS transmitted to the target layer. Given that VC allocation is performed only once per packet per hop BVA can be implemented in such a way that it does not become a system bottleneck. We evaluate our proposal with both synthetic and PARSEC benchmarks. The experimental results indicate that when compared with conventional pipelined bus or TDMA bus based systems, implementing vertical WS can reduce the bus critical path length by at least 31%, diminish the average packet transmission latency by at least 22%, and save the area cost

and power consumption of the output buffers incident to the bus by 47% and 43%, respectively. To deal with potential transient and permanent faults in 3D NoCs, in this chapter we also discuss the application of our aforementioned fault tolerant techniques to detect and correct soft errors in bus VC allocators, to utilize partially faulty vertical buses, and to tolerate deactivated or totally broken vertical buses.

7.1 Introduction

In 2 dimensional (2D) chips, the NoC induced system performance enhancement is still limited due to several aspects [28, 37], e.g., restricted floor planning choices, large clock tree network, long packet transmission latency, difficult to integrate components that are produced with different technologies, etc. With the emerging of 3 dimensional (3D) IC stacking technology, various 3D NoC architectures have been proposed [82], bringing the following benefits [111]: i) much shorter global interconnect; ii) higher packing density and smaller footprint; iii) higher performance due to low data transmission latency; iv) lower power consumption; and v) support for mixed-technology chips. In 3D chips, silicon tiers are vertically stacked and connected with Through Silicon Vias (TSVs) [7].

According to how the NoC components are organized, the existing 3D NoC structures can be briefly classified into two groups. The first group implements a 2D NoC on each silicon layer and utilizes TSVs as traditional links to connect vertically adjacent routers, e.g., [37, 45, 112], or as buses to connect all routers in the same Z-pillar, e.g., [28, 62]. The other group implements true 3D routers to connect 2D or 3D Processing Units (PUs), e.g., [53, 67, 73]. Despite the architecture differences, the 3D NoCs are still composed by routers and links, thus most 2D NoC design principles are still applicable and the main challenge relates to the implementation and utilization of vertical links.

When compared with moderate size planar wires, TSVs exhibit extremely low data transmission delay, but suffer from low manufacturing yield [63]. Thus in 3D NoC designs one should exploit the benefit of negligible TSV delay while reducing the TSVs amount. Among state of the art 3D NoC proposals, the 3D NoC-Bus hybrid structure serves this purpose well. When the vertical links are implemented as buses, it is possible to run the buses at higher speed than the routers due to the low delay of TSVs and the simplicity of the bus structure. Moreover, the buses can be shared by multiple routers on each silicon layer to reduce the amount of TSVs. The bus can be accessed by the routers incident to

it with a Time Division Multiple Access (TDMA) strategy [62], or be pipelined to enable concurrent data transmission [28].

In the existing NoC-Bus hybrid systems, data are usually transmitted in the vertical buses with the Packet Switching (PS) technology. It is well known that, when compared with PS, Wormhole Switching (WS) requires less silicon cost and enables lower packet transmission latency [24]. In symmetric NoC systems, each router port is solely connected with one neighboring router thus the allocation of one output Virtual Channel (VC), i.e., an input VC in the downstream router, to a packet can be easily done, because the output VCs are only utilized by packets from the current router and their status can be actively maintained by the output port [27]. However, in a NoC-Bus hybrid system, a vertically traveling packet can be destined to any other layer, and an input VC in the UPDOWN port, i.e., the port connected with the bus, can be competed by packets from different layers. Thus, for each UPDOWN output port, maintaining the VCs status in the UPDOWN input ports in other layers and assigning each cross layer packet a free output VC becomes much more complicated. Due to this existing 3D NoC-Bus hybrid structures postpone the vertical package VC assignation for the moment when the packet reaches its target layer, i.e., PS instead of WS is utilized for data transmission over the vertical buses.

In this chapter, we propose a Bus VC Allocation (BVA) mechanism that enables vertical WS in 3D NoC-Bus hybrid systems. Because the VC allocation is performed only once per packet per hop, the possibility that multiple BVA requests are asserted along the same bus in the same cycle is low. Thus in each cycle, the BVA mechanism forwards at most one request to the package target router and picks a free input VC there. In this way the routing path is reserved, and the packet flits can be transmitted with the WS technique on the buses. The BVA mechanism is evaluated on a $4 \times 4 \times 4$ 3D NoC-Bus hybrid system with both synthetic and real benchmarks traffic. The experimental results indicate that when vertical WS is implemented, the bus critical path in the pipelined and TDMA bus based hybrid systems are shortened by 31% and 34%, respectively, and the average packet transmission latency are reduced by 22% and 24%, respectively. Moreover, the area cost and power consumption of the output buffer incident to the bus are reduced by 47% and 43%, respectively. To deal with potential transient and permanent faults in 3D NoCs, in this chapter we also discuss the application of our aforementioned fault tolerant techniques to detect and correct soft errors in bus VC allocators, to utilize partially faulty vertical buses, and to tolerate deactivated or totally broken vertical buses.

The rest of the chapter is organized as follows. Section 7.2 presents a brief survey related with the existing 3D NoC structures. Section 7.3 introduces the proposed Bus VC Allocation (BVA) mechanism. Section 7.4 evaluates our proposal and compares it with tightly related work. Section 7.5 discuss the strategies to tolerate faults in 3D NoC vertical links. Section 7.6 concludes the presentation.

7.2 Related Work

The most intuitive way to implement a 3D NoC is to simply stack 2D Mesh NoCs and utilize TSVs to connect vertically adjacent routers. Despite simplicity, such 3D symmetric NoC is not taking advantage of the negligible inter-layer TSV delay [110]. To reduce the TSV amount, Hwang et al. [45] propose to connect only a few number of TSV routers to the vertical link, while the rest just connect with routers on the same silicon layer. This strategy substantially diminishes the TSV count but the vertically connected routers can easily cause communication hot spots and become the system bottleneck.

In [53], Kim et al. propose to utilize TSVs as intra-router connections to implement the 3D crossbar in a 3D Dimensionally-Decomposed router structure. Although the energy-delay product characteristic is enhanced, this approach requires many TSVs, thus such designs are not really applicable for state of the art 3D stacking technology.

Noticing that a large proportion of the network traffic takes place between the Processing Units (PUs) and the closest cache memories in the same pillar [62], Feero et al. [37] proposed a ciliated 3D NoC architecture, in which the routers locate on only one layer or a small number of layers, and each router is connected with multiple PUs residing in its Z pillar but on different layers. Such design offers an advantage in terms of energy dissipation when traffic is localized within a pillar. However, it has much lower throughput than 3D symmetric NoC systems and requires a large amount of TSVs.

In [62], Li et al. utilize TSVs to implement dynamic TDMA (dTDMA) buses to achieve one hop data transmission from a source layer to any destination layer. To alleviate the dTDMA bus drawback that it can only be occupied by one source-destination pair at any given time, Ebrahimi et al. [28] proposed a High-performance Inter-layer Bus Structure (HIBS), they pipeline the bus to enable concurrent data transmission. In such NoC-Bus hybrid systems, each bus can be shared by multiple routers on each layer to reduce the TSV amount. The buses are operated at higher clock frequency to provide enough

data throughput and prevent that they become the system bottleneck.

Many other 3D NoC structures exist [82] but when compared with the NoC-Bus hybrid structure, they have drawbacks, e.g., large amount of TSVs, obvious system performance degradation, incompatibility with existing 2D NoC technologies, etc. Conversely, the existing NoC-Bus hybrid designs have the drawback that do not allow WS on the vertical buses, and thus rely on complicated bus structures and exhibit long packet transmission latency. In this chapter, we first solve this issue with a a BVA mechanism and then discuss the application of dependable 2D NoC design principles in 3D NoC systems.

7.3 VC Allocation Along Vertical Buses

The 3D NoC-Bus hybrid structure embedding the proposed BVA mechanism is illustrated in Fig. 7.1 where only 3 layers are depicted, for the sake of simplicity. We note that the scheme is general and can be applied for more silicon layers and that the BVA arbiter always locates in the middle layer to reduce the TSV number. The zones marked with different colors can run at different frequencies, thus the requirement for universal clock distribution throughout the 3D chip is released and each layer can work at its optimized speed. The BVA units only exist around buses, thus their clock zone only occupies a small portion of the silicon area.

On each silicon layer, a 2D mesh NoC is implemented. Each router in the NoC has 6 physical ports, i.e., one UPDOWN port connected with the bus, and 5 ports connected with the local PU and the 4 neighboring routers located on its layer. On each NoC layer data are transmitted WS wise. For packets whose next hop is on another silicon layer, their head flits wait in the UPDOWN buffer until a free input VC in the target UPDOWN port is assigned to them by the BVA mechanism. Only after that, the packet flits can be injected into the bus along with the target layer number and the assigned VC index (VCID).

7.3.1 Problem Description

In symmetric NoCs each router port is solely connected with one neighboring router. As illustrated in Fig. 7.2, the state of each output VC is actively maintained in the output port by a Finite State Machine (FSM). When a packet needs to be transmitted to another router, it first applies for an output VC by asserting the VC Allocation (VA) request. If the request won both the local

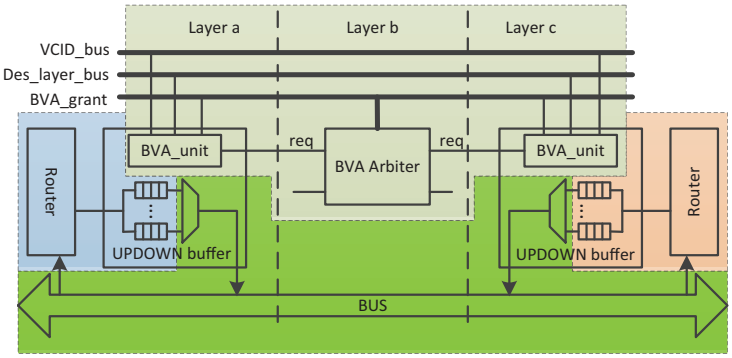


Figure 7.1: NoC-Bus hybrid system structure. The BVA arbiter locates in the middle layer and the colored zones can run at different clock frequencies.

and the global arbitration, a free output VC is picked from the free VC list and assigned to the packet.

With this conventional VA strategy, a routing path can be reserved by the head flit without waiting until the entire packet is received, which is time efficient and can operate with small input buffers, as only few flits are locally stored instead of integral packets. Such VA mechanism can be easily implemented in 2D NoCs as the output VCs are only utilized by packets from the local router. However in 3D NoC-Bus hybrid systems, a packet can be destined to any other layer, and an input VC in the UPDOWN port can be competed by packets from any other layers. For each UPDOWN output port, maintaining the VCs status in UPDOWN input ports in other layers and assigning each cross layer packet a free output VC becomes much more complicated.

In the existing 3D NoC-Bus hybrid structures, the packets are usually buffered in the UPDOWN buffers (see Fig. 7.1) before they are injected into the bus. As the matter of fact, a bus and the UPDOWN buffers incident to it can be considered as belonging to a virtual 3D router, i.e., the UPDOWN buffers are actually the input ports of the virtual 3D router, and the bus works as the crossbar. Implementing the aforementioned conventional VA in the virtual 3D router requires a large amount of TSVs. As illustrated in Fig. 7.2, each input port needs to send p request wires to the p output ports, and each output port needs to send p grant wires to the p input ports and $\log_2(v)$ wires to broadcast the assigned VCID, where v is the number of VCs. Moreover, each output port needs 1 extra signal to inform the input ports whether free output VCs exist. In an n-layer 3D NoC-Bus hybrid systems this implies that $2n^2 + n\log_2(v) + n$ TSVs are required and, as demonstrated in the next section, the proposed BVA

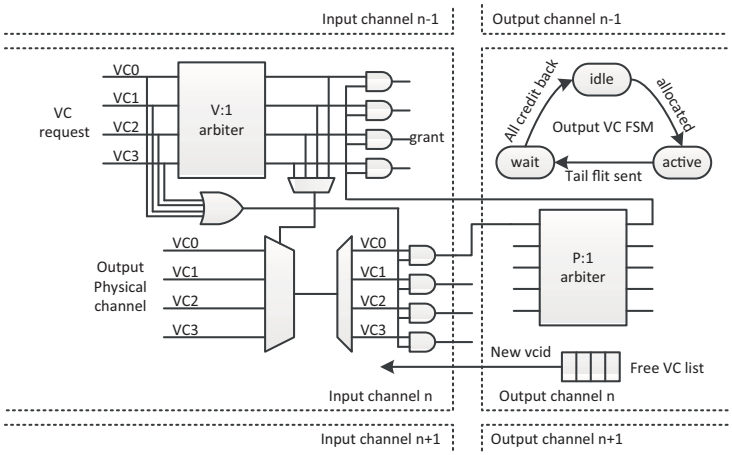


Figure 7.2: Conventional VC allocation mechanism. The number of VCs is v. The number of physical ports is p.

mechanism significantly reduce this value.

7.3.2 Bus VC Allocation Mechanism

The conventional VA mechanism assumes that multiple VA requests can be asserted by different input ports in the same cycle. However, at each hop, the VA application is done once per packet which means that for a packet length l the probability that a VA request is asserted at each port is $1/l$. Considering that packets do not arrive at the same port continuously, the actual probability is much smaller, thus the chance that multiple VA requests are asserted by different ports in the same cycle is also very small. In view of this analysis, we choose a BVA mechanism that assign free downstream input VC to only one packet per cycle.

The proposed BVA scheme is depicted in Fig. 7.3 and the associated timing diagram in Fig. 7.4. In each UPDOWN input port, an 1-bit *free_vc_exist* signal is asserted when at least one idle input VC exists, and sent to all the other routers along the bus. When a packet is destined to another layer, it is forwarded to the UPDOWN buffer, where if the *free_vc_exist* signal from the target router is high, its head flit asserts the VA request ① in Fig. 7.3 and 7.4. If multiple VCs are implemented in an UPDOWN buffer, each VC has the possibility to assert the VA request. The BVA request is generated by OR-ing the VA requests ②. The VA requests are sent to the local arbiter and the BVA request is

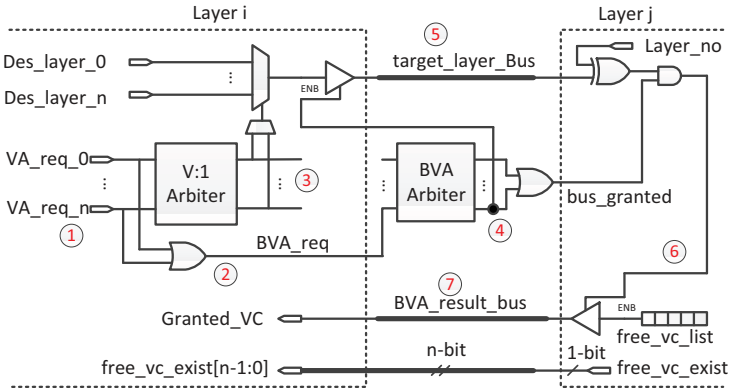


Figure 7.3: The proposed BVA scheme.

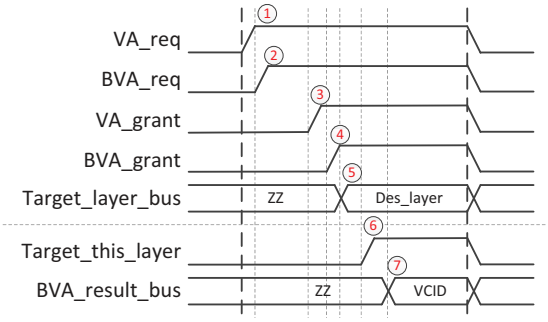


Figure 7.4: Timing diagram of the BVA mechanism.

sent to the BVA arbiter. After a certain delay the arbitration results are generated and as the BVA request and grant signals have to traverse several silicon layers, a packet receives the local *VA_grant* ③ earlier than the *BVA_grant* ④. The granted packet places its target layer number on the *Target_layer_bus* ⑤ while each UPDOWN input port monitors the BVA arbitration results and the *Target_layer_bus*. On the target layer of the granted BVA request ⑥, the UPDOWN input port chooses one idle VC from its free VC list and places the VCID on the *BVA_result_bus* ⑦. The *free_vc_exist* signal is also updated according to the remained free VCs. The source router reads the VCID from the *BVA_result_bus* and stores it into a dedicated register.

The number of TSVs (N_{TSV}) required by the BVA mechanism is calculated by (7.1), where n is the number of silicon layers, and v is the number of VCs in each physical channel. We note that the n *free_vc_exist* signals, the 1-bit *bus_granted* signal, the *Target_layer_bus* (width is $\log_2(n)$), and the

BVA_result_bus (width is $\log_2(v)$) are penetrating TSVs, while the n *BVA_req* signals and the n *BVA_grant* signals are just half way duet to the fact that the BVA arbiter is always placed in the middle of the 3D stack. For example, in a 4 layer 3D NoC-Bus hybrid system with 4 VCs per physical channel, the BVA requires only 13 TSVs for each pillar while the conventional VA requires 42.

$$N_{TSV} = 2n + \log_2(n) + \log_2(v) + 1. \tag{7.1}$$

7.3.3 Bus Data Transmission Policy

After a free input VC in the target router is reserved, a packet can transmit its flits to it via the bus with the WS technique. Each flit is transmitted together with the target layer number and the assigned VCID, thus flits belonging to different packets can be processed by the bus indiscriminately. In symmetric NoC systems, the VC buffer depth is usually less than the packet length if WS is implemented, and the credit return mechanism is utilized to inform the upstream router whether free buffer slots exist in the input VC. To save the TSVs required by the credit return mechanism, we set the input VC buffer depth in the UPDOWN port to the packet length, thus there is no need to inform the source router how many flits can still be transmitted. This can also guarantee that all flits injected into the bus will be absorbed by their target routers. We note that the buffer depth in the other ports can be smaller than the packet length.

In the dTDMA bus [62], a time slot is allocated to the packet that won the bus arbitration until all its flits are transmitted which means that the integral packet must be reassembled before the request is asserted. Otherwise, the reserved time slot is wasted by waiting for the un-arrived flits. When the BVA mechanism is utilized a bus request is asserted by each valid flit and the bus arbiter allocates time slots to individual flits instead of entire packets, which means that flits from different packages can be transmitted whenever the bus is available.

In HIBS [28] each bus stage is actually implemented as a 3 port router and a cross layer packet is only assigned a free input VC in the target UPDOWN input port when it arrives at the target layer. If all input VCs are occupied already, the packet has to wait in the bus stage buffer and causes Head of Line (HOL) blocking in the bus. A non blocking scheme was proposed to partially solve this issue by enabling the transmission of single hop packets when multiple hop packets are blocked, or vice versa, the blocking still can happen as it is possible that both kinds of packets are blocked. This scheme

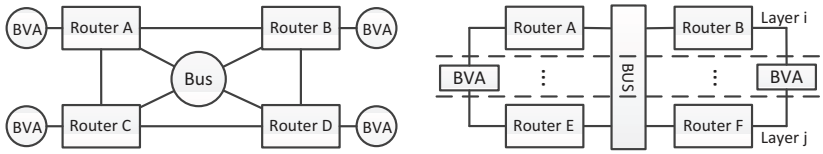


Figure 7.5: Cluster Mesh Inter-layer topology.

also requires that each bus stage must be able to store at least 2 integral packets for both upward and downward data flow direction. When the BVA mechanism is embedded, all flits are guaranteed to be absorbed by the target router, thus the HOL blocking is totally removed. Moreover, each bus stage just need to store several flits as the flits from different packets are equally processed by the bus. Thus the logic area cost of each bus stage is significantly reduced.

The buses can be shared by multiple routers, e.g., 2 or 4, on each layer to reduce the number of TSVs. As BVA grants only 1 BVA request in each cycle, we prohibit the inter-layer or intra-layer diagonal data transmission to maintain the simplicity and efficiency of the BVA mechanism. Each pillar still has a dedicated BVA arbiter, and when a router transmits flits via the bus, the target router must be right above or below it. Take the Cluster Mesh Inter-layer Topology (CMIT) [28], in which each bus is shared by 4 routers per layer, illustrated in Fig. 7.5 as an example, router A has to communicate with D and F via B or C, and B or E, respectively.

7.4 Evaluation

To put the implications of our BVA mechanism in a better practical prospective we embed it in TDMA and pipelined bus based 3D NoC-Bus hybrid systems labeled as TDMA_BVA and pip_BVA, respectively, and compare their figure of merit against that of dTDMA bus [62] and HIBS [28] based systems. We implemented $4 \times 4 \times 4$ NoCs with 4 VCs per physical port at RTL level by using Verilog HDL. The VC buffer depth is the same with the packet length, i.e., 8-flits, in the UPDOWN input ports and 4-flits in other input ports. The flit width and the link width in 3D symmetric NoC are all 32-bits. For the sake of fairness, the buses, either pipelined or TDMA based, are composed of 2 unidirectional data lanes, each is 32-bits wide in charge of the upward and downward data flow, respectively. In HIBS, the input buffer in each bus stage is able to store 2 packets for every data flow direction. While in pip_BVA, the

two Bus_FIFOs, upward and downward, in each bus stage are only set to 4-flits deep. The packets are transmitted by WS in each planar NoC and routed with the XYZ algorithm in the 3D NoC system.

The Routers and buses are synthesized using Cadence RTL Compiler with TSMC 45nm technology to estimate the critical path length, area cost, and power consumption. We assumed a TSV pad size of 3 μm with a 5μm pitch [63], and analysis with Cadence Virtuoso Spectre indicate a TSV delay of 20 ps per layer.

7.4.1 Critical Path Length

The critical path length of routers and buses in different NoC-Bus hybrid systems are illustrated in Fig. 7.6. We can observe that the BVA has shorter critical path than the routers. If the NoC layers number increases the BVA delay increases too and eventually surpasses the router delay. We note that given that BVA can work at different speed than the routers and buses this has no consequences on their implementation.

In the HIBS based system, each bus stage is actually a three port router and it makes use of VCs to partially solve the HOL blocking issue. When vertical WS is enabled, each bus stage just need to decide which of the two flits, one from the previous bus stage and one from the UPDOWN buffer, will be forwarded. Consequently, in pip_BVA, the critical path length of each bus stage is 31% shorter than that of HIBS, which means the bus can be twice faster than the routers.

In the dTDMA bus based system, each cross layer packet is assigned a free input VC when its head flit arrives at the target router. The "free input VC exists" flag in the UPDOWN input port is also updated in the meanwhile. This flag is then utilized by the centralized bus arbiter to decide whether another bus request can be granted. Consequently the dTDMA bus has a long critical path. In TDMA_BVA, the arbiter grants a request just according to the request's current priority, thus the buses have shorter critical path than that of dTDMA and can be 1.6 times faster than the routers.

7.4.2 Synthetic Traffic

The performance of different 3D NoC systems at various Flit Injection Rates (FIRs) under random and localized traffic patterns is illustrated in Fig. 7.7 and Fig. 7.8. In the localized traffic, 50% of the packets are destined to the

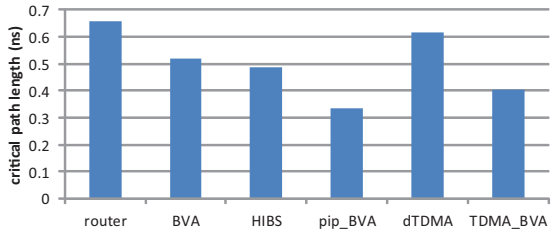
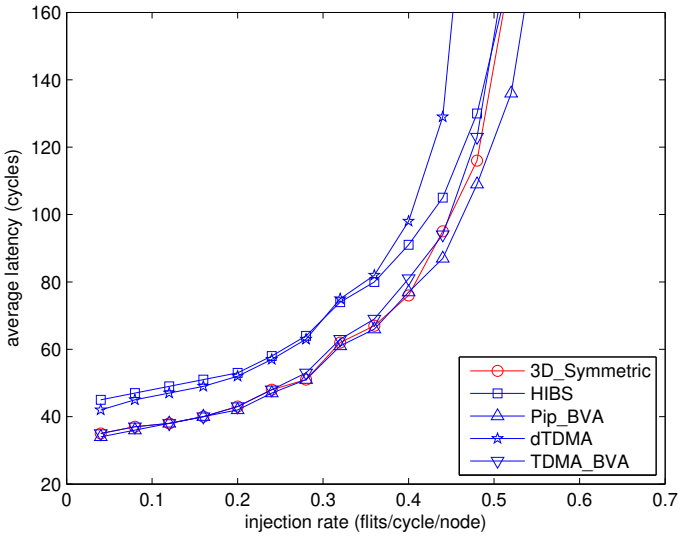


Figure 7.6: Critical path length of routers and buses.

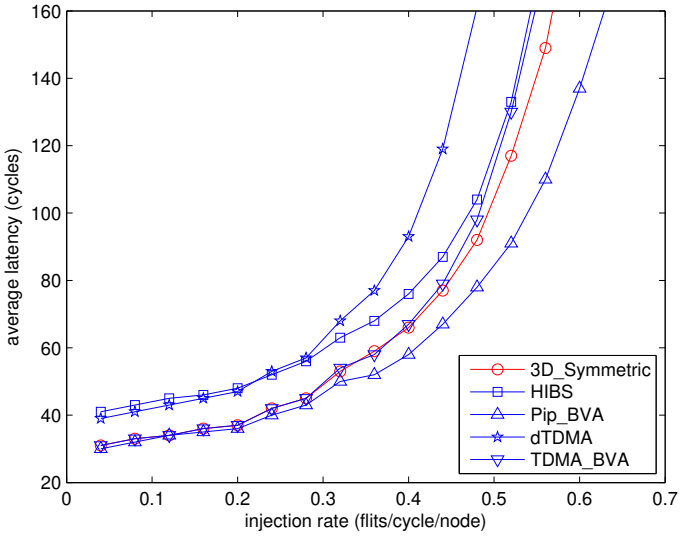
nodes in the same pillar, to simulate the practical case when task mapping is optimized [62]. The 3D symmetric NoC system embedding WS is also evaluated as a reference. The packet transmission latency is counted since the packet is generated in the source node till the tail flit is received by the destination node, i.e., the queuing time in the source node is included.

Our simulation results when the bus is not shared on each layer indicate that the average packet transmission latencies in HIBS and dTDMA bus based hybrid systems are on average 22% and 24% longer than that of 3D symmetric NoC system, respectively. This is because when WS is utilized in planar NoC and PS is utilized in the buses, extra delay is required to reassemble the integral packets in the UPDOWN buffer. When vertical WS is enabled, the extra delay is removed and both TDMA_BVA and pip_BVA can achieve similar or even better performance than the 3D symmetric NoC.

When each bus is shared by multiple routers, i.e., 2 and 4, per layer, the advantage of our proposal becomes more obvious (see Fig. 7.8). Note that due to the high traffic load, the buses in the hybrid systems saturate quickly when they work at the same frequency with the routers, especially when CMIT is implemented. However, according to the analysis in Section 7.4.1, the buses can be operated at higher frequencies, case in which we obtain substantial improvements. Note that although the maximum bus frequencies of the HIBS, dTDMA, and TDMA_BVA designs are only 35%, 7%, and 63% higher than that of the routers, we still evaluated their performance at 2x bus speed for comparison purpose. It is worth to note that pip_BVA always achieves the best performance, in terms of both average packet latency and saturation throughput, in all simulation contexts.

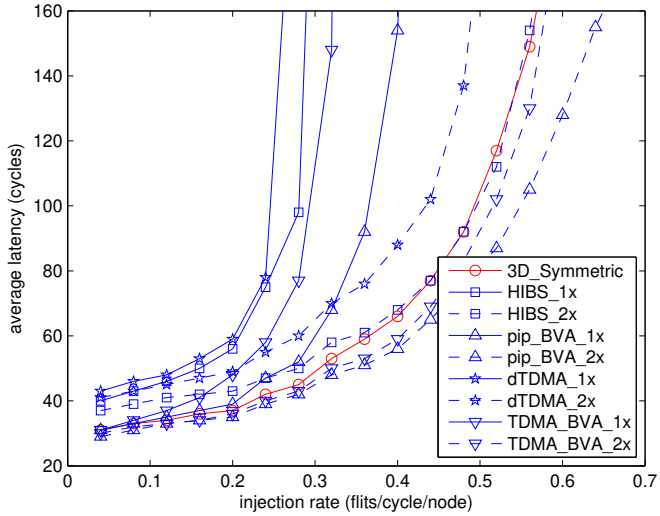


(a) Random traffic;

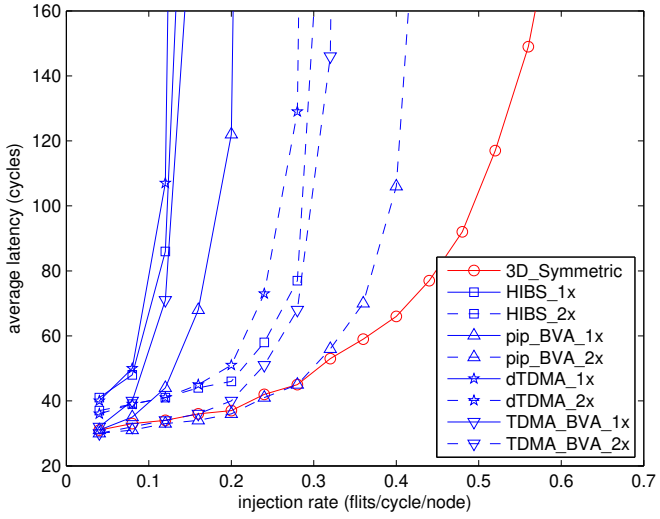


(b) Localized traffic;

Figure 7.7: Average packet transmission latency in different 3D NoC system when buses are not shared. The packet length is 8-flits.



(a) Bus is shared by 2 routers, localized traffic;



(b) Bus is shared by 4 routers, localized traffic;

Figure 7.8: Average packet transmission latency in different 3D NoC system when buses are shared. 1x, 2x means the bus frequency is 1, or 2 times higher than the NoC router frequency, respectively. The packet length is 8-flits.

7.4.3 BVA Efficiency

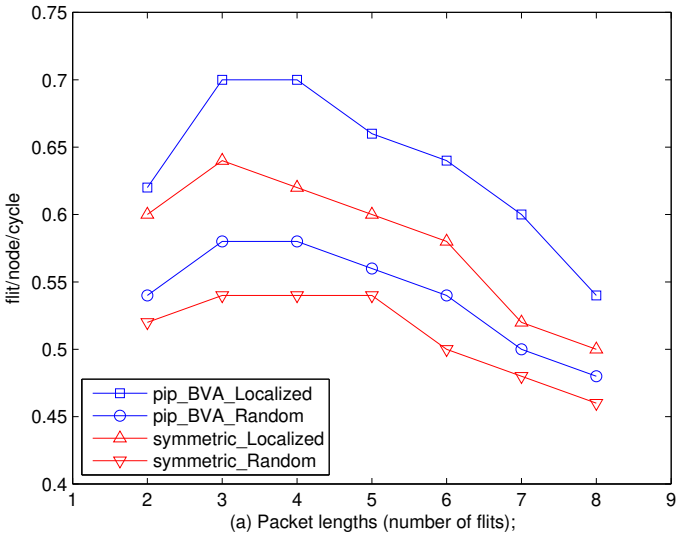
The proposed BVA mechanism grants only one BVA request from one silicon layer in each cycle. Thus the BVA efficiency is expected to decrease as the packet becomes shorter and the number of layers becomes higher. The impact of those issues on the pip_BVA design and the 3D symmetric NoC saturation throughput is illustrated in Fig. 7.9.

Against expectation, the experimental results indicate that the saturation throughput increases as the packet length decreases from 8 to 3. This can be explained by the fact that although shorter packets assert BVA request more frequently at the same FIR, they also release the VCs faster. The packet length has more system performance influence, e.g., when the packet length decreases from 3 to 2, the saturation throughput has a 11% and 7% reduction for random and localized traffic, respectively. We note that the same trend exists in the 3D symmetric NoC systems.

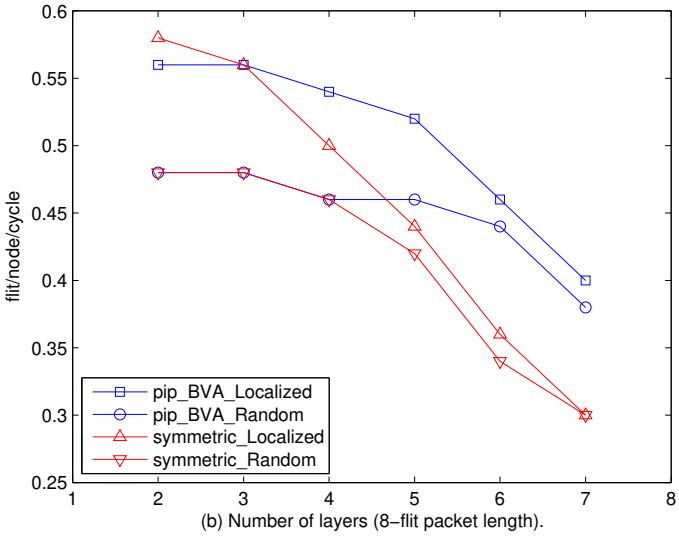
As expected, the system saturation throughput decreases as the silicon layers number increases. But the decreasing speed in 3D symmetric NoC systems is faster than that in pip_BVA. Thus the decreasing is mainly caused by the layers number increase but not the BVA mechanism. Note that the planar NoC size is always 4×4. This proves that the BVA is not the system performance bottleneck.

7.4.4 PARSEC Benchmarks

In this subsection, we evaluate our proposal with PARSEC benchmarks [9] traffic traces, which are recorded with the Netrace [43] tool on the M5 full system simulator [10]. We replay the traffic traces according to each packet time flag while maintaining the packets dependencies. The average packet transmission latencies for different benchmarks are illustrated in Fig. 7.10. The results indicate that when vertical WS is enabled, all 3D NoC-Bus hybrid systems provide similar packet transmission latency with the 3D symmetric NoC system, even when CMIT is implemented. Without BVA, the latencies in the HIBS and dTDMA bus based systems are on average 18% and 13% longer, respectively.



(a) Saturation throughput VS packet length;



(b) Saturation throughput VS layers number

Figure 7.9: The system saturation throughput at different packet length and layers number. The packet length is 8-flits.

7.4.5 Area and Power

The area cost and power consumption of routers and buses are presented in Table 7.1. The bus power consumption is derived when buses and routers work at the same frequency. Note that the routers are implemented in the same way in different NoC-Bus hybrid systems.

In each UPDOWN buffer, 4 VCs are implemented in our experiments. For HIBS and dTDMA bus based system, integration packets need to be reassembled in the UPDOWN buffer before they are injected into the bus. While when vertical WS is enabled, such requirement is released and the buffer depth can be reduced to, for example, 4-flits. Consequently the area and power cost of the UPDOWN buffer is reduced by 47% and 43%, respectively. For each data

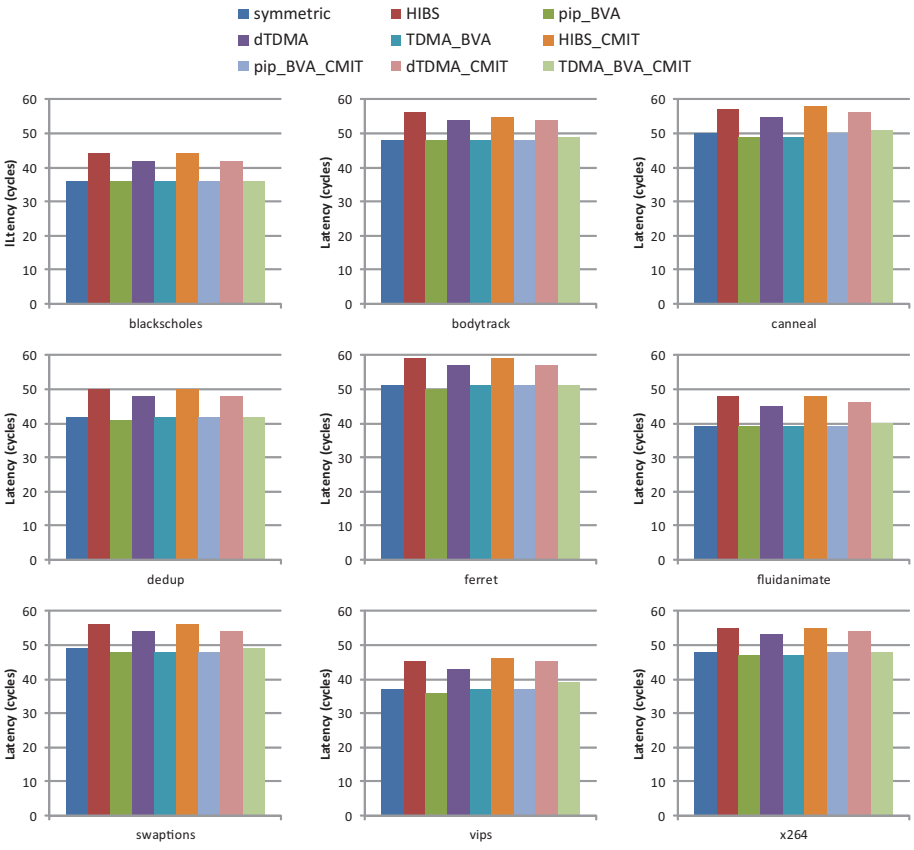


Figure 7.10: Average packet transmission latency of PARSEC benchmarks. The buses work at the same frequency with the routers.

Table 7.1: Area and power of router and bus stage in different 3D NoC systems.

		Power (mW)	Logic area (μm^2)	TSV#/area
Router		43.18 / 100%	68207 / 100%	–
UPDOWN buffer	PS	10.28 / 24%	13701 / 20%	–
	WS	5.48 / 13%	7854 / 12%	–
Bus stage	HIBS	13.06 / 30%	15740 / 23%	76 / 1900
	pip_BVA	2.86 / 7%	3644 / 5%	91 / 2275
	dTDMA	–	–	86 / 2150
	TDMA_BVA	–	–	97 / 2425

flow direction, the HIBS bus stage is required to register at least 2 packets to partially solve the HOL blocking issue. While when BVA is implemented, the bus stage FIFO just need to register several flits, e.g., 4 flits in our case, and the HOL blocking is completely removed. Thus the implementation cost is also significantly reduced.

When the bus is not shared on each silicon layer, enabling vertical WS requires 15 and 11 more TSVs than the original HIBS and dTDMA based design in each pillar, respectively. However, the area overhead induced by the TSVs is still negligible even they are much bigger than planar wires.

7.5 Fault Tolerance in 3D NoCs Vertical Links

At a certain fault rate, reducing TSVs amount can efficiently diminish the amount of faults in the vertical links, and thus reduce the fault tolerance capability requirements to the 3D NoC system. However, transient and permanent faults can still happen and should be tolerated to avoid severe system performance degradation. In the 3D NoCs where a 2D NoC is implemented on each silicon layer, the previously proposed fault tolerant strategies can be easily applied with small adjustment which takes the vertical links into consideration.

7.5.1 Transient Faults

Transient faults in the data path may flip data bits. Such soft errors can be easily detected and corrected with Error Correcting Codes (ECCs). In the case of uncorrectable errors, the contaminated flits or packets are retransmitted. It is even proved that only implementing the soft error resilience techniques at the

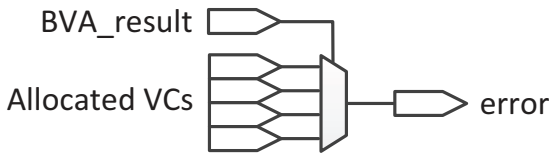


Figure 7.11: Structure to detect erroneous BVA results.

inter-die level with interleaved Hamming codes is enough to achieve highly reliable communication in the 3D NoC systems [77].

In 2D NoCs, the Error Detection and Correction (EDC) is usually performed Hop-By-Hop (HBH), which generally requires 3 cycles to start a flit retransmission, i.e., the flit is transmitted to the downstream router in the first cycle, the uncorrectable error is detected in the second cycle, and the retransmission request is sent back to the upstream router in the third cycle. In 3D symmetric NoCs, EDC can be implemented in the same way because vertical links are identically utilized as planar links. In 3D NoC-Bus hybrid systems, when TDMA buses are implemented, each bus needs one extra TSV to broadcast the retransmission requests, and the relative source and destination pair can be determined by register the bus arbitration results for 3 cycles. When the buses are pipelined, the EDC should be implemented at each bus stage but not just in the UPDOWN input ports because a flit may need multiple cycles to traverse the vertical bus.

Transient faults can also happen in the control plane, i.e., Routing Computation (RC), VA, Switch Allocation (SA), BVA, and Bus Arbitration (BA). In a router, RC, SA, and VA are related with physical ports in the same layer and thus the errors in their results can be detected and recovered with the method proposed in Chapter 3. When vertical WS is implemented, the BA is only utilized to decide which flit can be transmitted, thus an erroneous BA result only assigns the bus to another bus access request without cause any failure.

Similar with VA results, when soft errors happen in BVA results, the output VC can become (1) a wrong output VC, which does not exist or is not an eligible VC according to the RC results, or (2) an eligible output VC which is already assigned to another packet. In Chapter 3, we have proposed an I-IVAD method to detect type (1) errors, an O-DVAD method to detect type (2) errors, and relative mechanism to recover from the errors. The type (1) BVA errors can still be detected by means of the I-IVAD method and be recovered in the same cycle. However, although type (2) BVA errors can be detected by means of the O-DVAD method, they are hard to be recovered because a head flit in the

UPDOWN output ports may arrive at the target router several cycles after an output VC is granted to it, during which period more BVA errors can happen and the error recovery becomes more complicated.

Alternatively, we propose to check the correctness of the BVA results with the structure illustrated in Fig. 7.11 in the target router before the assigned VCID is sent back to the request initiator. Similar with the I-IVAD method, we maintain a list of allocated VCs in each UPDOWN input port. By simply check if the VC provided by the BVA is already allocated to an packet, duplicated VC allocation can be avoided. The drawback of this method is that the soft errors happen on the BVA_result_bus cannot be detected. As a compensation, the BVA results can be transmitted together with a parity check bit, so that a one-bit error can be detected at both the BVA request's source and target routers. Upon the detection of BVA errors, the BVA result is discarded and the request is asserted again to apply for another valid free output VC.

7.5.2 Partially Defected Vertical Buses

Although the absolute number of broken TSVs can be reduced by sharing each vertical link with multiple routers on each silicon layer or even implement serial vertical interconnects [78, 94], TSVs can still be defected and must be tolerated to maintain the functionality of the 3D NoC system.

Prefabricated spare TSVs can be utilized to replace broken TSVs during the manufacturing process [40, 63, 69] or at run time [60, 61]. The packet rebuilding/restoring method proposed by Yu et al. [114] and the Configurable Fault-Tolerant Serial Link (CSL) method proposed by Pasca et al. [76] do not rely one spare TSVs but they are actually different practice of the spare wire replacement method. The drawback of such solutions are that they induce complicated control logic and thus large area overhead. Our Flit Serialization (FS) method and other Partially Faulty Link Utilization strategies proposed for 2D NoC systems, e.g., [72, 100], can all be utilized as low cost alternative methods to tolerate broken TSVs with the penalty of reduced vertical link bandwidth.

However, the FS methods cannot be directly applied to 3D NoC-Bus hybrid systems in which the routers access to the buses are interleaved. When vertical WS is implemented, a TDMA bus is allocated to flits from different routers in different cycles, and a pipelined bus stage is shared by flits from the previous bus stage and the local router in the round-robin sequence. For the other partially faulty link utilization methods, e.g., [72, 100], the flit transmission latency on a link is merely decided by the link fault level. While for the FS

method, the flit transmission latency is also affected by the number of continuously transmitted flits (refer to Equation 4.1). For example, when the link is divided into 4 sections and 1 section is broken, the average flit transmission latency is 1.67 and 1.33 cycles if 2 and 3 flits are continuously transmitted, respectively. Thus to improve the link bandwidth utilization efficiency, the bus arbiter should allow the granted request initiator to hold the bus until multiple integral flits are transmitted. For example, if 1 out of the 4 link sections is broken and 2 flits are waiting to be transmitted in an UPDOWN output port, the output port should hold the bus until both flits are transmitted; while if more than 3 flits are waiting to be transmitted, the output port should release the bus after 3 flits are transmitted to enable fair bus access among the routers in the same pillar.

7.5.3 Fault Tolerant Routing

Similar with planar links, a vertical link has to be abandoned when it contains too many broken TSVs. Moreover, when an UPDOWN input/output port is broken, the router cannot receive/transmit packets from/to another silicon layer. In both cases, the packets have to be detoured to reach the destination.

It is obvious that Fault Tolerant Routing Algorithms (FTRAs) that are proposed for 2D NoC can be applied to each 3D NoC silicon layer. Thus the challenge is to find the optimal cross layer misrouting path in awareness of the broken vertical links. Targeting at different fault patterns, numerous FTRAs for 3D NoCs have been proposed, e.g., [2, 79, 83, 86, 117].

When routers in the same pillar are connected by a pair of unidirectional links, i.e., upward and downward packets are transmitted via physically independent links, it is possible that the upward link is broken while the downward one is still functional, or vice versa. Similar with partially broken planar interconnects, the broken vertical links should be tolerated while the unpaired functional ones should be utilized. In such case, we propose to combine our UnPaired Functional link aware FTRA with the adaptive inter-layer message routing algorithm proposed by Rusu et al. [86]. In their proposal, each router is attached to two Vertical Node Trees (VNTs) which are rooted in the nearest routers that have upward and downward output links, respectively. Packets that are destined for other layers are first transmitted to the VNT root nodes and then be transmitted to the target layers. Their routing algorithm makes no assumption about the topology of the 2D layers and thus can be easily integrated with any other planar routing algorithms. As unpaired functional planar and vertical unidirectional links are efficiently utilized, the 3D NoC system's

performance can be degraded more gracefully.

7.6 Conclusion

In this chapter, we proposed a Bus Virtual Channel (VC) Allocation (BVA) mechanism to enable vertical Wormhole Switching (WS) in 3D NoC-Bus hybrid systems. Because the VC allocation is performed only once per packet per hop, the possibility that multiple BVA requests are asserted along the same bus in each cycle is low. Thus in each cycle, the BVA mechanism forwards at most one request to its target router and picks a free input VC there. In this way, a routing path is reserved by the head flit, and the next flits in the package can be transmitted with the WS technique on the buses. We evaluate our proposal with both synthetic and PARSEC benchmarks. The experimental results indicate that when compared with conventional pipelined bus or Time Division Multiple Access (TDMA) bus based systems, implementing vertical WS can reduce the bus critical path length by at least 31%, diminish the average packet transmission latency by at least 22%, and save the area cost and power consumption of the output buffers incident to the bus by 47% and 43%, respectively. To deal with potential transient and permanent faults in 3D NoCs, in this chapter we also discussed the application of our aforementioned fault tolerant techniques to detect and correct soft errors in bus VC allocators, to utilize partially faulty vertical buses, and to tolerate deactivated or totally broken vertical buses.

When multiple silicon layers are stacked to construct 3D ICs, thermal dissipation becomes a critical issue that affects the chips' and the 3D NoCs' dependability and thus should be extensively studied in our future research work.

Note. The contents of this chapter is based on the the following paper:

C. Chen, M. Enachescu, S. D. Cotofana, **Enabling Vertical Wormhole Switching in 3D NoC-Bus Hybrid Systems**, Design Automation and Test in Europe (DATE), pp. 507-512, 2015.

8

Conclusions and Future Work

In this dissertation, we have presented several designs to improve the dependability of Networks-on-Chip (NoC) at the architectural level by tolerating transient and permanent faults as well as efficiently utilizing still functional NoC components. We first introduced a low cost method to allow for correct flit transmission even when soft errors are occurring in the router control plane. Then we proposed a Flit Serialization (FS) strategy to tolerate broken link wires and to efficiently utilize the remaining link bandwidth. Within the FS framework heavily defected links whose fault levels exceed a certain threshold value are deactivated to diminish the congestion in their upstream routers. Moreover, we designed a distributed logic based routing algorithm able to tolerate totally broken links as well as to efficiently utilize UnPaired Functional (UPF) Links in partially defected interconnects. We also introduced a link bandwidth aware run-time task mapping algorithm to improve the mapping quality for newly injected applications in the MPSoCs. Last but not least, we discussed the application of aforementioned strategies in 3D NoC systems and proposed a Bus Virtual channel Allocation (BVA) mechanism to enable vertical wormhole switching to improve the performance of 3D NoC-Bus hybrid systems. All proposals are evaluated in our mixed language NoC simulation platform and their advantage over state of the art counterparts are proved by means of experimental results.

8.1 Summary

The contents and contributions of this dissertation are summarized as follows:

In **Chapter 1**, we discussed the necessity to implement NoC based MPSoCs in modern Ultra Large Scale Integration (ULSI) systems, state of the art ICs dependability issues and their corresponding NoC design challenges, highlighted

the dissertation contributions, and introduced the dissertation organization. We pointed out that efficiently utilizing still functional NoC components is as important as tolerating faults to enable graceful system performance degradation and improve the system dependability.

In **Chapter 2**, we introduced the essential NoC background knowledge by covering the aspects as NoC topology, routing algorithm, switching policy, and router architectures. We also present the mixed language NoC simulation platform we utilize to evaluate and validate the contribution described in this thesis, the strategies to inject synthetic traffic and real application traces into the NoC, and the NoC performance evaluation metrics.

In **Chapter 3**, We proposed a low cost method to tolerate soft errors potentially occurring in router control plane functional units, i.e., routing units, Virtual Channel (VC) allocators, and switch allocators. Rather than relying on a Triple Modular Redundancy based implementation of each functional unit, we chose to exploit the intrinsic redundancy available in the router hardware structures and signals. In essence we detect Routing Computation (RC) errors by comparing RC results from the local Routing Unit (RU) and idle RUs available at neighboring input ports. The RC results are recalculated in case errors are detected or neighboring RUs are not available. We detect errors in the VC Allocation (VA) and Switch Allocation (SA) results by checking if they are consistent with the correct RC results, each NoC resource is exclusively assigned to one request initiator, and each request initiator is allocated only one NoC resource. VA/SA errors are corrected by redoing the failed procedures and retransmitting the flits. Experimental results on an 8×8 2D NoC indicate that: (i) in the routing units, the proposed method requires 38% more silicon real estate than the Σ & Branch method when the XY routing algorithm is utilized, but it is more general and can be utilized in conjunction with other routing algorithms; and (ii) in the combined VA/SA units, the proposed method is simpler and more effective than state of the art counterparts. When compared with the Triple Modular Redundancy strategy, for similar error detection and correction capabilities, the proposed method can reduce the area and power overhead in routing units by 53% and 38%, respectively, and in combined VA/SA units by 45% and 46%, respectively. The average packet transmission latency is less than 5% higher than the one of the baseline router with no soft error detection/correction mechanisms even if the soft error rate is as high as 0.1 errors/router/cycle.

In **Chapter 4**, We proposed a Flit Serialization (FS) method to tolerate broken link wires and to effectively utilize the remained link bandwidth. The FS ap-

proach divides the links and flits into several sections, and serializes sections of adjacent flits to transmit them on all available fault-free link sections to avoid the complete waste of partially defective links. The proposed transmitter and receiver are transparent to the router such that their utilization is not constrained by the router architecture and implementation or network topology. Experimental results obtained on synthetic traffic and PARSEC benchmarks indicate that FS reduces the latency overhead significantly and enables graceful performance degradation when compared with related partially faulty link utilization proposals. It reduces area cost and power consumption by up to 29% and 43.1%, respectively, when compared with spare wire replacement methods, and can achieve lower area*power/saturation_throughput values than all state of the art link fault tolerant strategies. We also propose the link augmentation with one redundant section as a low cost mechanism to further increase the link dependability. Experimental results indicate that when 10% of the NoC wires are broken, adding a redundant section to each link can improve the NoC saturation throughput by 18%.

In **Chapter 5**, We introduced a strategy to differently treat partially faulty links that have different fault levels as follows: (i) links whose fault level is lower than a threshold are still utilized by means of the FS method, while (ii) Heavily Defected (HD) links whose fault levels exceed the threshold are deactivated and tolerated by means of a Fault Tolerate Routing Algorithm (FTRA). To this purpose, we determined the optimal link deactivation threshold by comparing the zero load packet transmission latency on the HD links and that on the shortest alternative path, and proposed a distributed logic based FTRA to tolerate broken links as well as to efficiently utilize the UnPaired Functional (UPF) links in partially defected interconnects. The basic fault pattern tolerated by the UPF link aware FTRA (UPF-FTRA) is a fault wall, which is composed of adjacent broken links with the same outgoing direction. Messages are routed around the fault walls along the misrouting contours of the broken links. The proposed Routing Algorithm (RA) requires at least 3 Virtual Channels (VCs) and dynamically reserve them to the detoured messages to avoid deadlock. Our experiments indicate that, for random and localized traffic patterns, we achieve an average saturation throughput 20% higher than the Solid Fault Region Tolerant (SFRT) RA, and 22% and 14% higher than the Ariadne routing table based RA, respectively. Simulation results with PARSEC benchmarks also suggest that UPF-FTRA provides much lower packet transmission latency than SFRT and Ariadne. Synthesis results with Synopsis Design Compiler and TSMC 65nm technology indicate that, embedding the proposed RA into a baseline router results in 9% area overhead, which is only

1% higher than that of SFRT and does not increase for NoCs with bigger size. Simulation results we obtained at various wire broken rate configurations indicate that we achieve the highest saturation throughput if 4- or 8-section links with a flit transmission latency longer than 4 cycles are deactivated.

In **Chapter 6**, We proposed a run-time task mapping algorithm, which takes both the path traffic load and link bandwidth variation into consideration and maps applications onto contiguous near convex NoC regions to reduce the internal and external congestion. We relied on a backtracking strategy to guaranty that the maximum link traffic load does not exceed a given limit determined by the link bandwidth and a loose factor. Note that the loose factor is employed to adjust the maximum percentage of link bandwidth that can be utilized. To evaluate our proposal we mapped synthetic and real video processing applications on partially defective 8×8 NoCs. The experiments indicate that our approach substantially outperforms equivalent state of the art task mapping heuristics when NoC defects are present, e.g., for 5% broken wires, we achieve at least 16% communication cost reduction and 45% shorter average packet transmission latency.

In **Chapter 7**, We proposed a Bus Virtual Channel (VC) Allocation (BVA) mechanism to enable vertical Wormhole Switching (WS) in 3D NoC-Bus hybrid systems. The BVA mechanism address this issue by assigning in each layer to at most one cross layer packet a free input VC in its target router before injecting the packet into the bus. In this way, a routing path is reserved by the head flit, and the rest of the packet flits can be WS transmitted through the vertical buses. Given that VC allocation is performed only once per packet per hop BVA can be implemented in such a way that it does not become a system bottleneck. We evaluate our proposal with both synthetic and PARSEC benchmarks. The experimental results indicate that when compared with conventional pipelined bus or Time Division Multiple Access (TDMA) bus based systems, implementing vertical WS can reduce the bus critical path length by at least 31%, diminish the average packet transmission latency by at least 22%, and save the area cost and power consumption of the output buffers incident to the bus by 47% and 43%, respectively. To deal with potential transient and permanent faults in 3D NoCs, in this chapter we also discussed the application of our aforementioned fault tolerant techniques to detect and correct soft errors in bus VC allocators, to utilize partially faulty vertical buses, and to tolerate deactivated or totally broken vertical buses.

8.2 Future Research Directions

Although numerous strategies have been proposed to improve the NoC dependability from the architectural aspects, a lot of work still need to be done to produce NoC based MPSoCs that can meet the requirements of various kind of applications. In the following, we list several important research directions to further complete and improve the work presented in this dissertation.

- Sophistic trade off strategies that can select proper NoC structures according to the applications' requirements are required. For each dependability issue, numerous methods have been proposed to address it. According to the definition, the dependability of a system is its ability to avoid service failures that are more frequent and more severe than acceptable [6]. If the selected methods are too conservative, they utilize still functional NoC resources inefficiently and cause high area and power cost. Conversely, if the methods are overoptimistic, the corresponding NoC systems end up with low dependability. Thus sophistic trade off strategies should be able to determine the most appreciate NoC structures for specific applications.

- Accurate traffic models that can prototype the existing and emerging NoC workloads are expected. After the NoC architecture is determined, its performance and overall dependability need to be evaluated with proper benchmarks. However, due to the complicity of the applications that are suitable for NoC platforms and the dynamic behavior of the system, it is hard to obtain the prototype of the applications [66]. Consequently most researchers and designers still rely on synthetic traffic patterns which have high simulation speed but lack of accuracy. Although some general-purpose chip multiprocessor benchmarks such as SPLASH [108] and PARSEC [9] can be utilized, they may not be able to effectively stress the NoCs [66].

- Simulation software or platform that can take advantage of the parallelism of the multi-core processors are needed. Even though the commercial multi-core processors are already widely utilized in nowadays computers, most simulation software products still cannot efficiently utilize multiple cores to speedup the simulation. As a result, the simulation time is usually quite long, especially when real applications from benchmarks like PARSEC [9] are injected. Moreover, the simulation software should also be able to inject transient and permanent faults into

the NoCs to evaluate the influence of different dependability issues to the NoC performance.

- Power dissipation must be thoroughly considered in future NoC design. The future MPSoCs may contain thousands of cores, but in practice not all cores are active at the same time. It is economically more efficient to power on cores on demands and to shut down idle cores to save power [96]. An NoC, as the MPSoCs backbone communication infrastructure, should be able to support this feature from the communication point of view. Moreover, as the semiconductor industry enters the 3D era, power supply and dissipation is becoming increasingly serious concerns [89]. This issue must be thoroughly considered in the design of task mapping heuristics and NoC routing protocols.

We note that many other research directions exist. In addition, the dependability of the software running on the NoC based MPSoCs is also an important issue to define the overall system performance. Thus software/hardware co-design should be one of the disciplines in the practice to create dependable NoC based systems.

Bibliography

[1] AISOPOS, K., DEORIO, A., PEH, L., AND BERTACCO, V. Ariadne: Agnostic reconfiguration in a disconnected network environment. In *Proc. Interconnection Conference on Parallel Architectures and Compilation Techniques (PACT)* (Oct. 2011), pp. 298–309.

[2] AKBARI, S., SHAFIEE, A., FATHY, M., AND BERANGI, R. Afra: A low cost high performance reliable routing for 3d mesh nocs. In *Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE)* (Mar. 2012), pp. 332–337.

[3] ARTERIS. A comparison of network-on-chip and busses. White Paper, 2005.

[4] ASCIA, G., CATANIA, V., PALESI, M., AND PATTI, D. Implementation and analysis of a new selection strategy for adaptive routing in networks-on-chip. *IEEE Trans. Computers 57*, 6 (June 2008), 809–820.

[5] AVIRNENI, N., AND SOMANI, A. Low overhead soft error mitigation techniques for high-performance and aggressive designs. In *Proc. International Conferences on Dependable Systems and Networks (DSN)* (June 2009), pp. 185–194.

[6] AVIZIENIS, A., LAPRIE, J., RANDELL, B., AND LANDWEHR, C. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. Dependable Secure Comput. 1*, 1 (Jan./Mar. 2004), 11–33.

[7] BERNSTEIN, K., AND ET AL. Interconnects in the third dimension: Design challenges for 3d ics. In *Proc. Design Automation Conference (DAC)* (June 2007), pp. 562–567.

[8] BERTOZZI, D., JALABERT, A., MURALI, S., TAMHANKAR, T., STERGIOU, S., BENINI, L., AND MICHELI, G. Noc synthesis flow for customized domain specific multiprocessor systems-on-chip. *IEEE Trans. Parallel Distrib. Syst. 16*, 2 (Feb. 2009), 1–14.

[9] BIENIA, C. *Benchmarking Modern Multiprocessors*. PhD thesis, Princeton University, Jan. 2011.

[10] BINKERT, N., DRESLINSKI, R., HSU, L., LIM, K., SAIDI, A., AND REINHARDT, S. The m5 simulator: Modeling networked systems. *IEEE Micro 26*, 4 (June/Aug. 2006), 52–60.

[11] BJERREGAARD, T., AND MAHADEVAN, S. A survey of research and practices of network-on-chip. *ACM Computing Surveys 38*, 1 (Mar. 2006), 1–51.

[12] BLAKE, G., DRESLINSKI, R., AND MUDGE, T. A survey of multicore processors. *IEEE Signal Processing Magazine 26*, 6 (Nov. 2009), 26–37.

[13] BOGDAN, P., DUMITRAS, T., AND MARCULESCU, R. Stochastic communication: A new paradigm for fault-tolerant networks-on-chip. *Journal of VLSI Design 2007* (Feb. 2007), 1–17.

[14] BORKAR, S. Designing reliable systems from unreliable components: The challenges of transistor variability and degradation. *IEEE Micro 25*, 6 (Nov./Dec. 2005), 10–16.

[15] CARVALHO, E., CALAZANS, N., AND MORAES, F. Dynamic task mapping for mpsocs. *IEEE Design and Test of Computers 27*, 5 (Sept-Oct 2010), 26–35.

[16] CHAIX, F., AVRESKY, D., ZERGAINOH, N., AND NICOLAIDIS, M. Fault-tolerant deadlock-free adaptive routing for any set of link and node failures in multi-cores systems. In *Proc. Interconnection Symposium on Network Computing and Applications (NCA)* (June 2010), pp. 52–59.

[17] CHALASANI, S., AND BOPPANA, R. V. Communication in multicomputers with nonconvex faults. *IEEE Trans. Comput. 46*, 5 (May 1997), 616–622.

[18] CHAPIRO, D. *Globally-asynchronous locally-synchronous systems*. PhD thesis, Stanford Univerisyt, California, Oct. 1984.

[19] CHEN, C., AND CHIU, G. A fault-tolerant routing scheme for meshes with nonconvex faults. *IEEE Trans. Parallel Distrib. Syst. 12*, 5 (May 2001), 467–475.

[20] CHOU, C., AND MARCULESCU, R. Run-time task allocation considering user behavior in embedded multiprocessor networks-on-chip. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 29*, 1 (Jan 2010), 78–91.

[21] CHOU, C., OGRAS, U., AND MARCULESCU, R. Energy- and performance-aware incremental mapping for networks on chip with multiple voltage levels. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 27*, 10 (Oct 2008), 1866–1879.

[22] CONSTANTINIDES, K., PLAZA, S., BLOME, J., ZHANG, B., BERTACCO, V., MAHLKE, S., AUSTIN, T., AND ORSHANSKY, M. Bulletproof: A defect-tolerant cmp switch architecture. In *Proc. International Symposium on High-Performance Computer Architecture (HPCA)* (Feb. 2006), pp. 5–16.

[23] DALLY, W., AND TOWLES, B. Route packets, not wires: On-chip interconnection networks. In *Proc. Design Automation Conference (DAC)* (June 2001), pp. 684–689.

[24] DALLY, W., AND TOWLES, B. *Principles and Practices of Interconnection Networks*, 2004th ed. Morgan Kaufmann, San Francisco, CA, 2004.

[25] DHILLON, Y., DIRIL, A., CHATTERJEE, A., AND SINGH, A. Analysis and optimization of nanometer cmos circuits for soft-error tolerance. *IEEE Trans. VLSI Syst. 14*, 5 (May 2006), 514–524.

[26] DICK, R., RHODES, D., AND WOLF, W. Tgff: task graphs for free. In *Proc. International Workshop on Hardware/Software Codesign* (Mar. 1998), pp. 97–101.

[27] DUATO, J., YALAMANCHILI, S., AND NI, L. *Interconnection Networks: An Engineering Approach*, 2003th ed. Morgan Kaufmann, San Francisco, CA, 2003.

[28] EBRAHIMI, M., DANESHTALAB, M., LILJEBERG, P., PLOSILA, J., AND TENHUNEN, H. Cluster-based topologies for 3d networks-on-chip using advanced inter-layer bus architecture. *Journal of Computer and System Sciences 79*, 4 (June 2013), 475–491.

[29] EBRAHIMI, M., DANESHTALAB, M., PLOSILA, J., AND TENHUNEN, H. Minimal-path fault-tolerant approach using connection-retaining structure in networks-on-chip. In *Proc. IEEE/ACM International Symposium on Networks on Chip (NoCS)* (Apr. 2013), pp. 1–8.

[30] EBRAHIMI, M., ET AL. Haraq: Congestion-aware learning model for highly adaptive routing algorithm in on-chip networks. In *Proc. IEEE/ACM International Symposium on Networks on Chip* (May 2012), pp. 19–26.

[31] FARAHNAKIAN, F., EBRAHIMI, M., DANESHTALAB, M., PLOSILA, J., AND LILJEBERG, P. Optimized q-learning model for distributing traffic in on-chip networks. In *Proc. IEEE International Conference on Networked Embedded Systems for Every Application* (Dec. 2012), pp. 1–8.

[32] FARUQUE, M., KRIST, R., AND HENKEL, J. Adam: Run-time agent-based distributed application mapping for on-chip communication. In *Proc. Design Automation Conference (DAC)* (June 2008), pp. 760–765.

[33] FATTAH, M., DANESHTALAB, M., LILJEBERG, P., AND PLOSILA, J. Smart hill climbing for agile dynamic mapping in many-core systems. In *Proc. Design Automation Conference (DAC)* (May-Jun 2013), pp. 1–6.

[34] FATTAH, M., LILJEBERG, P., PLOSILA, J., AND TENHUNEN, H. Ajustable contiguity of run-time task allocation in networked many-core systems. In *Proc. Asia and South Pacific Design Automation Conference (ASP-DAC)* (2014), pp. 349–354.

[35] FATTAH, M., PALESI, M., LILJEBERG, P., PLOSILA, J., AND TENHUNEN, H. Shifa: System-level hierarchy in run-time fault-aware management of many-core systems. In *Proc. Design Automation Conference (DAC)* (June 2014), pp. 1–6.

[36] FATTAH, M., RAMIREZ, M., DANESHTALAB, M., LILJEBERG, P., AND PLOSILA, J. Cona: Dynamic application mapping for congestion reduction in many-core systems. In *Proc. International Conference on Computer Design (ICCD)* (Sep-Oct 2012), pp. 364–370.

[37] FEERO, B., AND PANDE, P. Networks-on-chip in a three-dimensional environment: A performance evaluation. *IEEE Transactions on Computers 58*, 1 (Jan. 2009), 32–45.

[38] FIORIN, L., AND SAMI, M. Fault-tolerant network interfaces for networks-on-chip. *IEEE Trans. Dependable Secure Comput. 11*, 1 (July 2013), 16–29.

[39] GLASS, C., AND NI, L. Fault-tolerant wormhole routing in meshes without virtual channels. *IEEE Trans. Parallel Distrib. Syst. 7*, 8 (June 1996), 620–636.

[40] GRECU, C., IVANOV, A., SALEH, R., AND PANDE, P. P. Noc interconnect yield improvement using crosspoint redundancy. In *Proc. IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)* (Oct. 2006), pp. 457–465.

[41] GRECU, C., IVANOV, A., SALEH, R., AND PANDE, P. P. Testing network-on-chip communication fabrics. *IEEE Trans. Comput. Aided Design Integr. Circuits Syst. 26*, 12 (Dec. 2007), 2201–2213.

[42] HERNANDEZ, C., SILLA, F., SANTONJA, V., AND DUATO, J. A new mechanism to deal with process variability in noc links. In *Proc. IEEE International Symposium on Parallel and Distributed Processing (IPDPS)* (May 2009), pp. 1–11.

[43] HESTNESS, J., AND KECKLER, S. W. Netrace: Dependency-tracking traces for efficient network-on-chip experimentation. Tech. Rep. TR-10-11, Department of Computer Science, The University of Texas at Austin, Austin, Texas, May 2010.

[44] HU, J., AND MARCULESCU, R. Energy- and performance-aware mapping for regular noc architectures. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 24*, 4 (Apr 2005), 551–562.

[45] HWANG, Y., LEE, J., AND HAN, T. 3d network-on-chip system communication using minimum number of tsvs. In *Proc. International Conference on ICT Convergence (ICTC)* (Sept. 2011), pp. 517–522.

[46] INSTRUMENTS, T. The Chip that Jack Built. `url=http://www.ti.com/corp/docs/kilbyctr/jackbuilt.shtml#top`, 2008.

[47] ITRS. International Technology Roadmap for Semiconductors – Interconnect. `url=http://www.itrs.net/Links/2005ITRS/Interconnect2005.pdf`, 2005.

[48] ITRS. International Technology Roadmap for Semiconductors – System Drivers. `url=http://www.itrs.net/Links/2011ITRS/Home2011.htm`, 2011.

[49] ITRS. International Technology Roadmap for Semiconductors – Test and Test Equipment. `url=http://www.itrs.net/Links/2011ITRS/2011Chapters/2011Test.pdf`, 2011.

[50] IWAI, H. Roadmap for 22nm and beyond. *Journal Microelectronic Engineering 86*, 7-9 (July/Sept. 2009), 1520–1528.

[51] KANG, Y., KOWN, T., AND DRAPER, J. Fault-tolerant flow control in on-chip networks. In *Proc. ACM/IEEE International Symposium on Networks-on-Chip (NOCS)* (2010), pp. 79–86.

[52] KARNIK, T., HAZUCHA, P., AND PATEL, J. Characterization of soft errors caused by single event upsets in cmos processes. *IEEE Trans. Dependable Secure Comput. 1*, 2 (Apr./June 2004), 128–143.

[53] KIM, J., AND ET AL. A novel dimensionally-decomposed router for on-chip communication in 3d architectures. In *Proc. International Symposium on Computer Architecture (ISCA)* (May 2007), pp. 138–149.

[54] KIM, J., NICOPOULOS, C., AND PARK, D. A gracefully degrading and energy-efficient modular router architecture for on-chip networks. In *Proc. International Symposium on Computer Architecture (ISCA)* (2006), pp. 4–5.

[55] KIM, J., PARK, D., NICOPOULOS, C., VIJAYKRISHNAN, N., AND DAS, C. Design and analysis of an noc architecture from performance, reliability and energy perspective. In *Proc. Symposium on Architecture for networking and communications systems (ANCS)* (Oct. 2005), pp. 173–182.

[56] KIM, S., AND HAN, T. Fault-tolerant wormhole routing in mesh with overlapped solid fault regions. *Journal of Parallel Computing 23*, 13 (Dec. 1997), 1937–1962.

[57] KOBBE, S., BAUER, L., LOHMANN, D., SCHRODER-PREIKSCHAT, W., AND HENKEL, J. Distrm: Distributed resource management for on-chip many-core systems. In *Proc. IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis (CODES+ISSS)* (Oct 2011), pp. 119–128.

[58] LAURENCIU, N., WANG, Y., AND COTOFANA, S. A direct measurement scheme of amalgamated aging effects with novel on-chip sensor. In *Proc. 21st IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)* (Istanbul, Turkey, October 2013), pp. 246–251.

[59] LAURENCIU, N. C., AND COTOFANA, S. Critical transistors nexus based circuit-level aging assessment and prediction. *Journal of Parallel and Distributed Computing* (August 2013).

[60] LEHTONEN, T., LILJEBERG, P., AND PLOSILA, J. Online reconfigurable self-timed links for fault tolerant noc. *Journal of VLSI Design 2007* (Feb. 2007), 1–13.

[61] LEHTONEN, T., WOLPERT, D., LILJEBERG, P., PLOSILA, J., AND AMPADU, P. Self-adaptive system for addressing permanent errors in on-chip interconnects. *IEEE Trans. VLSI Syst. 18*, 4 (Apr. 2010), 527–540.

[62] LI, F., NICOPOULOS, C., RICHARDSON, T., AND XIE, Y. Design and management of 3d chip multiprocessors using network-in-memory. In *Proc. International Symposium on Computer Architecture (ISCA)* (2006), pp. 130–141.

[63] LOI, I., ANGIOLINI, F., FUJITA, S., MITRA, S., AND BENINI, L. Characterization and implementation of fault-tolerant vertical links for 3-d networks-on-chip. *IEEE Trans. Comput. Aided Design Integr. Circuits Syst. 30*, 1 (Jan. 2011), 124–134.

[64] LU, R., CAO, A., AND KOH, C. Samba-bus: A high performance bus architecture for system-on-chips. *IEEE Trans. VLSI syst. 15*, 1 (Jan. 2007), 69–79.

[65] LU, Y., MCCANNY, J., AND SEZER, S. Exploring virtual-channel architecture in fpga based networks-on-chip. In *Proc. International SOC (System on Chip) Conference (SOCC)* (Sept. 2011), pp. 302–307.

[66] MARCULESCU, R., OGRAS, U., PEH, L., JERGER, M., AND HOSKOTE, Y. Outstanding research problems in noc design: System, microarchitecture, and circuit perspectives. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 28*, 1 (Jan. 2009), 3–21.

[67] MATSUTANI, H., KOIBUCHI, M., AND AMANO, H. Tightly-coupled multi-layer topologies for 3-d nocs. In *Proc. International Conference on Parallel Processing (ICPP)* (Sept. 2007), pp. 1–10.

[68] MURALI, S., BENINI, L., THEOCHARIDES, T., VIJAYKRISHNAN, N., IRWIN, M., AND MICHELI, G. Analysis of error recovery schemes for networks on chips. *IEEE Design and Test of Computers 22*, 5 (Sept. 2005), 432–442.

[69] NICOLAIDIS, M., PASCA, V., AND ANGHEL, L. Through-silicon-via built-in self-repair for aggressive 3d integration. In *Proc. IEEE International On-Line Testing Symposium (IOLTS)* (June 2012), pp. 91–96.

[70] NOLLET, V., AVASARE, P., EECKHAUT, H., DIEDERIK, AND CORPORAAL, H. Runtime management of a mpsoc containing fpga fabric tiles. *IEEE Trans. VLSI Syst. 16*, 1 (Jan 2008), 24–33.

[71] PALESI, M., KUMAR, S., AND CATANIA, V. Bandwidth-aware routing algorithms for networks-on-chip platforms. *IET Computers and Digital Techniques 3*, 5 (Sept. 2009), 413–429.

[72] PALESI, M., KUMAR, S., AND CATANIA, V. Leveraging partially faulty links usage for enhancing yield and performance in network-on-chip. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 29*, 3 (Mar. 2010), 426–440.

[73] PARK, D., EACHEMPATI, S., DAS, R., MISHRA, A., XIE, Y., VIJAYKRISHNAN, N., AND DAS, C. Mira: A multi-layered on-chip interconnect router architecture. In *Proc. International Symposium on Computer Architecture (ISCA)* (June 2008), pp. 251–261.

[74] PARK, D., NICOPOULOS, C., KIM, J., VIJAYKRISHNAN, N., AND DAS, C. Exploring fault-tolerant network-on-chip architectures. In *Proc. International Conferences on Dependable Systems and Networks (DSN)* (June 2006), pp. 93–104.

[75] PARKHURST, J., DARRINGER, J., AND GRUNDMANN, B. From single core to multi-core: Preparing for a new exponential. In *Proc. IEEE/ACM international conference on Computer-aided design (ICCAD)* (Nov. 2006), pp. 67–72.

[76] PASCA, V., ANGHEL, L., RUSU, C., AND BENABDENBI, M. Configurable serial fault-tolerant link for communication in 3d integrated systems. In *Proc. IEEE International On-Line Testing Symposium (IOLTS)* (July 2010), pp. 115–120.

[77] PASCA, V., REHMAN, S., ANGHEL, L., AND BENABDENBI, M. Efficient link-level error resilience in 3d nocs. In *Proc. International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)* (Apr. 2012), pp. 127–132.

[78] PASRICHA, S. Exploring serial vertical interconnects for 3d ics. In *Proc. Design Automation Conference (DAC)* (July 2009), pp. 581–586.

[79] PASRICHA, S., AND ZOU, Y. A low overhead fault tolerant routing scheme for 3d networks-on-chip. In *Proc. International Symposium on Quality Electronic Design (ISQED)* (Mar. 2011), pp. 1–8.

[80] PETERSON, W., AND WELDON, E. *Error-correcting Codes*, 1972th ed. MIT Press, 1972.

[81] PUENTE, V., GREGORIO, J., VALLEJO, F., AND BEIVIDE, R. Immunet: Dependable routing for interconnection networks with arbitrary topology. *IEEE Trans. Comput. 57*, 12 (Dec. 2008), 1676–1689.

[82] RAHMANI, A., LATIF, K., LILJEBERG, P., PLOSILA, J., AND TENHUNEN, H. Interconnects in the third dimension: Design challenges for 3d ics. In *Proc. NORCHIP* (Nov. 2010), pp. 1–6.

[83] RAHMANI, A., LATIF, K., LILJEBERG, P., PLOSILA, J., AND TENHUNEN, H. A stacked mesh 3d noc architecture enabling congestion-aware and reliable inter-layer communication. In *Proc. International Euromicro Conference on Parallel, Distributed and Network-based Processing (PDP)* (Feb. 2011), pp. 423–430.

[84] RANTALA, V., LEHTONEN, T., LILJEBERG, P., AND PLOSILA, J. Multi network interface architectures for fault tolerant network-on-chip. In *Proc. International Symposium on Signals, Circuits and Systems (ISSCS)* (July 2009), pp. 1–4.

[85] REED, D., AND GRUNWALD, D. The performance of multiprocessor interconnection networks. *IEEE Trans. Comput. 20*, 6 (June 1987), 63–73.

[86] RUSU, C., ANGHEL, L., AND AVRESKY, D. Adaptive inter-layer message routing in 3d networks-on-chip. *Journal of Microprocessors and Microsystems 35*, 7 (Oct. 2011), 613–631.

[87] SALMINEN, E., KULMALA, A., AND HAMALAINEN, T. Survey of network-on-chip proposals. White Paper, 2008.

[88] SCHWIEBERT, L., AND JAYASIMHA, D. N. Optimal fully adaptive wormhole routing for meshes. In *Proc. Supercomputing (DAC)* (Nov. 1993), pp. 782–791.

[89] SEPULVEDA, J., GOGNIAT, G., PIRES, R., CHAU, W., AND STRUM, M. An evolutive approach for designing thermal and performance-aware heterogeneous 3d-nocs. In *Proc. Symposium on Integrated Circuits and Systems Design (SBCCI)* (Sept. 2013), pp. 1–6.

[90] SHIRINZADEH, S., AND R.ASLI. A novel soft error hardened latch design in 90nm cmos. In *Proc. CSI International Symposium on Computer Architecture and Digital Systems (CADS)* (May 2012), pp. 60–63.

[91] SHIVAKUMAR, P., KISTLER, M., KECKLER, S., BURGER, D., AND ALVISI, L. Modeling the effect to technology trends on the soft error rate of combinational logic. In *Proc. International Conferences on Dependable Systems and Networks (DSN)* (2002), pp. 389–398.

[92] SINGH, A., SHAFIQUE, M., KUMAR, A., AND HENKEL, J. Mapping on multi/many-core systems: Survey of current and emerging trends. In *Proc. Design Automation Conference (DAC)* (June 2013), pp. 1–10.

[93] STRANO, A., HERNANDEZ, C., SILLA, F., AND BERTOZZI, D. Process variation and layout mismatch tolerant design of source synchronous links for gals networks-on-chip. In *Proc. IEEE International Symposium on System on Chip (SoC)* (Sept. 2010), pp. 43–48.

[94] SUN, F., CEVRERO, A., ATHANASOPOULOS, P., AND LEBLEBICI, Y. Design and feasibility of multi-gb/s quasi-serial vertical interconnects based on tsvs for 3d ics. In *Proc. IEEE/IFIP International Conference on VLSI and System-on-Chip (VLSI-SoC)* (Sept. 2010), pp. 149–154.

[95] TAMHANKAR, R., MURALI, S., STERGIOU, S., PULLINI, A., ANGIOLINI, F., BENINI, L., AND MICHELI, G. Timing-error-tolerant network-on-chip design methodology. *IEEE Trans. Comput. Aided Design Integr. Circuits Syst. 26*, 7 (July 2007), 1297–1310.

[96] TAYLOR, M. A landscape of the new dark silicon design regime. *IEEE Micro 33*, 5 (Aug. 2013), 8–19.

[97] TSAI, W., ZHEN, D., CHEN, S., AND HU, Y. A fault-tolerant noc scheme using bidirectional channel. In *Proc. Design Automation Conference (DAC)* (June 2011), pp. 918–923.

[98] UNSAL, O., TSCHANZ, J., BOWMAN, K., DE, V., VERA, X., GONZALEZ, A., AND ERGIN, O. Impact of parameter variations on circuits and microarchitecture. *IEEE Micro 26*, 6 (Nov./Dec. 2006), 30–39.

[99] VAJDA, A. *Programming Many-Core Chips*, 2011th ed. Springer, New York, NY, 2011.

[100] VITKOVSKIY, A., SOTERIOU, V., AND NICOPOULOS, C. A fine-grained link-level fault-tolerant mechanism for networks-on-chip. In *Proc. International Conference on Computer Design (ICCD)* (Oct. 2010), pp. 447–454.

[101] VITKOVSKIY, A., SOTERIOU, V., AND NICOPOULOS, C. A highly robust distributed fault-tolerant routing algorithm for nocs with localized rerouting. In *Proc. Interconnection Network Architecture: On-Chip, Multi-Chip Workshop (INA-OCMC)* (Jan. 2012), pp. 29–32.

[102] WANG, Y., COTOFANA, S., AND FANG, L. A unified aging model of nbti and hci degradation towards lifetime reliability management for nanoscale mosfet circuits. In *Proc. IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)* (June 2011), pp. 175–180.

[103] WANG, Y., COTOFANA, S., AND FANG, L. Lifetime reliability assessment with aging information from low-level sensors. In *Proc. ACM international conference on Great lakes symposium on VLSI (GLSVLSI)* (May 2013), pp. 339–340.

[104] WIKIPEDIA. Ambric. `url=http://en.wikipedia.org/wiki/Ambric`, 2014.

[105] WIKIPEDIA. Moore's law. `url=http://en.wikipedia.org/wiki/Moore%27s_law`, 2014.

[106] WIKIPEDIA. Teraflops Research Chip. `url=http://en.wikipedia.org/wiki/Teraflops_Research_Chip`, 2014.

[107] WIKIPEDIA. Network on a Chip. `url=http://en.wikipedia.org/wiki/Network_on_a_chip`, 2015.

[108] WOO, S., OHARA, M., TORRIE, E., SINGH, J., AND GUPTA, A. The splash-2 programs: characterization and methodological considerations. In *Proc. IEEE/IFIP International Symposium on Computer Architecture (ISCA)* (May 1995), pp. 24–36.

[109] WU, K., AND MARCULESCU, D. A low-cost, systematic methodology for soft error robustness of logic circuits. *IEEE Trans. VLSI Syst. 21*, 2 (Feb. 2013), 367–379.

[110] XIE, Y., CONG, J., , AND SAPATNEKAR, S. *Three-Dimensional Integrated Circuit Design: EDA, Design and Microarchitectures*. Springer, New York, NY, 2010.

[111] XIE, Y., LOH, G., BLACK, B., AND BERNSTEIN, K. Design space exploration for 3d architectures. *ACM Journal on Emerging Technologies in Computing Systems 2*, 2 (Apr. 2006), 65–103.

[112] YIN, A., XU, C., LILJEBERG, P., AND TENHUNEN, H. Explorations of honeycomb topologies for network-on-chip. In *Proc. International Conference on Network and Parallel Computing (NPC)* (Oct. 2009), pp. 73–79.

[113] YU, Q., AND AMPADU, P. Adaptive error control for noc switch-to-switch links in a variable noise environment. In *Proc. IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems (DFT)* (Oct. 2008), pp. 352–360.

[114] YU, Q., AND AMPADU, P. Transient and permanent error co-management method for reliable networks-on-chip. In *Proc. IEEE/ACM International Symposium on Networks on Chip* (July 2010), pp. 52–59.

[115] YU, Q., ZHANG, M., AND AMPADU, P. Exploiting inherent information redundancy to manage transient errors in noc routing arbitration. In *Proc. IEEE/ACM International Symposium on Networks on Chip (NoCS)* (May 2011), pp. 105–112.

[116] ZHANG, Z., GREINER, A., AND TAKTAK, S. A reconfigurabel routing algorithm for a fault tolerant 2d-mesh network-on-chip. In *Proc. Design Automation Conference (DAC)* (June 2008), pp. 441–446.

[117] ZHU, M., LEE, J., AND CHOI, K. An adaptive routing algorithm for 3d mesh noc with limited vertical bandwidth. In *Proc. IEEE/IFIP International Conference on VLSI and System-on-Chip (VLSI-SoC)* (Oct. 2012), pp. 18–23.

List of Publications

International Journals

1. **C. Chen**, Y. Fu, and S. D. Cotofana, **Toward Maximum Utilization of Remained Bandwidth in Defected NoC Links**, submitted.

International Conference/Workshop Proceedings

1. **C. Chen**, M. Enachescu, S. D. Cotofana, **Enabling Vertical Wormhole Switching in 3D NoC-Bus Hybrid Systems**, Proc. Design Automation and Test in Europe (DATE), pp. 507-512, Grenoble, France, Mar. 2015.
2. **C. Chen**, S. D. Cotofana, **Link Bandwidth Aware Backtracking Based Dynamic Task Mapping in NoC based MPSoCs**, Proc. International Workshop on Network on Chip Architectures (NoCArc), pp. 5-10, Cambridge, UK, Dec. 2014.
3. **C. Chen**, S. D. Cotofana, **Towards an Effective Utilization of Partially Defected Interconnections in 2D Mesh NoCs**, Proc. IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pp. 492-497, Tampa, Florida, Jul. 2014.
4. **C. Chen**, S. D. Cotofana, **A Low Cost Method to Tolerate Soft Errors in the NoC Router Control Plane**, Proceedings of International IEEE SoC (System-on-Chip) Conference (SOCC), Erlangen, Germany, pp. 374–379, Sep. 2013.
5. **C. Chen**, S. D. Cotofana, **An Effective Routing Algorithm to Avoid Unnecessary Link Abandon in 2D Mesh NoCs**, Proc. Euromicro Conference on Digital System Design (DSD), pp. 311–318, Santander, Spain, Sep. 2013.
6. Y. Lu, **C. Chen**, J. McCanny, and S. Sezer, **Design of interlock-free combined allocators for Networks-on-Chip**, 2012 IEEE International SOC Conference (SOCC), pp. 358-363, Niagara Falls, New York, Sep. 2012.
7. **C. Chen**, Y. Lu, and S. D. Cotofana, **A Novel Flit Serialization Strategy to Utilize Partially Faulty Links in Networks-on-Chip**, Proc. IEEE/ACM International Symposium on Networks on Chip (NoCS), pp.124-131, Lyngby, Denmark, May 2012.

Samenvatting

Agressieve schaling van halfgeleidertechnologie verschaft de middelen voor het verdubbelen van het aantal transistors op een enkele chip elke 18 maanden. Om efficiënt gebruik te maken van deze enorme chip resources zijn Multi-Processor Systems on Chip (MPSoCs), geïntegreerd met een Netwerk-on-Chip (NoC) communicatie-infrastructuur, op grote schaal onderzocht. Echter verhoogt de transistor miniaturisatie aanzienlijk de mogelijkheid van transient en permanente fouten, in het bijzonder voor NoCs aangezien ze geometrisch zijn verspreid over de chip. Om een betrouwbare communicatie service te leveren, moet de NOC zijn functionaliteit behouden en zijn prestaties verlagen in de aanwezigheid van fouten. In dit proefschrift stellen we verschillende nieuwe afgestemde NoC mechanismen voor om fouten te tolereren die veroorzaakt worden door bijvoorbeeld variability agents, ageing, agressieve omgevingsfactoren en het efficiënt gebruik van nog steeds functionele NoC componenten. We introduceren eerst een methode met lage kosten om correcte flit transmissie mogelijk te maken zelfs wanneer soft errors in de router control voor komen. Daarna stellen we een Flit Serialization (FS) strategie voor om defecte verbindingen te tolereren en om de resterende bandbreedte te gebruiken. Binnen het FS kader worden zwaar beschadigde verbindingen waarvan de fout niveau boven een bepaalde drempelwaarde overschrijden gedeactiveerd om congestie in hun upstream routers te verminderen. Bovendien ontwerpen wij een routing algoritme gebaseerd op gedistribueerde logica die in staat is om volledig gebroken verbindingen te toleren en daarnaast efficiënt gebruik maakt van UnPaired Functional (UPF) verbindingen in gedeeltelijk defecte verbindingen. We introduceren ook een run-time taak mapping algoritme die op de hoogte van de verbindingsbandbreedte is, om de mapping kwaliteit van nieuwe geïnjecteerde toepassingen in MPSoCs te verbeteren. Tot slot bespreken we de toepassing van de bovengenoemde strategieën in 3D NoC systemen en stellen we een Bus Virtual channel Allocation (BVA) mechanisme voor om verticaal wormhole switching mogelijk te maken om de prestaties van 3D NoC-Bus hybride systemen te verbeteren. Alle voorstellen zijn gevalueerd in onze gemengde NoC simulatie platform en hun voordeel ten opzichte van state of the art tegenhangers is bewezen door middel van experimentele resultaten.

Propositions

accompanying the PhD Dissertation

Towards Dependable Network-on-Chip Architectures
by Changlin Chen

1. A Network-on-Chip is a scalable and reliable communication infrastructure replacement of buses and crossbars in Multi-Processor Systems on Chip. [This thesis]
2. A partially defective link usually contains only a small number of broken wires, thus most of its bandwidth still can be utilized. [This thesis]
3. Efficiently utilizing still functional resources is as important as tolerating faults to improve the system dependability. [This thesis]
4. The most practicable strategy to provide a dependable service is not to use undependable devices.
5. Interest is the best motivation.
6. When you look around and find out that everybody can be better than you, you should think about changing your career.
7. The ability to find the questions is more important than that to answer the questions for PhD candidates.
8. If your method is becoming more and more complicated, most probably you are going in the wrong direction.
9. Incremental should never be a reason to reject a paper, as every contribution is based on previously published ones.
10. I can accept failure, everyone fails at something. But I cannot accept not trying. – Michael Jordan
11. Every choice has pros and cons, you will occasionally regret your choice no matter what you chose.
12. Happy wife, happy life. – Deborah Carr, professor of sociology at Rutgers University.

These propositions are regarded as opposable and defendable, and have been approved as such by the promotor, Prof. dr. K.L.M. Bertels.

Stellingen

behorende bij het proefschrift

Towards Dependable Network-on-Chip Architectures

door Changlin Chen

1. Een Netwerk-on-Chip is een schaalbare en betrouwbare communicatie-infrastructuur vervanger van bussen en crossbars in Multi-Processor Systems on Chip. [Dit proefschrift]
2. Een gedeeltelijk defecte verbinding bevat meestal slechts een klein aantal kapotte draden, waardoor het merendeel van de bandbreedte nog steeds kan worden gebruikt. [Dit proefschrift]
3. Het efficiënter gebruik van nog steeds functionerende middelen is even belangrijk als het tolereren van fouten om de systeem betrouwbaarheid te verbeteren. [Dit proefschrift]
4. De meest uitvoerbare strategie om een betrouwbare service te verlenen is om onbetrouwbaar componenten niet te gebruiken.
5. Interesse is de beste motivatie.
6. Als je om je heen kijkt en je komt er achter dat iedereen beter dan jou kan zijn, dan moet je nadenken over het veranderen van je carrière.
7. In staat zijn om vragen te vinden is voor promovendi belangrijker dan de vragen te beantwoorden.
8. Als je methode steeds ingewikkelder wordt ga je waarschijnlijk de verkeerde kant op.
9. “Incrementele bijdrage” mag nooit een reden zijn om een publicaties te verwerpen, omdat elke bijdrage gebaseerd is op eerder publicaties.
10. Ik kan mislukkingen accepteren, iedereen faalt wel eens. Maar ik kan het niet proberen niet aanvaarden. - Michael Jordan
11. Elke keuze heeft voor en nadelen, af en toe zal je spijt krijgen van je keuze ongeacht wat je gekozen hebt.
12. Een gelukkige vrouw leidt tot een gelukkig leven. - Deborah Carr, hoogleraar sociologie aan de Rutgers University.

Deze stellingen worden opponeerbaar en verdedigbaar geacht en zijn als zodanig goedgekeurd door de promotor, Prof. dr. K.L.M. Bertels.

Curriculum Vitae

Changlin Chen was born on June 1st, 1986 in Taian, Shandong Province, China. From September 2003 to July 2005 he studied at the School of Medicine, Shandong University (SDU) in Jinan, China. In September 2005, he changed his major from Clinical Medicine to Electronic Information Science and Technology and continued his study at the School of Information Science and Engineering, SDU. He received Bachelor's degree in Electronic Information Science and Technology in 2008. Subsequently, he graduated with Master's degree in Information and Communication Engineering in 2010, from National University of Defense Technology (NUDT) in Changsha, China.

In 2010 he was awarded a government scholarship from Chinese Scholarship Council (CSC), to pursue his PhD studies in the Netherlands. In December 2010, he joined the Computer Engineering laboratory of Delft University of Technology in the Netherlands, under the supervision of Associate Professor Dr. Sorin Cotofana. The major focus of his PhD studies are on dependable Network on Chip architectures. The results of this work are presented in the current dissertation.

His research interests include Network on Chip, Multi-Processor Systems on Chip, parallel computation, high speed signal processing, fault tolerance, circuit system design, etc.