EMBEDDED DYNAMIC PROGRAMMING NETWORKS FOR NETWORKS-ON-CHIP

RA'ED AL-DUJAILY

A Thesis Submitted for the Degree of
Doctor of Philosophy at Newcastle University

School of Electrical, Electronic & Computer Engineering
Faculty of Science, Agriculture & Engineering

March 2013

ABSTRACT

Relentless technology downscaling and recent technological advancements in three dimensional integrated circuit (3D-IC) provide a promising prospect to realize heterogeneous system-on-chip (SoC) and homogeneous chip multiprocessor (CMP) based on the networks-on-chip (NoCs) paradigm with augmented scalability, modularity and performance. In many cases in such systems, scheduling and managing communication resources are the major design and implementation challenges instead of the computing resources. Past research efforts were mainly focused on complex design-time or simple heuristic run-time approaches to deal with the on-chip network resource management with only local or partial information about the network. This could yield poor communication resource utilizations and amortize the benefits of the emerging technologies and design methods. Thus, the provision for efficient run-time resource management in large-scale on-chip systems becomes critical. This thesis proposes a design methodology for a novel run-time resource management infrastructure that can be realized efficiently using a distributed architecture, which closely couples with the distributed NoC infrastructure. The proposed infrastructure exploits the global information and status of the network to optimize and manage the on-chip communication resources at run-time.

There are four major contributions in this thesis. First, it presents a novel deadlock detection method that utilizes run-time transitive closure (TC) computation to discover the existence of deadlock-equivalence sets, which imply loops of requests in NoCs. This detection scheme, TC-network, guarantees the discovery of all true-deadlocks without false alarms in contrast to state-of-the-art approximation and heuristic approaches. Second, it investigates the advantages of implementing future on-chip systems using three dimensional (3D) integration and presents the design, fabrication and testing results of a TC-network implemented in a fully stacked three-layer 3D architecture using a through-silicon via (TSV) complementary metal-oxide semiconductor (CMOS) technology. Testing results demonstrate the effectiveness of such a TC-network for deadlock detection with minimal computational delay in a large-scale network. Third, it introduces an adaptive strategy to effectively diffuse heat throughout the three dimensional network-on-chip (3D-NoC) geometry. This strategy employs a dynamic programming technique to select and optimize the direction of data manoeuvre in NoC. It leads to a tool, which is based on the accurate *HotSpot* thermal model and *SystemC* cycle accurate model, to simulate the thermal system and evaluate the proposed approach. Fourth, it

presents a new dynamic programming-based run-time thermal management (DPRTM) system, including reactive and proactive schemes, to effectively diffuse heat throughout NoC-based CMPs by routing packets through the coolest paths, when the temperature does not exceed chip's thermal limit. When the thermal limit is exceeded, throttling is employed to mitigate heat in the chip and DPRTM changes its course to avoid throttled paths and to minimize the impact of throttling on chip performance.

This thesis enables a new avenue to explore a novel run-time resource management infrastructure for NoCs, in which new methodologies and concepts are proposed to enhance the on-chip networks for future large-scale 3D integration.

DECLARATION

I hereby declare that this thesis is my own work and effort and that it has not been submitted anywhere for any award. Where other sources of information have been used, they have been acknowledged.

Newcastle Upon Tyne, March 2013

RA'ED AL-DUJAILY

CERTIFICATE OF APPROVAL

I confirm that, to the best of my knowledge, this thesis is from the student's own work and effort, and all other sources of information used have been acknowledged. This thesis has been submitted with my approval.

Newcastle Upon Tyne, March 2013

ALEX YAKOVLEV

PUBLICATIONS

Referred journal publications

1. **R. Al-Dujaily**, N. Dahir, T. Mak, F. Xia, A. Yakovlev, "Dynamic Programming-based Runtime Thermal Management (DPRTM): An On-line Thermal Control Strategy for 3D-NoC Systems", *ACM Transactions on Design Automation of Electronic Systems*, (submitted), pp. 1 - 28, 2012.

2. **R. Al-Dujaily**, T. Mak, K. P. Lam, F. Xia, A. Yakovlev, C. S. Poon, "Dynamic thermal optimization in three dimensional networks-on-chip", *The Computer Journal*, (in press), pp. 1 - 15, 2012.

3. **R. Al-Dujaily**, T. Mak, F. Xia, A. Yakovlev, M. Palesi, "Embedded transitive-closure network for run-time deadlock detection in networks-on-chip", *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 7, pp. 1205 - 1215, 2012.

4. T. Mak, **R. Al-Dujaily**, K. Zhou, K. P. Lam, Y. Meng, A. Yakovlev, C. S. Poon, "Dynamic programming networks for large-scale 3D chip integration", *IEEE Circuits and Systems Magazine*, vol. 11, no. 3, pp. 51 - 62, 2011.

Conference, workshop and forum publications:

1. **R. Al-Dujaily**, T. Mak, F. Xia, A. Yakovlev, K. P. Lam and C. S. Poon, "Dynamic Thermal Optimization in 3-D NoC", DATE 2012 workshop in: Design, Automation and Test of 3D-ICs, 2012, pp. 1 - 2.

2. **R. Al-Dujaily**, T. Mak, K. Zhou, K. P. Lam, Y. Meng, A. Yakovlev, C. S. Poon, "On-chip dynamic programming networks using 3D-TSV integration", Embedded Computer Systems (SAMOS), 2011 International Conference on, July, pp. 318 - 325.

3. **R. Al-Dujaily**, T. Mak, F. Xia, A. Yakovlev, "Run-Time Resources Management for 3D Networks-on-Chip based Systems," UK Electronics Forum (UKEF 2011), Manchester, 4-5 July, 2011, pp. 1 - 7.

4. C. S. Poon, T. Mak, **R. Al-Dujaily**, K. Zhou, and K. P. Lam, "On-chip dynamic programming networks design in TSV-based 3D stacking technology," In GOMAC Tech. Conference Session 10 on 3D-IC, 19-22 March, 2011.

5. **R. Al-Dujaily**, T. Mak, F. Xia, A. Yakovlev and M. Palesi, "Run-time deadlock detection in networks-on-chip using coupled transitive closure networks," Design, Automation Test in Europe Conference Exhibition (DATE), 2011, March, pp. 497 - 502 (**BEST PAPER AWARD**).

6. **R. Al-Dujaily**, T. Mak, F. Xia, and A. Yakovlev, "A Methodology for Deadlock Detector Minimization in Interconnection Network," UK Electronics Forum (UKEF 2010), Newcastle, July, 2010, pp. 1 - 6.

7. **R. Al-Dujaily**, L. Dai, F. Xia, D. Shang and A. Yakovlev, "NoC Monitoring Infrastructure: A supervisory and Control Services" in the 21st UK Asynchronous Forum, Bristol, 14-15th September, 2009, pp. 1 - 6.

"There is no capital more useful than intellect and wisdom, and there is no indigence more injurious than ignorance and unawareness."

— Ali ibn Abi Talib

ACKNOWLEDGMENTS

The research presented in this thesis was carried out under the supervision of Prof. Alex Yakovlev, Dr Terrence Mak and Dr Fei Xia of Newcastle University. I wish to express my profound thanks to them for their support, inspiration, editorial control and counsel throughout the course of my studies.

I am grateful to the Iraqi Ministry of Higher Education and Scientific Research (MOHESR) for funding my PhD study through their Scholarship Programme and also wish to express my gratitude to the Iraqi Cultural Attaché in London.

I would like to thank Dr Maurizio Palesi for his help and advices in modifying the NoC simulator used in this thesis. We had joint paper publications.

I would like to thank my thesis examination committee (Professor Kees Goossens and Dr Albert Koelmans) for their support in this endeavour.

I am grateful to the members of the Microelectronics System Design group at Newcastle University for assisting me in my study. Many thanks go to my friends and colleagues — Nizar Dahir, Ghaith Tarawneh, Maxim Rykunov, Reza Ramezani and Petros Missailidis — for numerous inspiring discussions, productive criticism, endless support, and for simply being very pleasant company.

I would also like to acknowledge MIT Lincoln Laboratory (MIT-LL) for providing access to the MIT-LL 3D circuit integration technology which used to fabricate the 3D-IC presented in Chapter 4.

Last but not least, I wish to thank my family for their continuous support and motivation throughout the duration of my studies.

CONTENTS

I Thesis Chapters 1

1 INTRODUCTION 2
1.1 Motivation and Objective 2
1.2 Structure of the Thesis 4
1.3 Statement of Originality 5

2 BACKGROUND AND LITERATURE REVIEW 8
2.1 Introduction 8
2.2 Networks-on-Chip (NoCs) 9
2.3 Components of NoCs 9
2.3.1 Router 10
2.3.2 Link 11
2.3.3 Network Interface 12
2.3.4 Intellectual Property Core 12
2.4 NoCs Flow Control 13
2.4.1 Message-Based Flow Control 14
2.4.2 Packet-Based Flow Control 14
2.4.3 Flit-Based Flow Control 15
2.5 Routing Algorithms 16
2.5.1 Deterministic 16
2.5.2 Oblivious 16
2.5.3 Adaptive 17
2.5.4 Minimal vs Non-minimal 17
2.6 Deadlock and Livelock 17
2.6.1 Deadlock 17
2.6.2 Livelock 22
2.7 NoCs and the Emerging Technologies 22
2.7.1 Three Dimensional Integration 22
2.7.2 On-Chip Optical Interconnect 25
2.7.3 On-Chip Wireless RF Interconnect 25
2.8 Networks-on-Chip: A Review 25
2.8.1 The Emergence of NoCs 26
2.8.2 Routing Algorithms and Flow Control 27
2.8.3 Different Architecture Philosophy 29

3 RUN-TIME DEADLOCK DETECTION 38
3.1 Introduction 38
3.2 Related Work 41
3.3 Methodology for Deadlock Detection 42
3.3.1 Background and Assumptions 42

3.3.2 Equivalence Set Criterion for Deadlock 43
3.3.3 Equivalence Set Computational Complexity . . 46
3.4 Transitive Closure (TC) Network Architecture 48
3.4.1 TC Computation with TC-networks 48
3.4.2 Coupling TC-network to NoC 50
3.4.3 Hardware Implementation 54
3.5 Results and Discussion 58
3.5.1 Evaluation Methodology 58
3.5.2 Evaluation Results 59
3.6 Deadlock Recovery System 71
3.6.1 End-to-End Recovery Protocol 71
3.6.2 Deadlock Recovery System: Evaluation Results 75
3.7 Summary and Conclusions 77

4 3D-NOCS BASED TSV INTEGRATION 80
4.1 Introduction 80
4.2 Advantages of 3D-NoCs over 2D-NoCs 81
4.2.1 Average Shortest Path Computation 81
4.2.2 Throughput and Network Saturation 85
4.2.3 Deadlocks Formation Rate 86
4.3 Prototyping and Testing 88
4.3.1 3D TC-network coupled with 3D-NoC 89
4.3.2 Three-tier Ring Oscillator 89
4.3.3 Testing the 3D-IC chip 90
4.4 Results and Discussion 93
4.4.1 Ring Oscillator 93
4.4.2 TC-network for Deadlock Detection 93
4.5 Summary and Conclusions 94

5 RUN-TIME THERMAL ADAPTATION IN 3D-NOCS 97
5.1 Introduction 97
5.2 Related Work 98
5.3 Methodology for Run-Time Thermal Adaptation 99
5.3.1 Dynamic Programming 99
5.3.2 Coolest Path as Shortest Path 100
5.3.3 Dynamic Programming Network 102
5.3.4 Coupling DP-network with 3D-NoC 103
5.3.5 NoC Routing with DP-network 103
5.3.6 DP-network Convergence Time and Complexity 104
5.4 On-Chip Communication, Power and Thermal Models 106
5.4.1 Communication Model 106
5.4.2 Area and Power Model 106
5.4.3 Thermal Model 106
5.4.4 Tool Chain for Dynamic Thermal Optimization for 3D-NoCs 110
5.5 Results and Discussion 111
5.5.1 Evaluation Methodology 111

5.5.2 Evaluation Results 112
5.6 Summary and Conclusions 123

6 RUN-TIME THERMAL MANAGEMENT FOR 3D-NOCS 124
6.1 Introduction 124
6.2 Related Work 126
6.3 Dynamic Programming Run-time Thermal Management (DPRTM) 127
6.3.1 System Overview 127
6.3.2 Path Cost Computation 128
6.3.3 Coupling DPRTM with 3D-NoC 129
6.3.4 Throttling Techniques 130
6.3.5 NoC Routing in the DPRTM 131
6.3.6 DPRTM Deadlock Freedom 132
6.4 Hardware Implementation 132
6.5 Results and Discussion 135
6.5.1 Evaluation Methodology 135
6.5.2 Proactive Routing Evaluation Results 135
6.5.3 Reactive Routing Evaluation Results 141
6.5.4 DPRTM Evaluation Results 142
6.5.5 Area and Power Estimation 146
6.6 Summary and Conclusions 148

7 CONCLUSIONS AND FUTURE WORK 150
7.1 Summary and Conclusions 150
7.2 Future Work 152

II Thesis Appendices 155

A FABRICATED 3D CHIP SUPPLEMENTARY MATERIALS 156

B THERMAL MATERIAL PARAMETERS 161

III Thesis Bibliography 163

BIBLIOGRAPHY 164

LIST OF FIGURES

Figure 1.1 The trend of logic delay and interconnect delay with technology scaling (source data: International Technology Roadmap for Semiconductors (ITRS) [3]). 3
Figure 2.1 General network-on-chip (NoC) structure 10
Figure 2.2 Different network topologies beside a bus topology [36] 11
Figure 2.3 A typical network-on-chip (NoC) router microarchitecture [92] 12
Figure 2.4 An illustration of message segmentation to packets, flits and phits. These different granularities of data representation are directly associated with the resource allocation of different flow control technique [52]. 13
Figure 2.5 A deadlock cycle in a two dimensional (2D) mesh network 18
Figure 2.6 Different vertical interconnect technologies: an illustration 23
Figure 2.7 The possible eight turns in 2D mesh beside the allowed turns in different routing algorithms [65] 27
Figure 3.1 Four packets in a 2D network are waiting for one another and forming a deadlock configuration. 39
Figure 3.2 Graphical representation of Example 1 45
Figure 3.3 Detecting deadlock from a two dimensional network-on-chip (2D-NoC): (a) A scenario of deadlock formation; (b) the channel wait-for graph (CWG) of the network; (c) the TC of the CWG, the set of channels $\{ch_1, ch_5, ch_7, ch_9\}$ satisfy the deadlock-equivalence set (DES) definition (Eq.3.1). 46
Figure 3.4 Unit interconnection in a general TC-network where $1 \leqslant i, j, k \leqslant n; k \neq i, j$ [101]. The output of Unit (i,j) will be the input of other units according to the problem network structure. At each unit, there are h sites, which corresponds to the total number of neighbouring nodes of i, to carry out the inference operations as defined in the site function. 49
Figure 3.5 A TC-network coupled to a 2D mesh NoC. 51
Figure 3.6 A Hamiltonian cycle [22] for a token ring distribution 52

Figure 3.7 Schematic of the router. (Left) Top-level view. (Middle) Block implementing the TC-unit. (Right) Block implementing the TC-computation 56
Figure 3.8 An example of token-ring protocol implementation 57
Figure 3.9 Percentage of detected deadlocked packets to the total received packets using the TC-network and the timeout mechanism under *uniform* traffic scenario 59
Figure 3.10 Performance of the TC-network and the timeout mechanism under *uniform* traffic scenario with different injection rates 60
Figure 3.11 Percentage of total energy consumed to resolve detected deadlocks to the total NoC energy under *uniform* traffic scenario 61
Figure 3.12 Percentage of detected deadlocked packets to the total received packets using the TC-network and the timeout mechanism under *bit-reversed* traffic scenario 62
Figure 3.13 Performance of the TC-network and the timeout mechanism under *bit-reversed* traffic scenario with different injection rates 63
Figure 3.14 Percentage of total energy consumed to resolve detected deadlocks to the total NoC energy under *bit-reversed* traffic scenario 63
Figure 3.15 Percentage of detected deadlocked packets to the total received packets using different methods under multi-media system (MMS) 65
Figure 3.16 Performance of the TC-network and the timeout mechanism under MMS with different injection rates 66
Figure 3.17 Percentage of energy consumed to resolve detected deadlocks (retransmission) to the total NoC energy (transmission + retransmission) under MMS 67
Figure 3.18 TC-network convergence time for different network topologies and sizes 71
Figure 3.19 Transmission scenario 1, which illustrates how to resolve the deadlock if it forms part of the *data* packet. 72
Figure 3.20 Transmission scenario 2, which illustrates how to resolve the deadlock if it forms part of the *Ack* packet. 73
Figure 3.21 Transmission scenario 3, which illustrates how to resolve the deadlock if it forms part of the *Nack* packet. 74

Figure 3.22 Evaluation of deadlock recovery system with *shuffle* traffic distribution and different deadlock detection techniques. 77
Figure 3.23 Energy required to drain 1 Mbyte of data in the deadlock recovery system for *shuffle* traffic and using different deadlock detection techniques. . 78
Figure 3.24 Performance evaluation of deadlock recovery and deadlock avoidance systems with *shuffle* traffic distribution. 79
Figure 3.25 Energy required to drain 1 Mbyte of data in deadlock recovery and deadlock avoidance systems for *shuffle* traffic. 79
Figure 4.1 Eight nodes NoC interconnected in 2D 83
Figure 4.2 Eight nodes NoC interconnected in 3D 84
Figure 4.3 TSV unit length modelling in three dimensional networks-on-chip (3D-NoCs) compared to planar 3D-NoCs links 85
Figure 4.4 Average shortest path lengths versus grid size (log) 85
Figure 4.5 Throughput and saturation injection rate (IR) improvements of 3D-NoCs over two dimensional networks-on-chip (2D-NoCs) for 128 nodes connected as mesh (16×8 and $8 \times 4 \times 4$). 86
Figure 4.6 Throughput and saturation IR improvements of 3D-NoCs over 2D-NoCs for different network grid size. 86
Figure 4.7 Percentage of detected deadlocks for 64 nodes NoC configured as 2D mesh and 3D mesh with fully adaptive routing algorithm. 87
Figure 4.8 Percentage of energy consumed to resolve detected deadlocks to the total energy for NoC configured as 2D mesh and 3D mesh with fully adaptive routing algorithm. 87
Figure 4.9 3D stacked chip layout shows the input/output pins map 88
Figure 4.10 3D layout shows the topology of the implemented chip. 89
Figure 4.11 Schematic of 3D ring oscillator implemented on-chip 90
Figure 4.12 Snapshot showing the setup to test the chip. The chip is mounted in a special socket on top of a printed circuit board (PCB) then interfaced to the field-programmable gate array (FPGA) board using a SAM interface. 91

Figure 4.13 Examples of planar deadlock and 3D deadlock configurations with different sizes in the 3-tiers chip. 92
Figure 4.14 The start of TC computation and the deadlock detection signals 92
Figure 4.15 Ring oscillator output captured showing an oscillation of 448.91 MHz 93
Figure 4.16 (a) Frequency (log); (b) period (log) of the 3D-IC ring oscillator operation with different power supply values 95
Figure 4.17 The time delay to detect different-sized deadlocks in tier 1 and 2 in the 3D-NoC chip (error bars represent the maximum and minimum measurements). 96
Figure 4.18 Deadlock detection delay with scaling down the power supply voltage (error bars represent the maximum and minimum measurements). . . . 96
Figure 5.1 The shortest path in a weighted graph (bold line from S to D). It is clear this problem exhibits optimal substructure; a straight line indicates a single edge; a broken line indicates a shortest path between the two nodes it connects (other nodes on these paths are not shown). 101
Figure 5.2 A dynamic programming (DP)-network coupled to a 3D mesh NoC, drawing not to scale 104
Figure 5.3 The floorplan of the Intel 80-tiles chip fabricated at 65nm technology. 107
Figure 5.4 Side view of a typical integrated circuit (IC) package installed on a PCB showing the constituent layers [84]. 108
Figure 5.5 Dynamic compact model illustration: (a) resistance-capacitance (RC) model for Figure 5.4; (b) RC model for an example die with two architectural units (u1 & u2) and their connection to the upper and lower layers; (c) RC model for the heat-spreader layer, illustrating the four cardinal direction's lateral heat transfer paths. 109
Figure 5.6 Automated flow of the proposed tool for dynamic thermal optimization for 3D-NoCs 111
Figure 5.7 Temperature distribution of the 80-tiles chip subjected to *transpose* traffic scenario 113
Figure 5.8 Temperature occurrences over the 80-tiles chip with *transpose* traffic scenario 113
Figure 5.9 Maximum steady state temperature distribution of the 80-tiles chip subjected to *transpose* traffic scenario 114

Figure 5.10 Temperature variation over time for a selection of tiles interconnected as 8 × 10 2D mesh NoC and subjected to *transpose* traffic 115
Figure 5.11 Temperature distribution of the simulated 8 × 10 × 3 3D NoC subjected to *butterfly* traffic scenario 117
Figure 5.12 Temperature distribution over the 240-tiles chip with *butterfly* traffic scenario 118
Figure 5.13 Maximum temperature distribution over the 240-tiles chip 119
Figure 5.14 Temperature change over time for a selection of tiles under *butterfly* traffic. 120
Figure 5.15 NoC performance (Average delay versus Throughput) for *transpose* traffic 121
Figure 5.16 Maximum (max.) & minimum (min.) temperature for the hottest layer under *transpose* traffic. 121
Figure 6.1 The proposed dynamic programming-based run-time thermal management (DPRTM) for NoCs . . 128
Figure 6.2 Illustration of the routing paths in DPRTM system. 129
Figure 6.3 DP-network coupled to a 3D mesh NoC 130
Figure 6.4 Schematic design of 3D mesh NoC router with a DP-network. The computation unit is integrated to the local router to enable dynamic thermal-aware routing. The DP computational unit interconnects with other DP-units locating at adjacent tiles. 134
Figure 6.5 Realization of the DP computational unit 135
Figure 6.6 Maximum temperature of each layer for different routing algorithms under *uniform* traffic distribution. 137
Figure 6.7 NoC performance results for different routing algorithms under *uniform* traffic distribution. . . 138
Figure 6.8 Maximum temperature of each layer for different routing algorithms under *transpose* traffic scenario. 140
Figure 6.9 NoC performance results for different routing algorithms under *transpose* traffic scenario. . . . 141
Figure 6.10 Performance results for different routing algorithms under fixed throttled routers concentrated in the centre of the NoC. 143
Figure 6.11 The results of temperature regulation for a thermal limit of 70 °C for different run-time thermal management (RTM) methods 145
Figure A.1 Snapshot for the fabricated 3D stacked three-layer chip in the package after removing the top cover 156

Figure A.2 Snapshot from the digital signal analyzer showing the start of TC computation and the deadlock detection signal (detection time equal to 7.9 ns) 156
Figure A.3 Oscilloscope capture showing deadlock detection (loop size 4) 158
Figure A.4 Oscilloscope capture showing deadlock detection (loop size 6) 158
Figure A.5 Oscilloscope capture showing deadlock detection (loop size 8) 158
Figure A.6 Oscilloscope capture showing deadlock detection (loop size 10) 159
Figure A.7 Oscilloscope capture showing deadlock detection (loop size 12) 159
Figure A.8 Oscilloscope capture showing deadlock detection(loop size 14) 159
Figure A.9 Oscilloscope capture showing the output of the through layers (3D-IC) ring oscillator 160

LIST OF TABLES

Table 2.1 Deadlock handling strategies in packet-switching networks: a comparison 21
Table 2.2 Different vertical interconnect technologies: a comparison 24
Table 2.3 Different approaches to implement the on-chip networks 34
Table 2.4 Main featurtes and differences among the avilable on-chip networks simulators 36
Table 3.1 DES analysis of a TC-network for different network topologies 50
Table 3.2 TC-network improvement compared to timing based methods for different threshold values and traffic scenarios. 64
Table 3.3 Energy consumption (*mJ*) to drain 2 Mbytes of multi-media system data for different network loads and different deadlock detection methods 68
Table 3.4 Area and power contributions of the TC-unit and different timeout circuits to the total router area and power 70
Table 3.5 Percentage of detected deadlocked packets and percentage of total energy consumed to resolve them under *shuffle* traffic scenario with different deadlock detection schemes 76

Table 5.1 Performance metrics, maximum and minimum steady state temperature measurements (in C°) for different network traffic with and without DP-network 122
Table 6.1 Achievable performance metrics for different routing algorithms with different thermal limits under *uniform* traffic. 138
Table 6.2 Achievable performance metrics for different routing algorithms with different thermal limits under *transpose* traffic. 139
Table 6.3 The resulting network availability and the corresponding throughput different thermal limits for both RTM methods under *uniform* traffic. . . 144
Table 6.4 The resulting network availability and the corresponding throughput different thermal limits for both RTM methods under *transpose* traffic. . . 146
Table 6.5 Router plus DP-unit area and power synthesization result 147
Table 6.6 NoC plus DP-interconnects area and power result 148
Table A.1 Complete list of I/O pins map of the fabricated chip 157
Table B.1 The material parameters used in this work for the thermal simulation 162

LIST OF ALGORITHMS

3.1 Pseudo code to check the existence of a DES using TC computation 47
3.2 Pseudo code of the TC-unit computation 53

4.1 Floyd-Warshall algorithm to compute all-pairs shortest-paths from a directed weighted graph $\mathcal{G}$ 82

5.1 Pseudo code of the proposed dynamic thermal optimization routing using DP-network 105

6.1 Decide throttling level $TR_n(t)$ of the current node n_c . 131
6.2 Pseudo code for computing the routing directions performed by each DP-unit of the proposed DPRTM system 133

ACRONYMS

2D two dimensional

2D-NoC two dimensional network-on-chip

2D-NoCs two dimensional networks-on-chip

3D three dimensional

3D-IC three dimensional integrated circuit

3D-NoC three dimensional network-on-chip

3D-NoCs three dimensional networks-on-chip

ASIC application-specific integrated circuit

ASL application specific logic

AXI advanced extensible interface

BES best-effort service

CMOS complementary metal-oxide semiconductor

CMP chip multiprocessor

CPF coolest path-first

LTPF least throttled paths-first

CPU central processing unit

CRC cyclic redundancy check

CWG channel wait-for graph

DW-XYZ downward XYZ

DC David cell

DES deadlock-equivalence set

DMEM data memory

DOR dimension-ordered routing

DP dynamic programming

DPRTM dynamic programming-based run-time thermal management

DSP digital signal processing

DT distributed-throttling

DVFS dynamic voltage and frequency scaling

FIFO first in, first out

FPGA field-programmable gate array

GALS globally asynchronous locally synchronous

GS guaranteed service

GT global-throttling

I/O input/output

IC integrated circuit

ID identification

IMEM instruction memory

IP intellectual properties

IR injection rate

ITRS International Technology Roadmap for Semiconductors

LP linear programming

MEMS micro-electro-mechanical systems

MMS multi-media system

MIC many integrated core

MIPS microprocessor without interlocked pipeline stages

MPEG moving picture experts group

MPSoC multiprocessor system-on-chip

NI network interface

NMOS n-type metal-oxide-semiconductor logic

NoCs networks-on-chip

NoC network-on-chip

OCP open core protocol

OS operating system

PCB printed circuit board

PDA personal digital assistant

PIR packet injection rate

PMOS p-type metal-oxide-semiconductor logic

PTM predictive technology model

PVT process-voltage-temperature

QoS quality of service

RC resistance-capacitance

RLC resistance-inductance-capacitance

RF radio frequency

RO ring oscillator

RTL register transfer language

RTM run-time thermal management

SCC single-chip cloud computer

SoC system-on-chip

SRAM static random-access memory

TADW traffic-aware downward

TAVT thermal-aware vertical throttling

TC transitive closure

TDM time-division multiplexing

TSV through-silicon via

ATtAT average-throughput to average-availability

VC virtual channel

VLSI very large scale integration

VT vertical-throttling

XYZ-GT XYZ with global-throttling

Part I

Thesis Chapters

1

INTRODUCTION

1.1 MOTIVATION AND OBJECTIVE

The continuing advances in technology scaling over the past decades have led to important improvements in the portability, size and performance of electronic products, but with some real communication challenges. Increases in design complexity, system-wide synchronisation issues, larger process-voltage-temperature (PVT) variation, increased vulnerability to errors and increases in the global interconnect delay are all examples of the design challenges. These challenges led the very large scale integration (VLSI) designers to adopt a communication centric design with a systematic and structured approach to communicate in system-on-chip (SoC). This can be achieved through modularity, plug-and-play, intellectual properties (IP) core reuse and standard interfaces (*e.g.*, open core protocol (OCP) and advanced extensible interface (AXI)) with hybrid synchronous and asynchronous schemes (i.*e.*, globally asynchronous locally synchronous (GALS) systems).

The networks-on-chip (NoCs) became the choice of communication paradigm for SoCs and chip multiprocessors (CMPs) design to tackle many of the challenges associated with technology scaling. NoCs play an important role by providing a scalable, flexible and power efficient solution to integrate tens of IP cores in a single chip. For example, Intel 80-tile teraFLOPS [157] and Tilera 64-tile TILEPro [10] are multicore processor chips built on top of network-on-chip (NoC) communication fabric. Migrating from on-chip point-to-point and bus-based communication schemes to network schemes opens doors for research topics such as network topology selection, application mapping, data routing algorithms, fault-tolerant routing, quality of service (QoS), monitoring, debugging, *etc.* Some of these, such as topology selection and application mapping, can be tackled at design-time; while others, like adaptive routing and fault-tolerant routing algorithms, require design and run-time awareness.

In addition, shrinking feature size to the deep submicron technologies enables smaller and denser IP core design and thus allows increasing the number of cores integrated in a single chip. In terms of delay, however, scaling favours logics and short interconnects more than global interconnects for a constant die size, as shown in Figure 1.1. Therefore, data (messages or packets) transmitted using planar NoC with a large number of cores will suffer from bigger delay because of the increase of hop count and the global interconnects delay. The advent of the multi-level integrated circuit (IC) using three di-

mensional (3D) VLSI [135, 63, 106] opens a new prospect for design of on-chip networks by providing a new dimension to exploit novel geometric integration of silicon dies. A variety of vertical cross-die interconnection techniques are being developed [58]. In particular, the through-silicon via (TSV) [144, 20, 95] of three dimensional integrated circuit (3D-IC) connecting multiple die/wafer layers in a single chip provides opportunities to increase the integration capacity and also reduces the lengths of global interconnects. However, increases in yield and heat mitigation are major issues facing the 3D integration advancements. In particular, thermal issues are exacerbated by the advent of 3D integration. This will worsen the figures reported in [148] which state that over 50% of electronic product failures can be accounted for by thermal issues and hotspot formations.

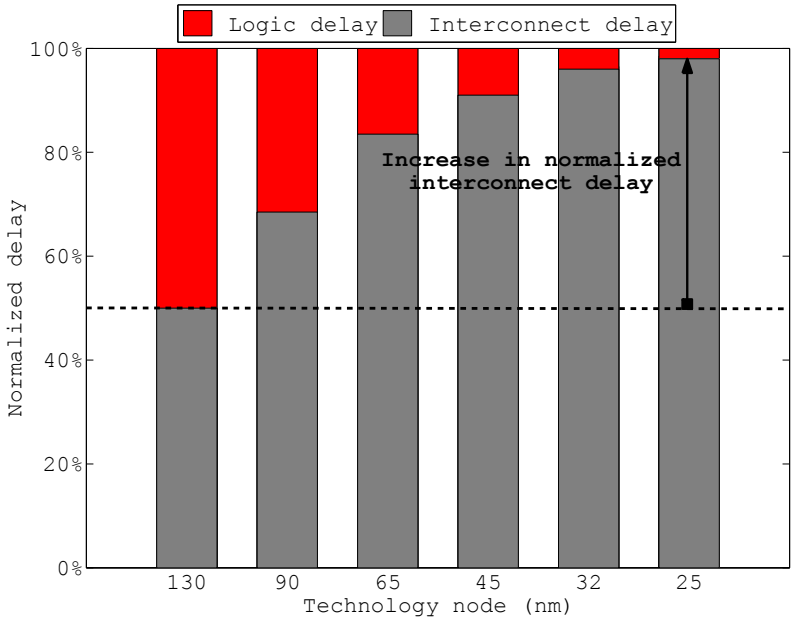


Figure 1.1: The trend of logic delay and interconnect delay with technology scaling (source data: ITRS [3]).

Moreover, on-chip system complexity grows significantly with continued increasing transistor density caused by aggressive technology scaling and the additional tightly coupled physical layers imposed by 3D integration. Therefore communication resources management in such complex systems becomes a major challenge. Examples of pervasive NoCs services that might require run-time management are dynamic packets routing, thermal diffusion, thermal management, fault-tolerant routing, managing the voltage and frequency islands, etc. This thesis argues that in-depth understanding of the NoC architecture is of vital importance to enable the designer to exploit the underlying architecture and find new efficient solutions for such challenges. The primary objective of this thesis is the exploration of

innovative methods that maximize the on-chip networks performance while complying to some design constraints.

This thesis also argues that with such an increase in the on-chip system complexity, design time methods may not be able to fully utilize the underlying hardware resources to dynamically adapt to the traffic, power and thermal constraints. The thesis aims to propose a design methodology that advocates the use of a run-time management infrastructure that can be realized using a distributed architecture which closely couples with the NoC infrastructure. This infrastructure can be defined and built to produce run-time solutions for different NoC based systems challenges. The work investigates several NoCs challenges, in particular: runtime deadlock detection, optimized thermal-aware routing and thermal-management of complex 3D multi-core systems. It can be extended using a similar methodology to tackle other run-time issues such as fault-tolerant routing, managing and optimizing the voltage and frequency islands.

1.2 STRUCTURE OF THE THESIS

The thesis is organised into seven chapters, as follows:

Chapter 2 provides on-chip network theory, architecture design and modelling to enable the reader to understand the motivation and the choices made in the subsequent chapters. It also reviews the emerging technologies that promise profound change in on-chip communication architecture. In addition, different methodologies and techniques for designing NoCs from various leading research groups and VLSI companies are reviewed.

Chapter 3 studies deadlock detection and recovery in networks-on-chip, as opposed to deadlock avoidance. It presents a new deadlock detection method that utilizes run-time transitive closure computation to discover the existence of deadlock-equivalence sets which imply loops of requests in NoCs. A distributed transitive closure (TC)-network architecture, which couples with the NoC architecture, is also presented to realize the detection mechanism efficiently. The parallel computation and network convergence properties of the TC network are discussed. Evaluation results using a cycle-accurate simulator and synthesis tools are presented. Detailed hardware realization architectures and schematics are offered.

Chapter 4 quantifies the advantages of using three dimensional networks-on-chip (3D-NoCs) compared to two dimensional networks-on-chip (2D-NoCs), using different design measures for different network sizes. Specifically, the average shortest paths length, network performance and deadlock formation rate are studied for 3D-NoCs

and 2D-NoCs. Moreover, the chapter presents the design and the chip fabrication results of the proposed 3D TC-network used for run-time deadlock detection. The prototype consisted of three tiers stacked in a single chip and interconnected using TSV technology.

Chapter 5 studies thermal variations and hotspot formations in the three dimensional network-on-chip (3D-NoC) stacked chip. It introduces an adaptive strategy to effectively diffuse heat throughout the 3D geometry. This strategy employs a dynamic programming network to select and optimize the direction of data manoeuvre in a NoC. This led to developing a tool chain, which is based on the accurate *HotSpot* thermal model and *SystemC* cycle accurate model, to simulate the thermal system and evaluate the proposed approach. Evaluation results using the developed tool chain are presented.

Chapter 6 proposes an adaptive, collaborative dynamic programming-based run-time thermal management (DPRTM) system for NoCs. It manages the routing workload dynamically to achieve thermal mitigation and control. Reactive and proactive schemes are interchanging to optimize the routing pathways depending on the critical temperature thresholds and the traffic developments. When the thermal limit is violated, DPRTM responds by throttling the routers where violation took place. The proposed DPRTM is evaluated and compared with other run-time thermal management (RTM) systems in terms of adaptation efficiency and thermal regulation.

Chapter 7 summarises the contributions of this thesis and directions for future work are identified.

1.3 STATEMENT OF ORIGINALITY

I have published a large amount of the work in this thesis in conference proceedings and journals. The materials have been explained in four major divisions, which are covered in separate chapters. Each chapter has an introductory section and related work section. The introduction in each chapter describes the rationale and contributions in further detail, while the related work sections present and discuss works pertinent to the issues addressed in the chapter. The main contributions of this thesis are listed as follows:

- Proposing a novel deadlock detection method that utilizes run-time transitive closure computation to discover the existence of deadlock-equivalence sets in NoCs. This detection scheme guarantees the discovery of all true-deadlocks without false alarms, in contrast with state-of-the-art approximation and heuristic approaches. A distributed TC-network architecture, which cou-

ples with the NoC architecture, is also presented to realize the detection mechanism efficiently [14].

- Introducing a deadlock-equivalence set criterion for detecting loops of packet requests and presenting a new deadlock detection scheme to discover the existence of deadlock-equivalence sets based on TC computation. Detailed hardware realization architectures and schematics are presented. The proposed method is rigorously evaluated using a cycle-accurate simulator and synthesis tools to demonstrate the effectiveness of the TC-network based detection method compared to the time-out mechanism. Experimental results confirm the merits and the effectiveness of the proposed method. It drastically outperforms timing-based deadlock detection mechanisms by eliminating false detections and, thus, reduces energy wastage in retransmission for various traffic scenarios including real-world application [19].

- Quantifying the effectiveness of using 3D-NoCs compared to 2D-NoCs using different design measures for different network sizes. The average shortest paths length, network performance (throughput and saturation injection rate) and deadlock formation rate all show a considerable improvement when adopting 3D-NoCs compared to 2D-NoCs [15].

- Introducing the design of TC-network architecture and its implementation, using three-layer 3D-IC. The vertical inter-unit communication is achieved by means of TSV. The three-layer chip prototype is fabricated using 150nm complementary metal-oxide semiconductor (CMOS) technology. Prototype testing results demonstrated the TC-network is effective and detects deadlock rapidly in a NoC platform [115, 116].

- Proposing a new adaptive strategy to effectively diffuse heat throughout the 3D geometry. This strategy employs a dynamic programming network to select and optimize the direction of data manoeuvre in a NoCs. This led to developing a tool chain, which is based on the accurate *HotSpot* thermal model and *SystemC* cycle accurate model, to simulate the thermal system and evaluate the proposed approach. The evaluation results demonstrated that the proposed approach can significantly diffuse the hotspots from a 3D geometry and maximum temperature can be reduced by 4°C [18, 17].

- Proposing a novel adaptive, collaborative dynamic programming-based thermal management system for NoC (DPRTM). The system manages the routing workload dynamically to achieve thermal mitigation and control. Routing in DPRTM adapts in two modes: coolest path-first (CPF) and least throttled paths-first (LTPF). The first mode is active while the chip works within the thermal limit

and aims to diffuse heat from the 3D chip geometry and prevent thermal hotspots for as long as possible. When the thermal limit is violated, DPRTM responds by throttling the routers where violation takes place. This action introduces irregularity in terms of communication performance in the network. Thus, DPRTM adapts to this irregularity by changing its routing strategy to give priority to the least throttled paths in order to minimize the performance impact of throttling. Since the proposed routing adaptation strategy is resilient to any throttling taking place, a distribute clock throttling strategy is employed in the DPRTM. The proposed DPRTM is rigorously evaluated and the results show that it overcomes other RTM systems in terms of adaptation efficiency and thermal regulation. Results show that DPRTM achieves up to 33% improvement in throughput compared to other RTM systems for 3D-NoC under the same thermal constraints [16].

2

BACKGROUND AND LITERATURE REVIEW

2.1 INTRODUCTION

In recent years, there has been an increasing interest in designing and developing SoC and CMP in order to deliver new levels of performance under strong time-to-market pressure while also achieving cuts in system costs. This requires the use of appropriate design methodologies, including incorporating a proper on-chip communication fabric to interconnect existing components in a plug-and-play manner. SoC is a single chip IC that integrates heterogeneous components, *e.g.*, general purpose microprocessors, digital signal processing (DSP), memories (static random-access memory (SRAM), Flash), application specific logic (ASL) and any other IP cores, in order to provide feasible solutions to a variety of design problems. On the other hand, a CMP IC consists of several homogeneous processor blocks to build a high performance processor. In recent years, industry chip vendors demonstrate an increasing concentration in multi-core products by releasing new processors with greater core counts. The state-of-the-art communication models to interconnect these SoC's and CMP's building blocks (IP cores) are either point-to-point or bus-based schemes [33].

However, the latest advancements in VLSI circuit design have heightened the need for different on-chip communication architecture to handle the increasingly large number of IP cores integrated in a single silicon chip. There is also an increased demand in bandwidth requirements of the on-chip communication fabrics to match the increase on the integrated IP cores and thus assist high core utilization. Moreover, technology downscaling at the deep submicron level decreases logic (transistors) delays but increases interconnect delays, in particular the global interconnect [76]. Global interconnects are used to transmit data across a chip. Therefore, it is becoming increasingly difficult to rely on the state-of-the-art on-chip communication schemes to fulfil different system requirements in terms of scalability, modularity, parallelism and synchronization while still complying with the time-to-market pressure. Thus provision for a communication-centric approach to design complex SoCs becomes vital [133, 126].

Recently, NoC emerged as a new on-chip communication paradigm to alleviate the limitations of the point-to-point and bus-based interconnection schemes. This chapter briefly introduces the basic concepts and theory behind the NoC paradigm in order that the reader can understand the motivation and the choices made in the context of this work. In the literature, there are prominent research efforts devoted to

addressing different on-chip network challenges and also a considerable number of different architectures have been proposed. A review for a variety of NoC architectures are presented and discussed in this chapter. The contributions of this chapter are as follows:

- Providing basic concept and theory behind the NoC's architecture, components, flow control, deadlock and livelock (Section 2.3 to 2.6)
- Reviewing NoC emerging technologies like three dimensional integration, optical interconnect and wireless interconnect (Section 2.7).
- Presenting a comprehensive survey of various NoC architecture and routing algorithms proposed by a number of leading research groups and VLSI design companies (Section 2.8).
- Surveying and presenting the different features of the available NoC simulator platforms

2.2 NETWORKS-ON-CHIP (NOCS)

NoC emerged as a new on-chip communication paradigm for SoC and CMP design. This new paradigm transmits packets between on-chip cores through a network of interconnected routers (a.k.a. switch) instead of using the classical point-to-point and bus-based interconnection, see Figure 2.1. NoC provides a scalable, flexible, high performance and power efficient solution to integrate hundreds of IP cores in a single silicon chip like SoC and CMP. Each of the IP cores can be an implementation of processor cores, memory modules, DSP blocks and embedded reconfiguration modules that consist of millions of transistors [141, 78]. In essence, NoCs are similar to interconnection networks for parallel computers with multiple processors. However, in the latter each IP is an individual chip and the communication takes place at the printed board level that integrated them. Whereas in NoC, all the components are integrated in a single chip and this make NoC design challenging under tighter area and power constraints. The past decade has seen rapid advances in the NoC field through an abundance of research papers [55, 133, 68, 35, 103], books [130, 90, 42, 36] and successful implementation projects [157, 59, 10].

2.3 COMPONENTS OF NOCS

An on-chip interconnection network consist of a router at every node, connected to neighbour via interconnects (on-chip wiring) and a network interface (NI) to bind each IP core to a router. The NoC main components (router, link and NI) are interconnected with IP cores to form a SoC or CMP. However, NoC has also been widely used as an

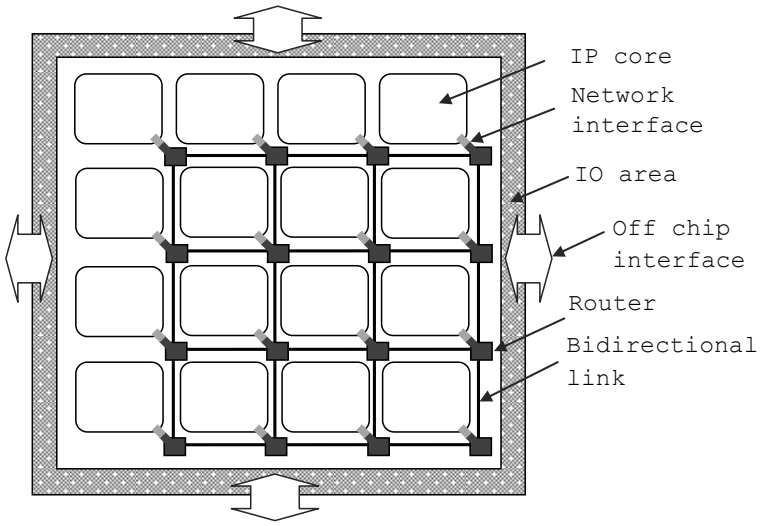


Figure 2.1: General network-on-chip (NoC) structure

abbreviation to include all on-chip networks [92]. The four on-chip components are commonly called 'Tile' in a tile-based chip design approach, where one tile is designed and then replicated to perform a complex on-chip system. The chip is divided into tiles, each with an IP core, NI, router and links.

The layout pattern of interconnections of all on-chip components is called network topology. Many topologies exist in the literature to constitute SoCs, the simplest being the shared bus and ring. Figure 2.2 shows different NoCs topologies along with the classical shared bus; some of these are general purpose and one is application specific. The selection of the appropriate topology will follow SoC and CMP design constraints such as performance, area, power, locality of traffic and quality of service. There follows a brief description of each of the on-chip components:

2.3.1 *Router*

An on-chip router is a block of logic that connects to other routers in the network through a number of links (interconnections) and to a local IP core (processing core) through NI. The router relays any received traffic ('traffic' refers to data transmitted through the network, also called packets or messages) towards the intended destination based on the address information stored on the received packet and its routing policy. The generic router microarchitecture consists of a set of input buffers, route computation unit, switch allocator, crossbar switch and a set of output buffers [90]. Other specific implementations may comprise also virtual channels (VCs) and pipelining different router functionality to improve throughput. However these extra functionalities will increase the router delay which is a primary contributor to the

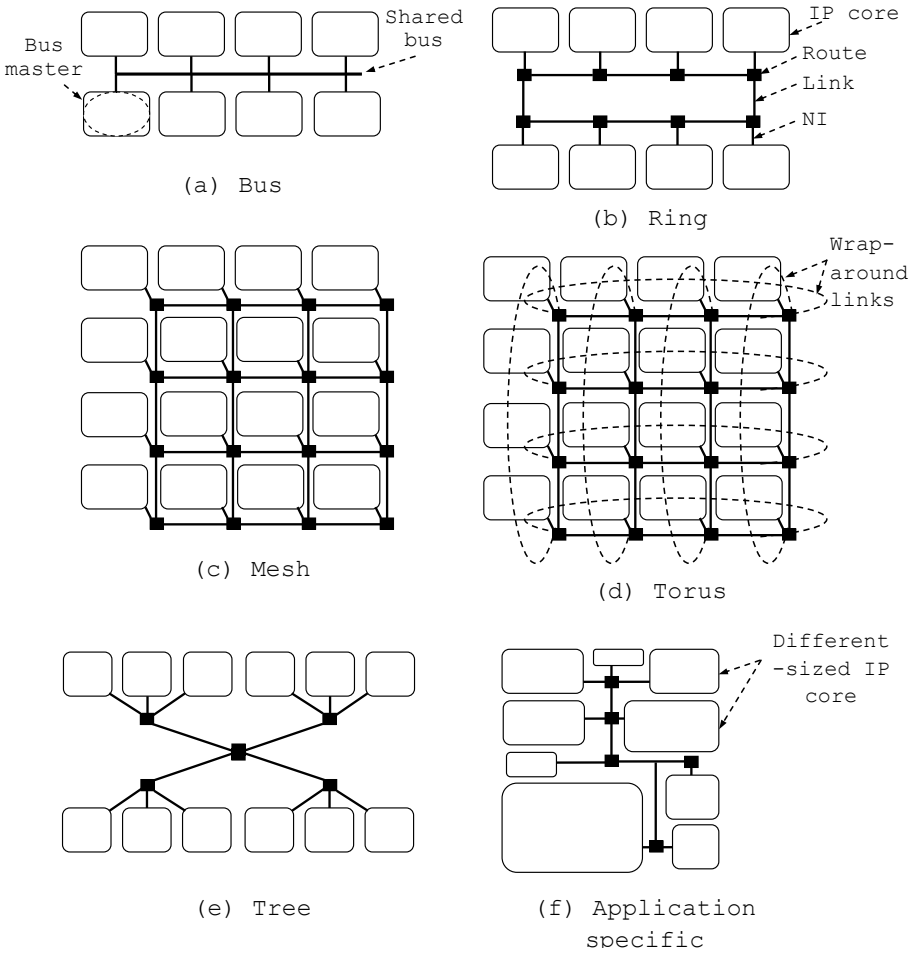


Figure 2.2: Different network topologies beside a bus topology [36]

overall NoC latency. Many router architectures exist [52, 61, 130, 92]. Figure 2.3 shows a block diagram for a typical credit-based router microarchitecture. Credits are basically the book-keeping of the local buffer state (occupied or empty) which is used to allocate buffer space. In the router, as received packets travel downstream on a link, credits travel upstream to allow access to buffers that have just emptied. The router building blocks in Figure 2.3 can be classified, based on their functionality, into two groups [52]. The first group represent the datapath and consists of input buffers, a crossbar switch and output buffers. The rest of the blocks represent the control unit and are responsible for administering the progress of packets through the resources of the datapath.

2.3.2 *Link*

A link is the interconnection medium (wiring) between neighbour routers in the network. To date, all NoC prototypes have used repeaters to improve signal reach of wires and conventional full-swing signals

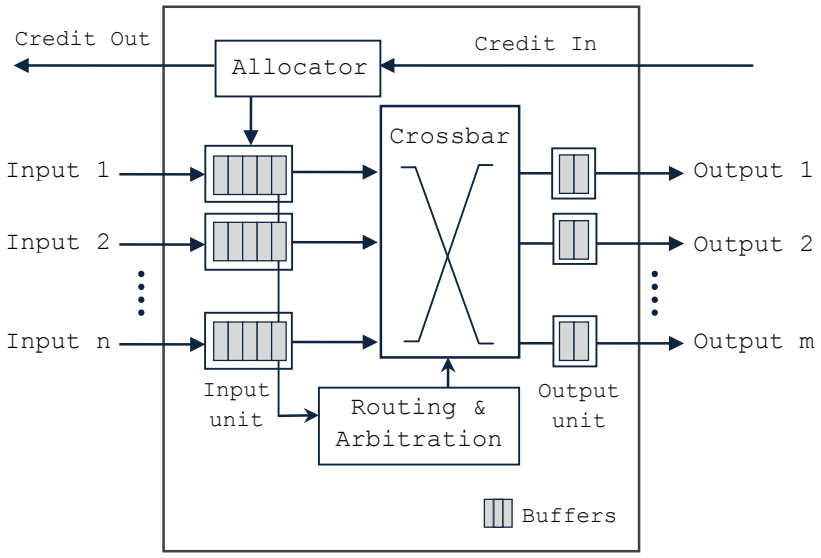


Figure 2.3: A typical network-on-chip (NoC) router microarchitecture [92]

[92]. As a result of the abundance of on-chip wiring resources, links tend to be wider in on-chip networks compared to off-chip networks. Several emerging interconnect materials, *e.g.*, nanowires, graphene nanoribbons, optical and wireless are listed in ITRS 2001 Edition [4] along with potential advantages and the primary concerns.

2.3.3 *Network Interface*

The NI serves as a gateway between the local IP core and other parts of the NoC. It is an interface wrapper that decouples computation (local IP core) attached in one face from the communication (local router) attached to the other face. NI enables passing messages (bidirectional) between routers and IP cores. This task involves performing two main operations [92]. If data is received from the local IP, the NI divides the data into suitable size packets (packetisation), adds the necessary header information and pushes them sequentially to the NoC through the attached router. On the other hand, if packets are received from the local router (coming from some source IP core through the NoC), then the NI starts assembling them after removing any unnecessary header information and sends the data to the local IP core.

2.3.4 *Intellectual Property Core*

An IP core in the electronic design context means a reusable block of logic or data which can be used as building blocks in the design flow of application-specific integrated circuit (ASIC) or field-programmable gate array (FPGA). A central processing unit (CPU), DSP block, memory, ethernet controllers and input/output (I/O) interface are all examples

of heterogeneous IP cores. The on-chip communication fabric (usually NoC) has to hold to standardized protocols so IP cores are designed in an abstracted way from the communication infrastructure. With this IP cores can be plug-and-play in the design to compose SoCs and CMPs [33].

2.4 NOCS FLOW CONTROL

Flow control is basically the mechanism that decides the allocation of network resources to the data passing through the network. A good flow control mechanism assigns network resources to the traversing data in an efficient way in order that higher bandwidth and lower average delay can be achieved in the network. The resources are buffers and links while the data could be messages or in smaller granularities such as packets and flits. A message can be segmented into one or more packets (depending on the size of the message and the maximum packet size allowed by the network); see Figure 2.4. Consequently, each packet is divided into fixed-length flits[1]. Each packet consists of head flit, body (or data) flit(s) and tail flit. The head and tail flits are the envelope required to deliver the letter (data flits) to the intended destination through the network. Some flow control mechanisms operate at message granularity, like *circuit-switching* techniques, and others operate at packet level, like *store and forward* and *cut-through* [92]. However, the popular flow control in the NoC community operates at the flit level, like *wormhole* and *virtual channels*. The following sections discuss these different flow control techniques.

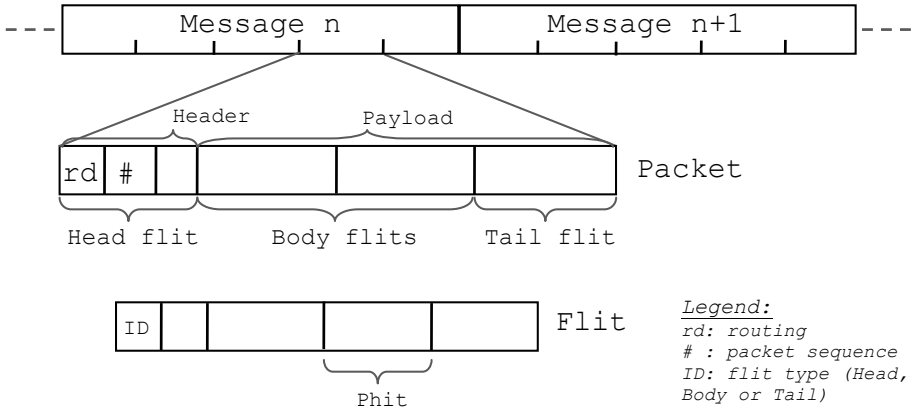


Figure 2.4: An illustration of message segmentation to packets, flits and phits. These different granularities of data representation are directly associated with the resource allocation of different flow control technique [52].

1 short for flow control units

2.4.1 *Message-Based Flow Control*

A typical example operating at the message level is the circuit-switching technique. The flow control in this technique works to allocate all the required links (channels) across multiple hops ahead of any payload transfer [90]. This is equivalent to establishing a path between any communication pair (source and destination) which is accomplished by sending the packet header first. Once a path is established, the source can start sending one or more packets along the path to the destination. When the data transfer is finished, the path is torn down by deallocating the reserved channels. One advantage of circuit-switching is that it requires no overhead for packetisation, packet header processing or packet buffering and thus savings can be achieved in terms of area and power. Conversely, the time to setup a path in this technique has a very adverse effect on the overall network performance (throughput and latency) as during this time the resources will be reserved, but used neither for data transfer nor are available to be used by other clients. However, apart from the path setup face, the hop-by-hop delay to acquire channels is avoided. The setup face delay time can be amortized if the data need to be transferred is sufficiently large. In general, circuit-switching suffers from poor network bandwidth utilization [36, 68].

2.4.2 *Packet-Based Flow Control*

In packet-based flow control, the network's resources are allocated at the level of the packets. In contrast to circuit-switching, this will increase network bandwidth utilization. However, in packet-switching flow control, each message must be segmented into suitably sized packets (packetisation). These packets are not necessarily the same size and each of them will traverse the network towards the target destination independently. Therefore, there is a possibility of the destination receiving packets in the wrong order. This will require an extra hardware overhead at the destination to ensure ordering. In addition, packet-switching requires sufficient buffers in each channel at each node to store traversing packets. These requirements are difficult to meet on NoC with tight area and power budgets. There are two types of packet switching flow control [92]; *store and forward* and *cut-through*.

2.4.2.1 *Store and Forward*

The store and forward flow control mechanism implies that each router waits to receive a full packet before forwarding it to the next hop in the network. Thus, each router channel must be equipped with a number of buffers sufficient to store a full packet. In addition to the higher area and power budget required to implement such a mechanism on

NoC, it causes a considerable delay at each hop if the packet size is large.

2.4.2.2 *Virtual Cut-Through*

The virtual cut-through flow control reduces the delay overhead at each hop by allowing a packet to proceed to the next router before receiving the full packet at the current router. However, a packet cannot be forwarded by the current router to the downstream router unless the latter has enough storage to hold the full packet [92]. Therefore, bandwidth and buffers are still allocated at the packet granularity. Interestingly, Intel single-chip cloud computer (SCC) prototype uses this kind of flow control [141, 143].

2.4.3 *Flit-Based Flow Control*

In the flit-based flow control mechanism, the network's resources are allocated at the flits level. In contrast to packet-switching, this will increase network bandwidth utilization and reduces the storage requirements in each router. Allocating resources at the flits level allows the on-chip router to comply with the tight area and power budgets of on-chip network. This made it very popular in different NoC implementations. The flow control at this level is called wormhole.

2.4.3.1 *Wormhole*

In wormhole flow control, network resources allocation takes place at the flits level. This makes it feasible for on-chip networks as it is possible to use a relatively small number of flit-buffers in each router even for large packet sizes. The packets in wormhole flow control are divided into smaller flits. A header flit is routed first and the rest will follow it in a pipeline manner, allowing a packet to occupy several channels simultaneously [129]. Thus, wormhole reduces the number of buffers required in each router, a desirable feature for NoCs. However, wormhole makes networks more prone to blocking and deadlock [112, 124] as relatively long packets will span over several hops. If a long packet is blocked during its flight to the destination, many physical resources will be blocked and consequently cannot be used by other traversing packets [52].

2.4.3.2 *Virtual Channels*

To overcome the blocking problem of the wormhole flow control, each physical channel is associated with several VCs (flit buffers). This allows other packets to use the physical channel even if some other packets are blocked [53]. Similar to wormhole, the resources allocation occurs at the flits levels and thus a header flit is routed first and the rest follow it in a pipeline manner. Unlike wormhole, associating each

physical channel to a number of logical channels (virtual channels) allows packets to pass blocked packets by making use of the idle channels bandwidth. VCs are very important for network optimization and management purposes such as throughput improvements, QoS, prioritization, head-of-line blocking mitigation and deadlock avoidance [156]. However, implementing VCs can be costly due to the storage requirements (number of buffers) per router and the necessary control logic circuit.

2.5 ROUTING ALGORITHMS

Given a particular network topology, routing is responsible for choosing a path between any two communicating nodes (source-destination pair). Selection of the routing algorithm is crucial to achieving good network performance. An effective routing algorithm must take the following into consideration: load balancing, throughput, latency, fault-tolerance, deadlocks, livelocks and, above all, the limited resources available on chip (area and power budgets). Despite that fact that an abundance of routing algorithms have been proposed in the literature [52, 92], the dimension-ordered routing (DOR) algorithm (also known as *XY* routing for two dimensional (2D) mesh) is the most commonly used in NoCs because of its simplicity. In DOR routing a message navigates the network dimension-by-dimension. Routing algorithms can be classified into three kinds: deterministic, oblivious and adaptive [52]. This is based on how the routing algorithm chooses between the available set of paths ($P_{S,D}$) from source node (S) to destination node (D).

2.5.1 *Deterministic*

Regardless of how many paths are available and the network (links and routers) status, deterministic routing always selects the same path between a given source and destination pair. DOR and source routing are typical examples of deterministic routing algorithms [92]. These kinds of algorithms perform a poor load balancing over the network if the traffic is unbalanced. In spite of this, DOR routing is commonly used because it is easy to implement and can easily guarantee freedom from deadlock.

2.5.2 *Oblivious*

This kind of routing algorithm sends messages (or packets) from S to D using different paths; however, it is unaware of the network's present status (congested, faulty). A routing algorithm that distributes the traffic uniformly across all available paths ($P_{S,D}$) is an example of oblivious routing [36].

2.5.3 *Adaptive*

As the name implies, adaptive routing adapts to the network's state and accordingly selects a path to route a message. Thus the network's state is the criterion for decision making. Different adaptive routing algorithms have been proposed [52, 61] and mainly they differ in which network's state they take into consideration. For example, for a fault tolerant design the link/router state (up or down) matters most, while for performance driven design the available resources (buffers) in the downstream routers are important. Other factors, such as load balancing, energy saving, heat diffusion, power supply noise minimization or a combination of these, could be the route selection criterion for the adaptive routing. Moreover, employing temporal and spatial information of these factors could yield different outcomes and thus it is the responsibility of the designer, constrained by the application and the resources, to select one or more factors (network's state) to optimize his/her routing algorithm.

2.5.4 *Minimal vs Non-minimal*

Routing can also be categorized as minimal and non-minimal algorithms [52]. Minimal routing means that the algorithm always chooses the shortest path between sender and receiver. On the other hand, non-minimal routing algorithms can choose longer paths (in terms of number of hops) to route messages which may increases latency. However, if part of the network is congested and the routing algorithm chooses a non-minimal path to avoid the congested region this will yield better load balancing and might decrease the overall latency [52].

2.6 DEADLOCK AND LIVELOCK

2.6.1 *Deadlock*

Whenever there are two or more competing actions and each waiting for the other to finish the configuration called a deadlock. A deadlock is an abstraction level concept. It could take place at different hardware and software design levels, *e.g.* operating system level, parallel computing, distributed systems, telecommunication systems, *etc*. This work, however, constrains on the deadlock that take place at the network level. Deadlock may appear in networks that allow unrestricted use of resources. Deadlock occurs when a set of consumers, packets or connections, cyclically wait for each other to progress [52]. If this kind of dependency forms a cycle, the network is deadlocked. Figure 2.5 illustrates a simple single cycle deadlock situation in a flit-based flow control network. There are eight packets within a 2D mesh network with minimal adaptive routing. In this example, P_2 is holding all the

channels needed (Ch_2 and Ch_4) and progressing towards its destination (node 2), while the other packets in the network are at a standstill, hence each of them has acquired some of the channels required and are still waiting for other channels occupied by other packets. Among these, P_1, P_5, P_7 and P_8 are forming a deadlock cycle which implies a loop of packets' requests, where the packets are waiting for each other in a cyclic way. These deadlocked packets cannot reach their destinations unless the network is recovered from the deadlock situation. As a consequence of the deadlock, packets P_3, P_4 and P_6 are also blocked in the network. It is important to distinguish between deadlock and blocked packets. In contrast to the blocked packets, recovering a single packet from the deadlocked packets is sufficient to break the cycle [61].

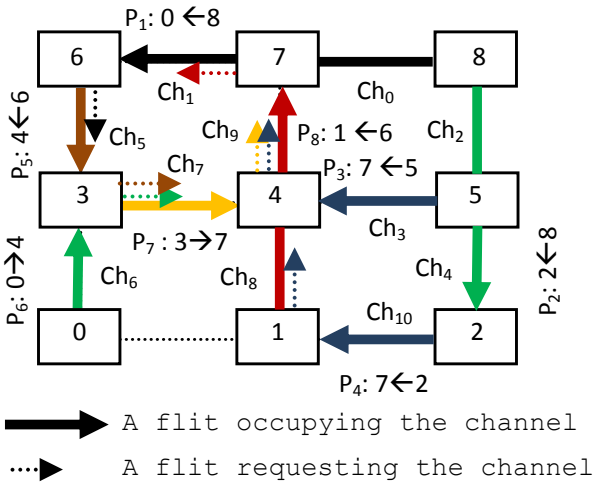


Figure 2.5: A deadlock cycle in a 2D mesh network

It is crucial to remove deadlocks when implementing an interconnection network as they may paralyze the network and increase packet blocking dramatically. In packet switching networks, there are two strategies to deal with deadlocks: avoidance and recovery [52].

2.6.1.1 *Deadlock Avoidance Strategy*

In deadlock avoidance, resources are granted to the propagating packets in a way that the overall communication channels are deadlock free. There are two main techniques to implement avoidance, one being to use any variation of the turn model which prohibits the routing algorithm from making certain turns in the network. A typical example is the DOR routing which restricts four turns out of eight possible turns in 2D mesh networks. Improvement to this is the partially adaptive routing that prohibits only two turns [65, 47]. This technique avoids deadlock by limiting the routing alternatives which makes it not always able to avoid traffic congestions. It also diminishes fault

tolerance routing capabilities. Despite these, partially adaptive routing is widely used in on-chip networks due to its simplicity.

The second technique relies on using virtual channels. Proposed by Dally [52], it is also known as the strict ordering of resources. The idea is to allocate resources (i.e., VC buffers) in some order (ascending/descending). This method has been generalized by Duato [61] which basically ensures a full adaptive routing algorithm in the network using one class of resources while the second class is used as escape channels. In general, this kind of avoidance technique increases the number of VCs required which unfavourably affects router speed [46]. Although this kind of avoidance is very popular in interconnection networks, it has never been popular in the NoC community.

2.6.1.2 *Deadlock Recovery Strategy*

Deadlock recovery strategy implies resources are granted to packets without any restrictions and check. Consequently, deadlocks may form in the network at any time and efficient detection and recovery mechanisms are required to quickly identify and recover from any possible deadlock. Detection is usually implemented in a distributed way using a timeout mechanism [124, 140, 105]. It utilizes a heuristic scheme to monitor the activity of each channel at each node in the network. In its simple form, a packet is suggested to be in a deadlock if the channel has been inactive for more than a given threshold time value [22]. The minimum hardware components to implement such a scheme are: a counter, comparator, a latch for each physical channel and, in the case of a programmable threshold value, a register to store the threshold value. The threshold value could be any value between 2 to 1024 as reported in [124].

Correct detections of this mechanism are very sensitive to network load and message length transmitted over the channels. Therefore it is difficult to find an optimal value for the threshold time used in this technique. There is a trade-off between the number of false detections and the speed of detection, while both of them remain crucial. Several techniques have been proposed to reduce the number of false deadlock detections [124, 140, 104, 105]. As a general comment, all these works are based on the timeout mechanism and pose a difficulty in tuning the threshold value (i.e., select a unique threshold value for different traffic and network loads). Therefore, a method which accurately detects deadlock without false alarms is required. A method which quickly detects deadlock independent of the network's traffic and load is desirable. Chapter 3 investigates this further.

Regarding the topic of deadlock recovery, there are two approaches to implement it. These are different in how they deal with the packets suspected to be part of the deadlock, to break the cycle dependency. A regressive recovery is basically based on the abort and retry mechanism [96] which kills the suspected packets. A progressive recovery,

however, uses additional hardware to bypass the suspected packets to their destination sequentially [22] or concurrently [23]. Another kind of progressive recovery is a software based recovery technique [123] in which the suspected packets are ejected into a temporary node(s) and re-injected, after a time out, to progress to their destinations.

Table 2.1 presents a detailed comparison between the two popular deadlock avoidance techniques and the deadlock recovery technique.

Criterion	Deadlock Avoidance 1 (using resource classes or restricted routing function)	Deadlock Avoidance 2 (using routing sub-functions)	Deadlock Recovery
Strategy	Conservative	Less conservative	Optimistic
Routing constraints	Use strict ordering of the resources or restricting the routing algorithm so that in either case there will be no cyclic dependencies between network channels.	Reduce the routing algorithm restrictions by allowing cyclic dependencies between channels while providing some escape paths to avoid deadlocks	Allow the use of unrestricted fully adaptive routing algorithm
Additional hardware	Additional hardware is required to apply the routing constraints and prevent the cyclic dependencies.	Required dedicated resources to provide escape path(s), one or more VC and it requires two different routing functions; a main and sub routing functions for the network and escape paths respectively. The topology dictates the required number of VCs	Required deadlock detection and recovery circuitry
Performance	Produces non or partially adaptive routing algorithms, *e.g.*, *dimension-order* [52], turn-model routing, *etc*.	Could outperform the avoidance strategy 1 by increasing the routing flexibility in the network [52]. However, the escape paths are still restricted in terms of routing	Produces true full adaptive routing with no restrictions in the routing algorithm. Potentially outperforms the avoidance techniques [124, 140, 105].
Network saturation	Not allowing cyclic dependencies to be formed and thus no severe network performance degradation close to network saturation.	The main-routing function allows cyclic dependencies to be formed in the network and thus might produce severe performance degradation when the network is close to saturation. This could happen if blocked packets are forming cyclic dependencies in the network faster than they are resolved using the routing sub-function for the escape paths. [124, 140].	Allowing deadlock to occur, as it has been shown to be an infrequent event if sufficient routing freedom is provided [124]. However, deadlocks formation rate increase close to network saturation. If deadlocks forming rate is higher than the recovering rate, severe performance degradation could occur [124, 140].
Faulty link and/or router	Deadlock-prone if a static fault arises (link/router) and extra measure is needed to ensure deadlock-freeness in such a case.	Deadlock-prone in the case of a static faulty link/router taking place in the network and hence extra measures are needed to ensure deadlock-freeness in such a situation.	Network supports a recovery mechanism to remove any deadlock, *e.g.*, any deadlock introduced by a static fault or by the packet dependency cycle.

Table 2.1: Deadlock handling strategies in packet-switching networks: a comparison

2.6.2 *Livelock*

This is a situation in the network where a packet is continuously moving but never reaches its intended destination. This destructive situation can only happen with non-minimal adaptive routing algorithms. However, it can be avoided by limiting the number of misrouting allowed to each packet traversing the network [61]. Minimal adaptive routing can be perceived as the algorithm that allows zero misroutes to each packet. One important reason to use the non-minimal routing is to avoid a faulty link or router in networks. Another important reason is to balance the load across the network when it is subjected to unbalanced traffic. In this study (Chapter 6), a non-minimal adaptive routing algorithm is proposed and used to diffuse (balance) the heat generated in the chip and to avoid throttled network resources. It has been shown in [98], if the minimal paths are given preference over the non-minimal ones in the routing algorithm, the probability of not reaching the destination is approaching zero.

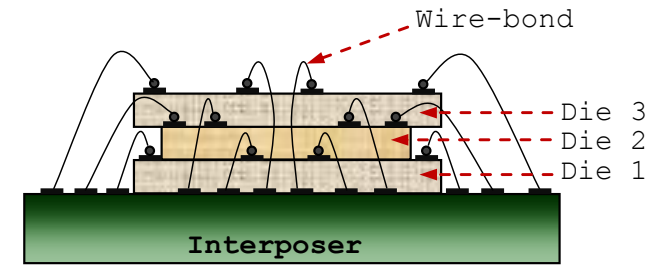
2.7 NOCS AND THE EMERGING TECHNOLOGIES

This section reviews the new emerging technologies that promise a fundamental change in the classical on-chip interconnection methods and means. It focuses on three dimensional integration, optical interconnect and wireless interconnect.

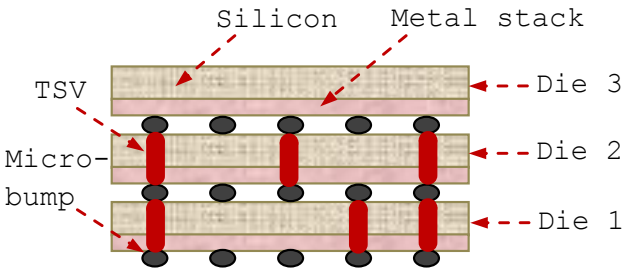
2.7.1 *Three Dimensional Integration*

Three-dimensional ICs present an attractive design approach to continue Moore's law of doubling the IC functionality to reduce system size and to enable heterogeneous IC integration. Previously, wire-bonding connects several dies in 3D stack was the only viable technology to achieve 3D-ICs, such as stacking memory dies on processor dies in cell phones, PDAs and flash cards [58]. This kind of integration provides small size products, but with little or no advantage in terms of minimizing interconnections delay among tightly coupled physical dies. Recent technological advances, such as micro-bumps and TSVs, offer new ways to connect dies in a cost-effective manner while exploiting the close proximity of the stacked dies. Figure 2.6 illustrates the three mentioned methods of 3D integration and Table 2.2 compares them in terms of manufacturing, performance gain and limitations.

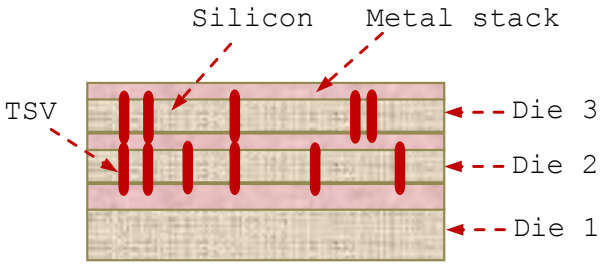
These 3D integration technologies open a new dimension to exploit innovative geometric integration of silicon dies [58] which can provide attractive feature like smaller form-factor, higher transistor density, increased performance and heterogeneous integration. In particular, the TSV [144, 20, 95] of 3D-IC technology provides opportunities to increase the integration capacity and also reduces the lengths of global



(a) Wire bonded



(b) Micro-bump (face-to-back)



(c) TSV

Figure 2.6: Different vertical interconnect technologies: an illustration

interconnects. This makes it the most appealing design approach for the research community [144, 149, 20, 95]. Decreasing the interconnect length is a very important feature that increases the NoCs scalability to connect thousand of IP cores. It was reported in [93] that using 3D architecture can achieve 50% reduction in global interconnect length compared to similar designs using a single layer 2D planar, combined with comparable reduction figures in delay and power. Chapter 4 investigates the merits of adapting the 3D-IC methods in designing future SoC and CMP based NoC by comparing it with the 2D planar design. The development of TSV based approach faces two main challenges, the first being how to increase the chip yield and the second, how to mitigate the heat generated inside the stacked layers? The latter is considered in Chapter 5 and Chapter 6.

Feature	Wire bonded	Micro-bump	Through-Silicon Via
Level of integration	Die	Die	Wafer
Technology used	Simple existing technology	Existing technology	New relatively complex technology
Manufacturing	Already in use in commercial products (volume production)	Already in use in commercial products	There exist some commercial products but mostly still under research and development
Vertical interconnect delay	High	Medium	Low
Density of vertical interconnect	Low	Medium	Medium to High
Integration limit	Restricted by assembly process	Restricted by thermal-limit and assembly process	Restricted by thermal-limit and yield
Performance and feasibility	High power, low speed and high manufacturing cost in high-quantities	Provide a compromise between performance and manufacturing cost	Low power, high speed and low manufacturing cost in high-quantities

Table 2.2: Different vertical interconnect technologies: a comparison

2.7.2 *On-Chip Optical Interconnect*

On-chip networks provide highly-scalable throughput when connecting large numbers of IP cores. However, with the increase in number of cores, the packets latency is increased and could become unpredictable. Thus, optical on-chip communication has been explored to tackle the latency challenge and also increase the communication bandwidth and reduce the power consumption [29, 131]. It is based on optical interconnections and optical routers [73]. These are become viable only because of the developments in nanoscale photonic devices [27, 163]. Introducing optical interconnects is expected to yield the following advantages: wide bandwidth, low power consumption, reduces crosstalk and removes electromagnetic interference. However, on-chip photonic processing and buffering are a challenging task and require some overhead due to electro-optical and opto-electric signal conversion. Moreover, there is a research direction [108, 164] that advocates the use of hybrid *photonic-electric* on-chip networks in order to secure the advantages and alleviate the disadvantages of each approach. In general, all these design techniques and technologies are still in the research and development phase.

2.7.3 *On-Chip Wireless RF Interconnect*

In another effort to increase the on-chip aggregated bandwidth and reduce packets' transmission latency, wireless radio frequency (RF) interconnect is proposed [43]. The main idea is to transmit electromagnetic wave instead of voltage signalling. Information is modulated onto a carrier wave using one of the modulations schemes. Binary phase-shift-keying is a popular modulation scheme for on-chip communication [44]. It simply encodes the binary data by changing the phase of the carrier wave between 0^o and 180^o. Wireless RF on-chip interconnect is a promising research topic, potentially yielding the following advantages [43]: increased bandwidth, provision of a multi-I/O service and on the fly reconfigurability. However, there are some challenges and constraints such as: additional circuit overhead, power consumption and on-chip signal filtering. As a general comment, on-chip wireless communication is an important future trend in VLSI circuit design and requires considerable efforts in research and developments before being realised in commercial products.

2.8 NETWORKS-ON-CHIP: A REVIEW

This section presents a broad picture of the research conducted by different leading research groups in the field of NoCs. The works related to the NoC's challenges addressed in this study are given in the respective chapters. This section is by no means a complete overview

of the NoC field. The interested reader is directed to the following NoC's research survey papers: T. Bjerregaard and S. Mahadevan [40], Marculescu et al. [121] and Marculescu and Bogdan [122].

2.8.1 *The Emergence of NoCs*

In the past decade, several researchers have advocated the use of on-chip networks to build future versatile SoCs (e.g., Dally and Towles, 2001; Benini and De Micheli, 2002; Kumar et al, 2002). Dally et al [55] first proposed replacing global on-chip wiring with an on-chip network that routes packets between different on-chip modules [55]. They suggested the proposed new design methodology would improve performance, increase modularity and make the design more structured over the conventional global wiring between different on-chip modules (tiles). In this study, the authors estimated that the on-chip router would require 6.6% of the tile area and a portion of wiring layers for the on-chip interconnects. The study also identified three areas for further research and investigations: the type of topology, flow control method and circuits design. In [33], a high level design perspective for building SoC using a layered-micronetwork design methodology is presented by Benini and De Micheli. They suggested a new paradigm that utilised the mature models, techniques and tools from interconnection networks (used for parallel computers with multiple processors) field and applied them to SoC design. In their study [33], they expected the increasing complexity of future SoC in terms of global-wire delays, performance demands, energy consumption, synchronization between different clock domains and devices reliability would be likely to dictate adapting the new paradigm (networks on chip) to design future SoCs. Kumar et al [99] presented a design methodology for a network on chip. The design methodology consisted of two phases to create systems: first derive the necessary network architecture from a template and, secondly, map the application to the architecture. Rather than using general terms, their NoC template is based on a 2D mesh topology and uses packet switching flow control.

NoCs is an emerging field, leading to different research groups contributing to a wide range of aspects such as topology exploration, routing algorithms, flow control, fault tolerance, application mapping, simulation, prototyping, implementation and surveying. Examples of NoC proposals that relate to different network topologies are: SPIN [70] which uses a fat-tree topology; CLICHÉ [99] uses a 2D mesh topology; SoCIN [166] uses a 2D torus topology and also a folded torus topology is proposed in [55]. All these studies utilize wormhole flow control in their NoC proposals. Although different topologies are studied in the literature, a 2D mesh is the most popular in the research community. Moreover, many VLSI leading companies adopt 2D mesh for their NoCs based products and prototypes [88, 143, 10].

2.8.2 *Routing Algorithms and Flow Control*

In terms of routing algorithm and flow control technique, there are significant works published in the parallel and distributed computing field [52, 61] which are applicable to on-chip networks. The main focus of these works is upon the design of a high-performance, reliable and flexible router architecture. This includes the development of congestion-aware, deadlock-free and livelock-free routing algorithms. In particular, the deterministic *XY* for 2D mesh topology [147] and the partially adaptive turn-model for mesh and hypercube topologies [65] are the popular routing algorithms used in the NoC community. These algorithms are simple to implement and do not need any virtual channels to avoid deadlock. The *XY* routing simply routes a packet along the horizontal direction (X-axis or East and West) and then along the vertical one (Y-axis or North and South). In essence, this routing algorithm prohibits four out of eight possible turns in 2D mesh (see Figure 2.7). This algorithm can also easily be extended to higher dimension meshes by routing a packet along each dimension separately. Glass and Ni [65] observed that prohibiting half of the possible turns in the 2D mesh is redundant and it is possible to guarantee deadlock-freeness by removing only two turns out of eight possible turns. Thus, their work proposed three different partially adaptive routing algorithms, namely *west-first*, *north-last* and *negative-first* (see Figure 2.7). All of these are known as the turn model for adaptive routing. For instance, the *negative-first* prohibits *north-west* and *east-south* turns to avoid deadlock in 2D mesh. The work also extended to higher dimensional meshes and hypercubes [65].

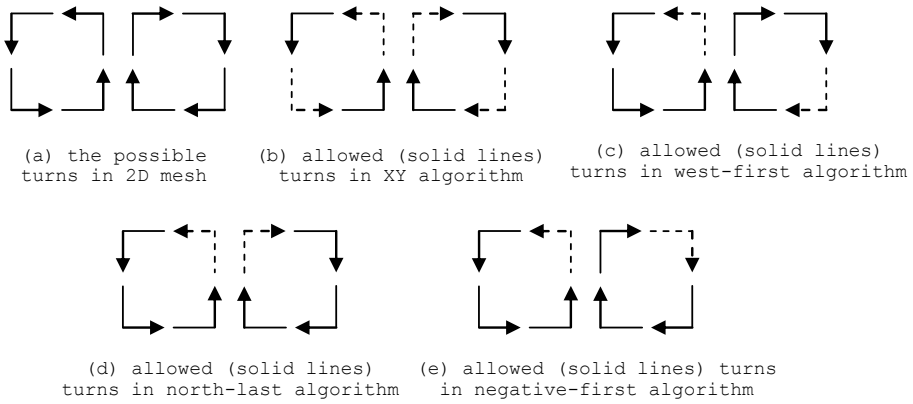


Figure 2.7: The possible eight turns in 2D mesh beside the allowed turns in different routing algorithms [65]

The turn model is used later by Chiu [47] to propose his *odd-even* adaptive routing algorithm. The algorithm can be perceived as combining two variations of the turn model presented in [65], e.g., *west-first* and *negative-first*, and alternates between them depending on the location to avoid deadlock. The resultant routing algorithm has higher

degree of adaptiveness compared to the turn models; however, it is applicable only for meshes. A routing scheme called *DyAD* is proposed in [81] for NoCs which dynamically changes between *XY* and *odd-even* routing algorithms based on the network congestions' conditions. *DyAD* is proposed to combine the benefits of the deterministic routing at low injection rates and the adaptive routing capabilities at high injection rates to avoid congestions. Different variations of the turn model routing and deterministic routing exist in the literature [107, 26, 110]. The key difference among these works is either the criteria to switch between the routing modes or the kind and location of information probed from the network to select a routing direction among those given by adaptive routing.

The above-mentioned routing mechanics can also be called algorithmic routing [52], where the routing directions are computed using a certain algorithm. In general, there is a trade-off between algorithm simplicity and routing adaptiveness. Simple algorithms are easier to realize in hardware which is favourable in NoCs. However, easy algorithms like the XY routing could yield a very poor load balancing and premature network saturation compared to more sophisticated adaptive routing (*e.g.*, [26]). In general, algorithmic routing is restricted to regular topologies, which is not always the case with NoC design.

On the other hand, table-based routing can be used for any network topology. Table-based routing uses memory or registers to store the routing information either at the source node or at each hop (distributed). Source routing is commonly used for deterministic routing and it has the advantage of being simple, scalable and fast [52]. Conversely, distributed table-based routing is more appropriate for adaptive routing and it has the advantage of being more economic in terms of storage requirements [52]. In NoCs community distributed table-based routing appears to be more popular than source table-based routing. For example, the work presented in [132] proposes a methodology for the design of an application of a specific adaptive routing algorithm for NoC and realizes it using distributed table-based routing. In this kind of work, the routing information is computed and stored in the distributed tables at design time. Run-time routing tables update mechanisms to accommodate dynamical network changes also exist in the literature. For instance, the work in [117] proposes a parallel computational architecture that provide run-time shortest paths computation to implement adaptive routing for NoCs. The proposed realization of the routing is also based on distributed tables. Distributed table-based routing is also adopted in this thesis (Chapter 5 & 6) to implement the proposed thermal aware routing.

2.8.3 *Different Architecture Philosophy*

Various NoC architectures proposed by number of leading research group and VLSI design companies exist. These include Nostrum [6], Hermes [21], Xpipes [51, 34], MANGO [39], RAW [8], SpiNNaker [9], Æthereal [68], Intel [59, 157] and TILERA [10]. The choice of architecture is constrained by the target application, die size, power and performance. In the following, a review of different on-chip networks approaches is presented. Table 2.3 also summarizes the features and highlights the main differences among different on-chip networks platforms proposed in the literature along the lines of the NoC used in Chapters 3&4 and Chapters 5&6.

2.8.3.1 *Nostrum: The KTH Royal Institute of Technology Approach*

Nostrum [6] is a research project that studies and develops a NoC platform (called Nostrum) for SoC design. Nostrum NoC can be briefly described as a packets switching on-chip network that adopts 2D regular mesh topology. It utilizes adaptive deflective routing which could be useful to manage faults and congestion in the network. Deflection routing is an approach to resolving packets contention (two or more packets competing on the same output) without the need for buffering. This can minimize the area and power of the on-chip router; hence no, or minimal, buffers are required. Nostrum offers two QoS; best effort and guaranteed latency services. The potential SoC applications for Nostrum are multicore processors and multimedia applications.

2.8.3.2 *Hermes: The Pontifícia Universidade Católica Approach*

The Hermes project is a parameterized network infrastructure targeted at implementing low area cost NoC [127]. Hermes implements a wormhole packet switching flow control and is also able to support VC's implementation with different flit width and buffer depth. There are various implementations of Hermes NoCs [21] (Hermes, Hermes TB, Hermes TU, Hermes SR and Hermes CRC). Some of these implementations differ in terms of network topology (2D mesh, 2D Torus, 1D Torus), or in terms of supported routing algorithm (*XY*, *west-first*, unidirectional). Also, some differ in terms of number of VCs supported, support of error-control coding and the support of QoS. The flit width, buffer depth and routing algorithm in each input channel are parameterized. Scheduling and the arbitration algorithm can also differ for different implementations. In general, a centralized arbitration is used in each router to grant access to packets that arrive simultaneously.

2.8.3.3 *Xpipes: The University of Bologna and The Stanford University*

Xpipes [51, 34] is a composable NoC architecture that targets high performance and reliable multi-processors communication [34]. The architecture consists of a library of design time composable and tunable soft macros. The (routers, NIs and links) are highly parameterized and can be instantiated and synthesized to generate both application-specific heterogeneous or general-purpose homogeneous architectures. Thus Xpipes is not restricted by any topology and it has been validated in [35] for mesh, torus, hypercube, clos and butterfly topologies with different routing algorithms. Xpipes implements a cyclic redundancy check (CRC) at the link-level with a go-back-N retransmission strategy. This makes it resistant to interconnect errors. This design of Xpipes NoC is facilitated and then synthesized using a tool called *xpipesCompiler*. This tool can automatically instantiate an application-specific communication structure using Xpipes macros. It can also tune flit size, number of bits assigned for CRC check, maximum number of hops between any IP communication pairs, packet size, etc. Moreover, *xpipesCompiler* permits optimization of system level parameters such as removing redundant buffers from ports.

2.8.3.4 *MANGO: The Technical University of Denmark Approach*

MANGO is a message-passing asynchronous NoC developed at the Technical University of Denmark [39]. Its clockless implementation, bundled-data circuits and delay insensitive signal encoding, make it suitable for SoC design based on the coarse-grained GALS concept. MANGO provides both guaranteed service (GS) and best-effort service (BES) over OCP NI interfaces. The VCs in Mango are implemented as separate physical buffers. Allocating a sequence of VCs in a path between any given source-destination pairs creates the connection-oriented GS. Also, an asynchronous latency guarantee scheduling scheme is proposed to access router links.

2.8.3.5 *RAW: MIT University Approach*

The Laboratory for Computer Science at MIT designed and implemented the Raw microprocessor, the world's first many-core processor chip [155], that aimed to solve the wire delay problem at the deep submicron. The Raw processor consists of 16 identical programmable tiles and provides a software interface to resources on-chip. The tiles interconnect as a 2D mesh topology. Each tile consists of computational resources and communicational resources. The computational resources are: a microprocessor without interlocked pipeline stages (MIPS) style processor; floating-point unit; data cache and software-managed instruction cache. While the communication resources consist of four separate networks and thus each tile is connected with eight point-to-point 32-bit buses to neighbour tiles. Two of these networks are static

(routes predefined at design time) and two are dynamic networks (routes computed at run-time). The concept behind splitting the communication infrastructure into two types is to use the static networks to provide low latency traffic required for software circuits (mimic ASIC place-and-route facility) and parallel scalar codes (*e.g.*, *Specint* and *Specfp*) which are compile time predictable. However, the dynamic network is used to route data related to unpredictable operations; *e.g.*, interrupts, cache misses and unpredictable messages between tiles (long messages). Dynamic networks use dimension-ordered routing with wormhole flow control (32 flits the maximum supported packet size). Raw processor models have been widely studied by researchers in different contexts, see [8], like processor architecture design, software design, network routing, fault-tolerance, thermal impact, *etc.*

2.8.3.6 *SpiNNaker Chip: Manchester University Approach*

SpiNNaker is a project inspired by the human brain to build massively-parallel computer architecture [138]. The main objectives of the project are to provide a real-time platform to simulate a large scale neural network and propose a new energy-efficient computer architecture that is radically different to conventional supercomputing architecture. The building block of the SpiNNaker project is the SpiNNaker multicore SoC that contains 18 ARM968 processor nodes connected by packet-switched asynchronous communications infrastructure. The processors are synchronous. This make the chip follows the GALS methodology. The NoC consists of a single big bespoke router per chip that implements packets multicast protocol, which again is inspired by neurobiology. The router uses a routing-table to store the routing information and the packets are source routed. The chip is equipped with six bidirectional and self-timed inter-chip channels for off-chip communication. This allows the design to interconnect many SpiNNaker chips using different network topologies (off-chip network) to establish a million core machines. The design team are working to develop a 48-chip (SpiNNaker chip) board to act as building block to facilitate building bigger machines that can contain up to 1,036,800 ARM processor cores and will require approximately 100kW (106 machine [9]).

2.8.3.7 *Æthereal: Philips Approach*

The Æthereal project is a first industrial NoC framework, initially referred to as networks on silicon, developed at Philips/NXP Research Laboratories and aims to provide a platform with QoS to support SoCs design with real-time applications. The research in the Æthereal project evolved over 10 years at different levels of design, modelling, prototyping and implementation. A heterogeneous NoC architecture is adopted with both guaranteed and best-effort services. The GS is

implemented using pipelined time-division multiplexing (TDM) circuit switching and target real-time applications traffic which requires preserved data integrity, guaranteed throughput and bounded latency services. These serveices are necessary for predictable systems design. However, the BES implementation is realized using a conventional wormhole flow control [68]. The BES is used to increase NoC's resources utilization.

Goossens and Hansson in [67] declared mixing the GS and BES in the Æthereal was a classical mistake because the BES traffic worsens the *performance:cost* ratio of the design because router design with only GS is more area efficient and can run twice as fast as a router with mixed services (GS and BES). A commercial application, high-end digital TVs, retrofitted to the Æthereal NoC based SoC is presented in [153]. At the present time, the Æthereal NoC is used, with other components, to build a multi-processors platform called *CompSoC* [2, 72] which aims to reduce system complexity and enables faster performance verification.

2.8.3.8 *MIC: Intel Approach*

The Intel many integrated core (MIC) architecture is a multi-core computer construction that evolved from earlier Intel research projects on teraFLOPS multi-core chip and SCC. First, the Intel® Tera-scale Computing Research Program [88] unveiled in 2007 a prototype of multicore processor chip (also called teraFLOPS processor) based on tile-based design. The tile-based design approach enables designers to use the tile as the building block to rapidly compose a chip by repeatedly instantiating tiles across the chip. The teraFLOPS processor consists of 80-tiles each of which comprises compute element (core), memory element (instruction memory (IMEM) and data memory (DMEM)) and communication element (router and links), and is designed to run at 4GHz. The core contains two programmable floating point engines and the communication element contains 5-port (links) and a message-passing router. The communication elements are interconnected in a 2D mesh network fashion (network-on-chip) with a source routing algorithm. The 80-core prototype chip is fabricated in a 65-nm process technology and able to deliver teraflops performance (trillion floating point operations per second). The chip is also equipped with fine-grain power management so that each compute and communication element can be activated or put to sleep based on the running application.

Subsequently, the Intel® Tera-scale Computing Research Program introduced in 2009 a prototype chip fabricated in 45-nm process technology and tested to implement IA-32 architecture on a many-multicore processor, called SCC. It is a CPU chip that contains 48 cores, each capable runoff running a separate operating system (OS) and software stack. Similar to 80-tile chip, the SCC is also based on tile-based design

except each tile contains two cores (each is Pentium class x86-32 cores), two L2 caches and router. A packet-based on chip network is used with a message-passing programming model to communicate among the compute nodes (cores). This on-chip network architecture supports "scale-out" protocol that is able to scale to 1,000s of processors in a cloud datacenter [143]. An advanced dynamic voltage and frequency scaling (DVFS) fine-grain power management is used in SCC compared to the teraFLOPS processor chip. In addition to clock gating at tile level, individual tiles (two cores) can run at different frequencies and each group of four tiles can run at different voltages. This enables the SCC to run all 48 cores concurrently over a range of operating power, 25W to 125W. Moreover in SCC the clock gating can be used at individual ports of the NoC's routers. It has been reported in [141, 78] that the NoC performance in the new prototype has been improved 2.8X per watt compared to the teraFLOPS chip. The control of the voltage and frequency are given to special software applications. The higher computation and power management capabilities of the SCC makes it a potential candidate for a variety of applications such as web servers, physics modelling, and financial analytics [143].

Finally, Intel announced that a commercial release of a multicore microprocessor chip (called *Knights Corner* and containing more than 50 processing cores), based on Intel MIC architecture to be built in 22-nm process technology, will take place in late 2012 to 2013. Intel also announced and released in 2012 a development kit, called *Knights Ferry*, which aimed to aid software and hardware developers to prepare for Knights Corner.

2.8.3.9 *Simulation Platforms*

In terms of simulation and performance evaluation, there are several network simulators running in different environments. Some of these are general computer network simulators like the open-source Network Simulator (ns-2 [12]) and the proprietary OPNET [13]. Others can either be used for both on-chip and off-chip networks like Booksim [1] or dedicated for on-chip networks simulation like Noxim [62], NetMaker [128], Nirgam [89]. Table 2.4 summarizes the main features and differences among the above listed on-chip networks simulators along the lines of the simulator used in Chapters 3&4 and Chapters 5&6.

NoC work	Topology	Flow control & buffering	Different QoS	Routing & routing deadlock handling	Error control	Implementation level and tested applications
Nostrum [6]	2D mesh	- Packet switching - Minimal input and output buffers	Best effort and guaranteed latency	- Deflective routing - Inherently deadlock free	Introduced in [160] at link & network levels	- PANACEA [71] is a 4 × 4 prototype chip fabricated at 250µm technology (100MHz). - Multi-core processor and multi-media are potential applications
Xpipes [51]	Unrestricted – validated for 2D mesh, torus, hypercube, Clos	- Wormhole with VCs - Output buffers	Not supported	Dimension ordered, split-traffic across minimum/all paths - Deadlock avoidance	CRC check and link-level retransmission	- register transfer language (RTL) synthesis - Tested for moving picture experts group (MPEG)-4 design
SoCIN [166]	2D mesh, torus and folded torus	- Wormhole - Input buffers (customizable)	Not supported	- Deterministic XY - Deadlock avoidance	Extendable to include party bit check	- Synthesized targeting FPGA board (73MHz maximum frequency) - No possible application listed
SPIN [70]	Fat-tree	- Wormhole - Input buffers and 2 shared output buffers	Not supported	- Deterministic and adaptive (deflective) - Deadlock avoidance	Not supported	- ASIC layout using 0.13µm CMOS technology - No possible application listed
Hermes [21]	2D mesh or 2D torus or 1D torus	- Wormhole with or without VCs - Input buffers	BES and GS in some versions	- Deterministic *XY* or *west-first* - Deadlock avoidance	link, source CRC and Hamming code	- Synthesized targeting FPGA board (56.7MHz maximum frequency) - No possible application listed
Chapters 3&4	Unrestricted – validated and tested for 2D and 3D meshes	- Unrestricted – tested for wormhole - Input buffers	Not supported	- True fully adaptive routing - Deadlock recovery	Not supported	- Fabricated using a die-stacked 3-layer using TSV and 150nm CMOS technology. - Tested for multi-media system benchmark. - CMPs and SoCs are potential applications.

... to be continued

Table 2.3: Different approaches to implement the on-chip networks

NoC work	Topology	Flow control & buffering	Different QoS	Routing & routing deadlock handling	Error control	Implementation level and tested applications
Chapters 5&6	Unrestricted – validated and tested for 2D and 3D meshes	- Unrestricted – tested for wormhole - Input buffers	Not supported	- Coolest path first and least throttled path first adaptive routing - Deadlocks avoided based on the turn-model rules	Not supported	- Simulation and synthesis using Synopsys Design Compiler and mapped onto the FARADAY UMC 65nm - 3D based TSV integration of CMPs and SoCs are potential applications.
MANGO [39]	Unrestricted – validated only for 2×2 mesh	- Mixed of message and flit flow control - Output buffers for each VC	GS and BES	- *XY* source routing for BES traffic and multi-hop virtual circuit switching for GS - Deadlock avoidance	Not supported	- Simulations of a 0.13µm standard cell with 333MHz maximum frequency - GALS systems is potential applications
RAW [8]	2D mesh (4 separate networks)	- Wormhole for dynamic networks - Input buffers	Two static & two dynamic networks	- Dimension-ordered routing - Deadlock avoidance	Not supported	- 16-tile ASIC prototype in 0.15µm - Microprocessor, software circuits like gigabit routers and video & audio processing
Æthereal [68]	Unrestricted - except every path must be reversible	- GS use TDM circuit switching and BES use wormhole - Input buffers	GS and BES	- Contention-free routing used for GS with slot-table and source routing for BES - Deadlock avoidance	Not supported	- ASIC synthesisation and layout using 65nm technology and FPGA - Tested for a commercial application (high-end digital TVs, automotive)
TeraFLOPS [88]	2D mesh	- wormhole with VCs (2-VC) - Input buffers (16 for each VC)	Not supported	- Source routing - Deadlock avoidance (two logical network using VCs)	Not supported	- Fabricated chip at 65nm process technology with 5.67Ghz - Tested for several benchmarks
SCC [143]	2D mesh	- Virtual cut-through - Input buffers (24-entry buffer space assigned dynamically VCs)	Not supported	- Dimension-ordered XY routing - Deadlock avoidance	Not supported	- Fabricated chip at 45nm process technology with 2.35Ghz maximum frequency - Web servers, physics modelling, and financial analytics are potential applications.

Table 2.3 (continue): Different approaches to implement the on-chip networks

Simulator name	Maintained by:	Topologies and flow control	Routing algorithms	Traffic for evaluation	General comments
Booksim [1]	Stanford University	- 2D mesh, torus, flattened butterfly, trees, etc. - Wormhole with VCs	Diverse routing algorithms, e.g., deterministic, adaptive, minmmal, nonminimal, etc.	Uniform, bit complement, bit reverse, shuffle, transpose, tornado, neighbor and random permutation	- C++ cycle-accurate interconnection network simulator - originally developed for Dally et al. [52] book and then updated to simulate NoCs. - very comprehensive in terms of supported topologies and routing algorithms - well maintained, last update in the website was in April 2012.
Noxim [62]	University of Catania	- 2D mesh only - Wormhole	XY, west-first, north-last, negative-first, odd-even, dyad and table based routing	uniform, transpose, bit-reversal, butterfly, shuffle and table based traffic	- SystemC cycle accurate NoC simulator - command line interface - widely used in NoC community - easy to expand - limited in terms of topologies and architecture
Netmaker [128]	Cambridge University	- No topology specified - Wormhole with VCs	Deterministic *XY*	Uniform random	- SystemVerilog cycle accurate simulation - FPGA emulation - ASIC synthesization - provide packet prioritisation - last update in April 2009
Chapters 3&4 (modified Noxim)	Newcastle University	- 2D and 3D meshes - Wormhole	Noxim routing algorithms plus true fully adaptive routing	Noxim traffic plus multimedia system traffic	In addition to Noxim features: - Deadlock detection and recovery methods are added - Extended to 3D dimension

... to be continued

Table 2.4: Main featurtes and differences among the avilable on-chip networks simulators

Simulator name	Maintained by:	Topologies and flow control	Routing algorithms	Traffic for evaluation	General comments
Chapters 5&6 (modified Noxim)	Newcastle University	- 2D and 3D meshes - Wormhole	Noxim routing algorithms plus *XYZ*, 3D *negative-first*, coolest path first and least throttled path first	Noxim traffic plus 3D uniform random, 3D transpose	- Integrated with NoC power simulator (Orion [94]) - Integrated with NoC thermal simulator (HotSpot [151])
Nirgam [89]	University of Southampton and Malaviya National Institute of Technology-India	- 2D mesh and 2D torus - Wormhole with VCs	- 2D mesh: deterministic *XY* and partially adaptive *odd-even* - 2D torus: source routing	Customizable constant bit rate and bursty traffic	- SystemC cycle accurate simulation - Customizable and extendable - Well documented
Worm_sim [79]	Crnegie Mellon University	- 2D mesh and 2D torus - Wormhole with VCs	*XY*, *odd-even* and *DyAD*	Uniform, transpose1, transpose2 and random with hotspot	- C++ cycle accurate simulation - customizable and extendable - support Orion power model - well documented - last update Feb. 2010

Table 2.4 (continue): Main featurtes and differences among the avilable on-chip networks simulators

3 RUN-TIME DEADLOCK DETECTION

3.1 INTRODUCTION

Networks-on-Chip (NoCs) emerged as an on-chip communication architectural paradigm that applies networking theory and methods to VLSI systems incorporated on a single silicon chip [33, 55]. Such an architecture consists of a network constructed from multiple point-to-point data links, namely channels, interconnected by routers. The routers are connected to a set of distributed IP cores where each IP core can be an implementation of processor cores, memory modules, DSP blocks and embedded reconfiguration modules, *etc*. Communication among these distributed IPs usually utilizes a packet-switching method where messages are divided into suitably-sized blocks, called packets and flits. The packets can be relayed from any of the source-destination communication pairs, over several data links, by making routing decisions at routers. NoCs bring a remarkable improvement in performance, flexibility, scalability and power efficiency over conventional bus interconnections. However, in NoCs communication deadlocks may appear at the router network and cause an impasse in the communications, hence leading to performance degradation or system failure.

A deadlock is a situation in NoCs wherein two or more packets are waiting for one another to release channels and are unable to make progress. The network is deadlocked if a chain of waiting packets forms a cycle. As a simple deadlock example, consider the dependency cycle of four packets shown in Figure 3.1. Each of the four packets in the figure holds some channels, but it cannot proceed further until it acquires another channel currently held by one of the other of the packets. However, no packet can release the channel needed by another until it acquires its own requested channel. The packets are deadlocked and will remain in this state, halting all the occupied channels, until some intervention brings them back to life by breaking the dependency cycle. Deadlocks can paralyze network communications by halting the resources occupied by the deadlocked packets which, in turn, could increase the likelihood of other packets blocking [162]. Hence the removal of deadlocks is crucial. There are two strategies to deal with deadlocks: *deadlock avoidance* [52] and *deadlock recovery* [61].

In deadlock avoidance, resources are granted to packets in a way that the overall network is deadlock free. This can be based on a turn model which prohibits the routing algorithm from making certain turns in the network [65, 47] or based on the strict ordering of virtual channels [61]. In general, avoidance techniques require restricted rout-

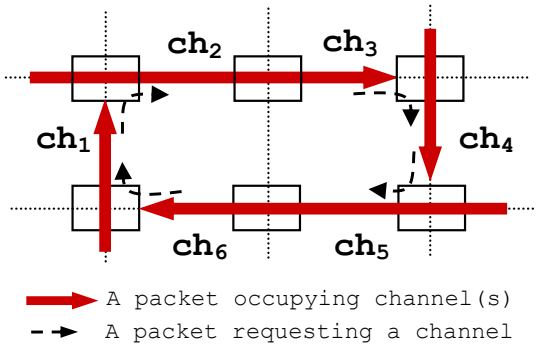


Figure 3.1: Four packets in a 2D network are waiting for one another and forming a deadlock configuration.

ing functions or additional resources to prevent deadlocks [52, 146]. Due to its simplicity the turn model technique is popular in NoCs, even though it limits the routing alternatives and diminishes fault tolerance capabilities [102]. Moreover, it is not applicable to arbitrary network topologies.

Alternatively, deadlock recovery implies that resources are granted to packets without any routing restrictions, potentially outperforming deadlock avoidance [22, 140, 96, 112]. Deadlocks may occur and efficient detection and recovery mechanisms are required to intervene. However, detecting deadlock in a network is challenging because of the distributed nature of deadlocks. Heuristic approaches, such as timeout mechanisms, are often employed to monitor the activities at each channel for deadlock speculations. These techniques may produce a substantial number of false detections, especially with the network close to saturation where blocked packets could be flagged as deadlock. Several techniques have been proposed for reducing the number of false detections in general computer networks [124, 112, 105], but they are all still based on the timeout idea and finding the best threshold values for different network settings is not an easy matter.

To recover from deadlock, there are two kinds of deadlock recovery schemes: *regressive* and *progressive*. A regressive recovery is based on the abort-and-retry mechanism [96], which kills the suspected packets. A progressive recovery uses additional hardware in each router node (central buffers) to bypass the suspected packets to their destination sequentially [22] or concurrently [23].

Unlike general computer networks, where internode information can only be exchanged through packets, on-chip networks can take advantage of additional dedicated wires to transmit control data between routers. This chapter exploits this NoCs-specific capability and proposes a new deadlock detection method which guarantees true deadlock detection. A run-time transitive closure (TC) computation scheme is employed to discover the existence of deadlock-equivalence sets which imply loops of requests. Also, the proposed detection

scheme can be realized using a distributed architecture, which closely couples with the NoC infrastructure, to speed up the necessary computation and to avoid introducing traffic overhead in the communication network. The major contributions of this chapter are:

- Introducing a deadlock-equivalence set criterion for detecting loops of packet requests.
- Presenting a new deadlock detection scheme to discover the existence of deadlock-equivalence sets based on transitive closure computation.
- Introducing a distributed architecture, TC-network, to implement the proposed detection technique. This architecture allows the process of deadlock detection to raise exception, for the purpose of their elimination in the NoC as early as possible, as opposed to performing exhaustive TC calculations for the entire network.
- Evaluating the proposed deadlock detection scheme through experimental studies and comparisons with the state-of-the-art timeout detection schemes using various traffic scenarios.
- Evaluating the hardware area and power overhead of the TC-network.
- Introducing a new deadlock recovery technique based on the end-to-end communication protocol.
- Evaluating the proposed deadlock detection method, TC-network, with the proposed deadlock recovery technique in a full deadlock recovery system.

Preliminary results of this study and a sketch of the proposed architecture were presented in [14]. In [19], a complete theoretical framework of the TC computational approach is presented. Hardware architecture for the framework realization is detailed and experimental results on real-life applications are included. The method and the results of integrating the TC-network into a complete deadlock recovery system are also included.

This chapter is organized as follows: Section 3.2 summarizes deadlock-recovery related work from the field of general computer networks. Section 3.3 illustrates the deadlock-equivalence set principle and general methodology for deadlock detection. Section 3.4 presents a TC-network for the realization of deadlock detection in NoCs along with its hardware implementation. Section 3.5 includes experimental results and discussion. Section 3.6 introduces and evaluates a deadlock recovery technique and Section 3.7 summarizes and concludes this chapter.

3.2 RELATED WORK

The study of deadlock recovery in the context of NoCs is rare and, as a consequence, the following information has been mainly acquired from the field of general computer networks. In [162, 137], the authors conclude that deadlocks can be highly improbable in interconnection networks when sufficient routing freedom is provided and fully exploited by the routing algorithm. Thus it is not favourable to limit the adaptivity of the routing algorithm to avoid an infrequent event, *e.g.*, using the turn model [65, 47], nor to complicate the routers' design by devoting virtual channels specifically to prevent deadlocks [61, 52]. Since then, deadlock recovery has gained acceptance for its potential for outperforming deadlock avoidance provided efficient detection and recovery mechanisms exist [112, 124, 140, 105]. Deadlock *detection* and *recovery* are the two important stages of any deadlock recovery scheme [61].

In the detection stage, the network must discover at run-time any deadlock configuration. However, detecting deadlocks at run-time is challenging because of their highly distributed characteristics. Thus deadlock detection is usually implemented in a distributed way using a timeout mechanism [102, 22, 23, 96, 112, 124, 140, 105]. In its simple form, a packet occupying a channel is suggested to be in a deadlock if the channel has been inactive for a given threshold time value [22] and thus the recovery stage is started. The minimum hardware components for each physical channel in the router in order to implement such a scheme are: a counter, comparator, latch and a register to store the threshold value, in case a programmable threshold is required. Correct detections of deadlock in this mechanism are very sensitive to network load and message length transmitted over network channels. A long threshold time leads to more accurate deadlock detections, but takes a longer time to discover deadlock and thus increases packets blocking and dramatically degrades network performance [140]. On the other hand, a short threshold value detects deadlock faster, but with a higher probability of false detections (false positives) and thus could saturate deadlock recovery channels, in the case of a dedicated central channel used for recovery in each router [22], which again can degrade the network performance [140].

Several techniques have been proposed for interconnection networks to reduce the number of false positive deadlocks. In [124] a packet is suggested as being deadlocked if all requested channels by a blocked packet are inactive for a given timeout. To further reduce the number of false deadlock detections, the mechanism presented in [140] is intended to identify only one packet in a sequence of blocked packets as being deadlocked. The work of [140] is less vulnerable to false deadlock detections at the expense of extra hardware (two comparators, two latches and two threshold values). In [105] the author proposed

a technique that employs special control packets (called probes) to cross along inactive channels for more accurate deadlock detection. In general, all these works are based on the timeout mechanism and pose a difficulty in tuning the threshold value, i.e., to select a unique threshold value for different traffic and network loads. Therefore, a method which accurately detects deadlock without false alarms is required. A method which quickly detects deadlock independent of the network's traffic and load is desirable.

In the recovery stage, there are two kinds of deadlock recovery schemes: *regressive* and *progressive*. A regressive recovery is based on the abort-and-retry mechanism [96] which removes the suspected packets from the network. While a progressive recovery resolves deadlocked configuration without removing suspected packets from the network. For instance, the DISHA progressive recovery scheme utilizes additional hardware in each router (central buffer) to bypass the suspected packets to their destination sequentially [22] or concurrently [23]. The bandwidth of these central channels is a fraction of the original network bandwidth. Therefore if the detection technique is detecting a lot of false deadlocks this will saturate the recovery bandwidth and as a result will degrade the network's performance [140].

3.3 METHODOLOGY FOR DEADLOCK DETECTION

The following sections present the proposed deadlock-equivalence set principle and the general methodology for the proposed deadlock detection method.

3.3.1 *Background and Assumptions*

The network of concern in this work is a collection of router nodes connected by channels. Each router is connected to a single IP core that can inject/consume packets via the router. A fully adaptive routing algorithm with minimal paths is used. 'Minimal' in this context means that the routing algorithm always chooses the shortest path between sender and receiver. A wormhole flow control technique [54] is employed, a method which has been widely used in NoCs [146]. The packets in wormhole flow control are divided into smaller flits. A header flit is routed first and the rest will follow it in a pipeline manner and this will allow a packet to occupy several channels simultaneously. Thus, wormhole reduces the number of buffers required in each router which is a desirable feature for on-chip networks. However, wormhole makes networks more prone to blocking and deadlock [162, 129].

In this chapter, NoCs are studied without using any deadlock avoidance techniques, but adopting deadlock recovery with an accurate deadlock detection method (proposed in the next section). In line with

existing work on deadlock detection/recovery [22, 140, 124, 105, 112], this work assumes that a channel buffer cannot contain flits belonging to different packets. Moreover, a packet arriving at its destination is eventually consumed. In other words, the network is deadlock free at the protocol interaction between the NoC and IP cores. It is important to notice that this thesis focuses on studying the kind of deadlock that is caused by routing cycles in the network and not the one arises by message dependencies caused by protocol interaction between the NIs and the network. Moreover, the used terminology for an *agent* that own and request *resources* (channels) in this work is a *flit*. However, it is equally true that an agent is a message or a packet.

3.3.2 *Equivalence Set Criterion for Deadlock*

This section introduces the proposed method for deadlock detection in NoCs. It first introduces some important definitions and then defines a deadlock-equivalence set (DES) criterion for detecting loops of packet requests. It uses the TC computation to determine whether there is a set of channels in the NoCs forming a DES. The following definitions lead to that of a deadlock in NoCs:

Definition 1 *A SoC and/or CMP consists of communication infrastructure called a NoC and computation/storage cores called IPs.*

Definition 2 *A Network-on-chip $\mathcal{N}(\mathcal{V}, \mathcal{C})$ is a strongly connected directed graph, where $\mathcal{V}$ is a set of elements called Vertices that represent a set of router nodes and $\mathcal{C} = \mathcal{V}x\mathcal{V}$ is a set of ordered pairs, $(c_i, c_j) \neq (c_j, c_i)$, called edges that represent a set of Channels that connect routers. A single channel is only allowed to connect a given pair of routers in one direction.*

Definition 3 *An $\mathcal{I}$ is a set of processing/storage elements that represent the IPs integrated on-chip. Each IP has one injection channel and one delivery channel directly connected to a router $(v \in \mathcal{V})$.*

Definition 4 *The routing function R in $\mathcal{N}$ is a function that returns the output channel, $c_{out} \in \mathcal{C}$, for each current node, $v_c \in \mathcal{V}$, and destination node, $v_d \in \mathcal{V}$, so that $c_{out} \neq c_{in}$. In other words, deflection routing is not allowed and no channel has the same network node as both its source and destination.*

The information routed and propagated in the NoC are packets/flits. These represent the agents that own and/or request the network resources (channels). The resource ownership and request in a NoC at any particular time can be expressed as a channel wait-for graph (CWG). [52].

Definition 5 *A CWG is a directed graph $\mathcal{G} = (\mathcal{C}, \mathcal{E})$ where $\mathcal{C}$ is a set of vertices in the network $\mathcal{N}$ and represents the set of channels. $\mathcal{E}$ is a set of*

ordered pairs called edges which represents the set of channel occupation and requisition status. At any particular state of the NoC, there exists an edge $(u, v) \in \mathcal{E}$ *either if* (1) *there is a head flit in channel* u *requesting channel* v*, or if* (2) *there is a flit in channel* u *and another flit in channel* v *and both of them belong to the same packet. Case* (1) *refers to the request status and is drawn as dashed arcs in the CWG, while case* (2) *refers to the ownership status and is drawn as solid arcs.*

For instance, case (1) can be seen in Figure 3.1, where a head flit occupying ch_1 and requesting ch_2. Also case (2) is shown in the figure where a data flit occupying ch_2 and its head flit in ch_3. A matrix representation of the CWG can be extracted based on the Adjacency Boolean matrix.

Definition 6 *The Adjacency Boolean matrix* A *of a directed graph* $\mathcal{G} = (\mathcal{C}, \mathcal{E})$ *is an* $n \times n$ *matrix, where* n *is the cardinality of* $\mathcal{C}$ *and* $a_{i,j} \in A, \forall_{i,j} \in n$*. A is constructed as follows: (1)* $a_{ij} = 1(\texttt{True})$ *iff* $(i, j) \in \mathcal{E}$ *(edge exists between vertex* i *and vertex* j*), (2)* $a_{ij} = 0(\texttt{False})$ *otherwise (including when* $i = j$*, see Definition 4).*

Given a directed graph $\mathcal{G}$, it is possible to answer reachability questions using the concept of transitive-closure. For example, can one get from node (vertex) u to node v in one or more edges (hops)?

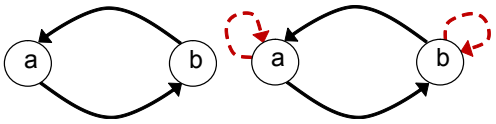
Definition 7 *The Transitive Closure (TC) of* $\mathcal{G}$ *is a directed graph,* $\mathcal{G}^+ = (\mathcal{V}, \mathcal{E}^+)$*, which contains an edge* (u, v) *if, and only if, there is a directed path from* u *to* v *in one or more hops. It means that if* $\mathcal{G}$ *contains the edges* (u, w) *and* (w, v)*, then* v *can be reached from* u *(transitivity property).*

The directed graph $\mathcal{G}^+$, the transitive-closure of $\mathcal{G}$, is the result of adding to $\mathcal{G}$ only the edges that cause $\mathcal{G}$ to satisfy the transitivity property and no other edges (i.e., not adding edges that do not represent paths in the original graph). Thus, the TC of a directed graph is obtained by adding the fewest possible edges to the graph such that it is transitive. It can be computed from A using Floyd-Warshall algorithm [49] (see Algorithm 3.1, *lines 12-18*) and will be denoted as T.

At this point, it is necessary to introduce the deadlock-equivalence set criterion for detecting a loop of packet requests.

Definition 8 *Given a set of channels* $\mathcal{C} = \{c_1, c_2, \ldots, c_l\}$*,* $l =\mid \mathcal{C} \mid$*), from the network* $\mathcal{N}$ *and consider a subset* $\mathcal{M} = \{c_1, c_2, \ldots, c_m\}$ *of channels for some* $m \leqslant n$*.* $\mathcal{M}$ *is a deadlock-equivalence set (DES) iff in all its channels there are flits waiting for one another in a cyclic manner to progress to their respective destinations,* i.e., c_1 *occupied by a flit and it requests* c_2*,* c_2 *occupied by a flit and it requests* $c_3, \ldots, c_{m-1}$ *occupied by a flit and it requests* c_m *and* c_m *occupied by a flit and it requests* c_1*.*

In NoCs characteristics, the CWG, at a particular time, has for any node at most one outgoing arc. Hence, a node can appear in a DES



(a) Original graph (b) TC graph

Figure 3.2: Graphical representation of Example 1

only once and cannot appear in multiple DES's at the same time. Thus, members of a set of simultaneous DES's, $\mathcal{S} = \{S_i\}$, are pair wise disjoint; that is, S_i, $S_j \in S$ and $i \neq j$ implying $S_i \cap S_j = \phi$. The TC computation can be used to determine whether there is a set of channels in the network forming a DES. To demonstrate the idea so far, consider the following example:

Example 1 *Given a channel* i *and a channel* j*, where* $i, j \in \mathcal{C}$*, and assumes there is a flit occupy each of these channels. Also, each flit request the other channel,* i.e.*, the flit occupy channel* i *requesting channel* j *and the the one occupy channel* j *requesting* i*. Then the corresponding adjacency Boolean matrix is:*

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

The transitive closure of A *is:*

$$T = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix},$$

i.e., $T_{1,2} = T_{2,1} = 1$ *and* $T_{1,1} = T_{2,2} = 1$*. The last two imply a loop of channel requests as channel* i *requesting itself and channel* j *requesting itself. These will be shown as self-reflexive paths in the TC graph (see Figure 3.2).*

This can be extended to a subset $\mathcal{M} = \{c_1, c_2, \ldots, c_m\}$ of channels for some $m \leqslant n$, such that all pairs of elements in $\mathcal{M}$ meet the self-requesting condition:

$$\text{DES}(i,j) = \begin{cases} 1, & \text{if } T_{i,j} = T_{j,i} = T_{i,i} = T_{j,j} = 1, \\ 0, & \text{otherwise.} \end{cases} \quad \forall i, j \in \mathcal{M} \qquad (3.1)$$

To recover from a deadlock situation, the dependency cycle formed by the DES should be resolved. A deadlock can be resolved if one or more of the packets forming the deadlock is removed from the network [140, 96, 112]. The next section presents the DES computational complexity for NoCs.

3.3.3 *Equivalence Set Computational Complexity*

The DES provides a simple criterion for deadlock detection. The technique is to generate the CWG from the network and then derive the TC of the CWG and identify the channels that satisfy the criterion. Figure 3.3 illustrates this idea. Given a network at any particular state (Figure 3.3a) the CWG can readily be drawn (Figure 3.3b). The derived TC graph (Figure 3.3c) clearly shows four vertices (channels) with self-reflexive paths and all pairs of these satisfy the condition of the DES (Eq.3.1).

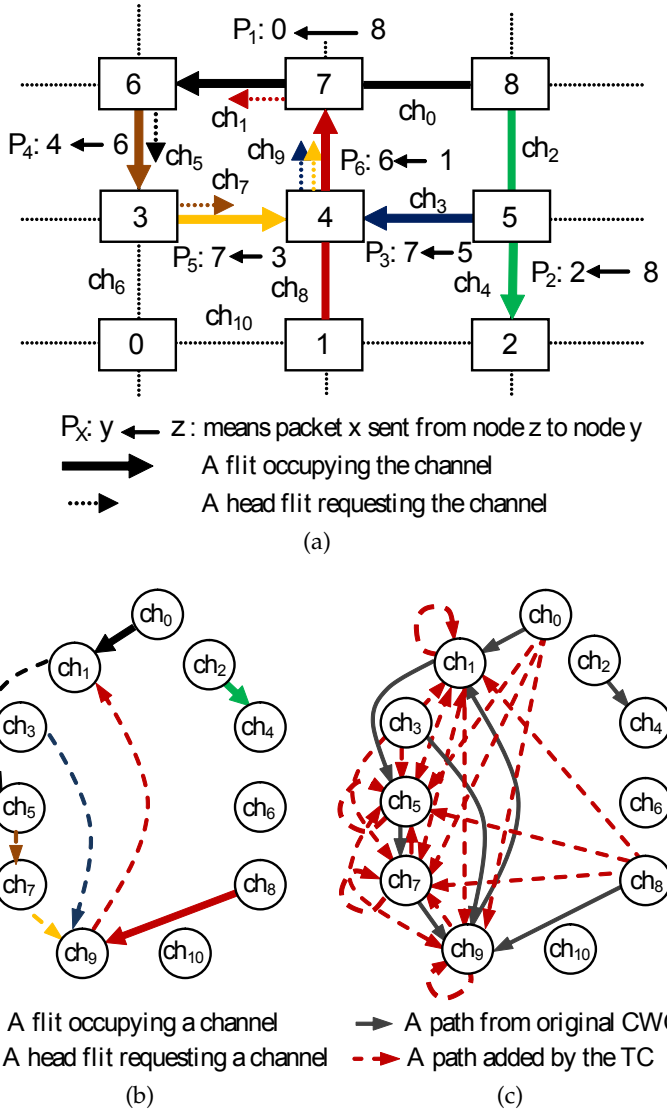


Figure 3.3: Detecting deadlock from a 2D-NoC: (a) A scenario of deadlock formation; (b) the CWG of the network; (c) the TC of the CWG, the set of channels $\{ch_1, ch_5, ch_7, ch_9\}$ satisfy the DES definition (Eq.3.1).

The computational procedure of transitive closure is outlined in Algorithm 3.1. It has three nested loops containing an $O(n)$ core. In lines 2-10, it converts the directed graph, CWG, to a Boolean Adjacency matrix to reflect the existing paths in the graph. In lines 12-18, the transitive closure is computed. The code state T^k should show a path from vertex i to vertex j if (1) T^{k-1} already shows a path from i to j, passing through one of the vertices in $1 \dots k-1$, or (2) T^{k-1} shows a path from i to k and a path from k to j; hence there will be a path from i to j through k. Lines 19-23 check if there exist any self-reflexive path in the computed TC, i.e., there is at least single channel i where the transitive-closure matrix $T_{i,i} = True$.

Algorithm 3.1 Pseudo code to check the existence of a DES using TC computation

Definition: CWG is the channel wait-for graph for a NoC $\mathcal{I}$.
Input: $\mathcal{C}$ is a set that contains all the vertices in CWG.
$\mathcal{E}$ is a set that contains all the edges in CWG.
Output: Return *True* if there exists a deadlock-equivalent set in the CWG (self-reflexive path).

1: $n \Leftarrow |\mathcal{C}|$
2: **for** $i = 1$ to n **do**
3: **for** $j = 1$ to n **do**
4: **if** $(i \neq j \wedge (i,j) \in \mathcal{E})$ **then**
5: $A_{ij} \leftarrow True$
6: **else**
7: $A_{ij} \leftarrow False$
8: **end if**
9: **end for**
10: **end for**
11: $T^0 \Leftarrow A$
12: **for** $k = 1$ to n **do**
13: **for** $i = 1$ to n **do**
14: **for** $j = 1$ to n **do**
15: $T^k_{ij} \leftarrow T^{k-1}_{ij} \vee \left(T^{k-1}_{ik} \wedge T^{k-1}_{kj}\right)$
16: **end for**
17: **end for**
18: **end for**
19: **if** ($\exists i : T^n_{i,i}$) **then**
20: **return** *True*
21: **else**
22: **return** *False*
23: **end if**

The computation of the TC is expensive. In terms of software the algorithm requires a computational complexity of $O(n^3)$, where n is the number of vertices in the CWG (i.e., number of network channels).

The algorithm requires only a complexity of $O(n^2)$ as storage [49]. In hardware, however, there are many proposed previous works to speed up the TC computation time to $O(n)$ using a systolic array of $O(n^2)$ processing elements [100, 142]. In this work, however, the NoC distributed architecture is used to simplify the computation complexity. A distributed architecture to realize the TC computation is presented in the next section. Also, the reflexive property is exploited to further simplify DES detections in NoCs to realize the proposed deadlock detection.

3.4 TRANSITIVE CLOSURE (TC) NETWORK ARCHITECTURE

3.4.1 *TC Computation with TC-networks*

Dynamic programming (DP) can yield the solution for the transitive closure [49] with an opportunity for solving the computation using a parallel architecture. Mapping TC computation to a parallel computational platform can be achieved with the introduction of a TC-network. The network has a parallel architecture and can be used to compute the TC solution through the simultaneous propagation of successive inferences. Lam and Tong [101] introduced DP-networks to solve a set of graph optimization problems with an asynchronous and continuous-time computational framework. This inference network is inherently stable in all cases and has been shown to be robust with an arbitrarily fast convergence rate [101]. A parallel computational network for solving the dynamic routing problem for NoCs is also proposed in [118]. This work introduces a TC-network to discover the existence of DES at run-time. Similar to the DP-network [101, 118], the proposed TC-network can be realized using a distributed architecture which closely couples with the NoC infrastructure.

The TC-network is constructed by the interconnection of autonomous computational units. The structure of a unit and links in a general inference network is shown in Figure 3.4. Each unit represents a binary relation (i, j) between two objects and there are h sites, neighbouring units, to perform the inference action as defined in the site function. The value of the corresponding relation between i and j is then determined by resolving the conflict among all of the site outputs. Basically, if $S_k(i, j)$ represents the site output at the k-th site and $g(i, j)$ stands for the unit output of unit (i, j), then the TC computation can be stated in terms of network structure:

$$S_k(i, j) = g(i, k) \wedge g(k, j) \tag{3.2}$$

$$g(i, j) = \vee_{\forall k} S_k(i, j) \tag{3.3}$$

where $\wedge$ (AND function) is the inference for the site function and the conflict-resolution operator for the unit function. The operator $\vee$ (OR function) denotes the unit which resolves the binary relation (i, j). The computational units will be interconnected in the same way as the NoC structure. Each unit represents a router node and a link represents a communication channel. A distributed network can readily be implemented using such realization.

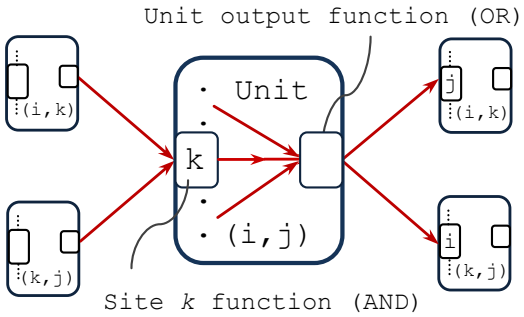


Figure 3.4: Unit interconnection in a general TC-network where $1 \leqslant i, j, k \leqslant n; k \neq i, j$ [101]. The output of Unit (i,j) will be the input of other units according to the problem network structure. At each unit, there are h sites, which corresponds to the total number of neighbouring nodes of i, to carry out the inference operations as defined in the site function.

The delay of TC-network convergence to deadlock-detection depends on the size of the DES and the network topology, which both determine the delay of information propagation within the network and the delay of each computational unit. As can be seen, each unit involves $O(|A|)$ AND operations and one OR operation where $|A|$ is the number of adjacent edges. Hence, the solution time is $O(k|A|)$ where k is the number of iterations evaluated by each unit. In a software computation, k equals to the number of nodes in the network which guarantees that all nodes have been updated. However, in the hardware implementation with parallel execution, k will be determined by the network structure and $|A|$ AND operations can be executed in parallel. Each computational unit can simultaneously compute the new expected output for all neighbour nodes. The delay of TC-network convergence to deadlock detection is directly proportional to the size of the DES, which determines the delay of information propagation within the TC-network and the delay of TC computational units. The worst case delay for the TC-network to converge to the deadlock detection is investigated for several popular network topologies in the NoCs literature. Some topologies are excluded because they are inherently deadlock free in conjunction with the routing function stated in Definition 4): e.g., tree, bus, star, butterfly, etc. Table 3.1 shows the largest DES for several network topologies as a function of the network size (k). The induction is used to find these equations. Consider the

Network topology	# of nodes	# of channels	Smallest DES	Largest DES
k-ary 1-cube	k	$2k$	k	k
k-ary 2-cube	k^2	$4k^2$	4	$3k^2 - 8$
k-ary 2-mesh	k^2	$4k^2 - 4k$	4	$3k^2 - 3k - 2$

Table 3.1: DES analysis of a TC-network for different network topologies

k-ary 1-cube (ring) topology: the maximum number of channels in such a network is $2k$, while the largest and the only possible DES is k as this will be the longest dependency cycle. Likewise for the k-ary 2-mesh network topology (2D mesh) of k^2 nodes with k rows and k columns, the smallest DES in this network will span over four nodes while the largest DES will be $3k^2 - 3k - 2$ and thus, the detection time will depend on the network topology and on how deadlocks are distributed over the network nodes.

3.4.2 *Coupling TC-network to NoC*

An on-chip communication network itself defines the graph vertices of its TC-network. This provides an opportunity for TC computation embedding a TC-unit at each node. Unlike general computer networks, where internode information can only be exchanged through packets, on-chip networks can take advantage of additional dedicated wires to transmit control data between routers. The TC-network shown in Figure 3.5 consists of distributed computational units and links between them. The topology of the network resembles the defined graph topology, which is the communication structure of a NoC. Although a 2D mesh network is used here as an example, the proposed methodology can be generalized to any network topology constrained only by Definition 2. At each node, there is computational unit which implements Eqs. 3.2 and 3.3 for each channel. The output of the unit will be propagated to neighbour units via interconnects. The TC-network tightly couples to the NoC and each computational unit locally exchanges the run-time information, such as local channels' occupation and request status, with the router.

In deadlock detection/recovery strategy the ultimate goal of detecting a deadlock is to invoke a recovery mechanism. Recovering one packet from each DES will be sufficient to break the deadlock dependency cycle. However, performing exhaustive TC calculation for the entire network will discover concurrently all the channels in the network that are members of the same DES and will add recovery overhead by requesting to release all channels. Returning to Figure 3.3c, the TC graph exhibits four self-reflexive paths around ch_1, ch_5, ch_7 and ch_9 and all of them are members of the same DES

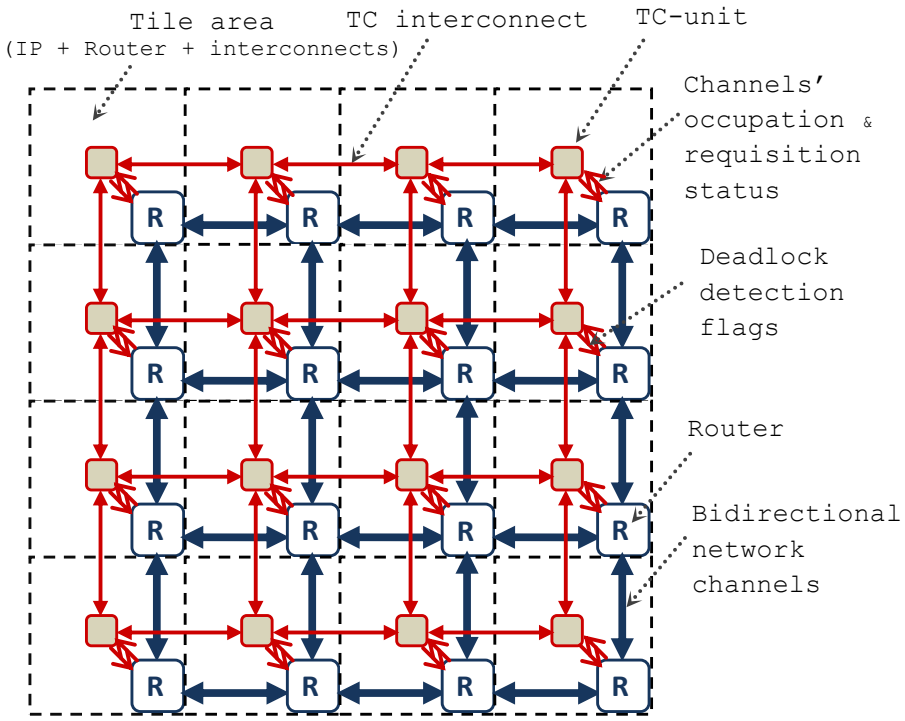


Figure 3.5: A TC-network coupled to a 2D mesh NoC.

(satisfy Eq. 3.1). To reduce the hardware complexity of TC-networks the transitive-closure algorithm can be solved by finding a path in a multiple-source single destination network. In this case, it is possible to employ a light-weight network, a few logic gates and a few wires, based on mutual exclusion units to implement the TC-network to discover the existence of a self-reflexive path for each channel (i.e., there exists i channel in TC matrix (T) where $T_{i,i} = True$). The mutual exclusion circuit ensures that only one channel can use the TC-network at any time and channels are checked sequentially using a simple token-ring protocol (hence, finding path(s) in multiple-source multiple-destination network).

One possible token-ring path for mesh and torus networks is the one presented in [22], a Hamiltonian cycle (see Figure 3.6). The token-ring could be implemented as an asynchronous circuit [22] or could be clocked using router clocks. However, the synchronous implementation is not a desirable option as it requires the same clock to span the whole chip. This work implements an asynchronous token-ring protocol circuit. However, this choice requires synchronizer circuits to achieve the synchronization with the TC-unit in each router node (see Section 3.4.3).

Circulating the token in the network is equivalent to changing the destination and finding a path in the multiple-source single destination problem. Each TC-unit seizing the token will initiate checking of the corresponding router channels. The rest of the units will implement

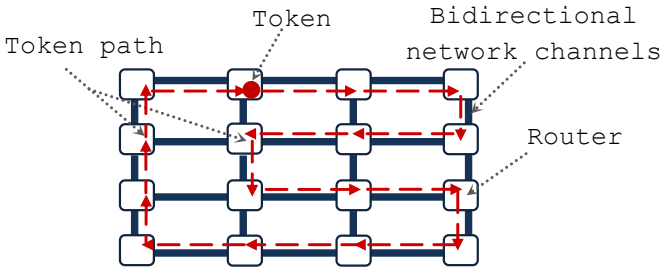


Figure 3.6: A Hamiltonian cycle [22] for a token ring distribution

a transitive property that passes the test signal to neighbour units if, and only if, there is a chain of channel dependencies between its input and output. This makes the TC-network self-pruning and will keep switching power to minimum. The function performed in the TC-unit that seizes the token is described in Algorithm 3.2. It starts checking each channel in the router node. In a case where a self-reflexive path is detected for a particular channel, line 10, it means the channel is part of a DES and that channel corresponding deadlock flag is set (line 11), which may be used by the router node to trigger a recovery. Otherwise, the deadlock flag is reset and the next channel will be tested. Once the TC-unit finishes checking all the channels of the corresponding router, it passes the token to the next neighbour unit. The delay time of the TC-network to converge and provide useful information will mainly depend on the DES size. However, the TC-network can produce a legitimate output even if it does not converge in a single clock cycle, since a deadlock is a steady and persistent event [52]. *Example 2* below illustrates, with a high level of description, how to detect deadlocks using Algorithm 3.2.

Example 2 *To illustrate* Algorithm 3.2 *in a simplified manner, consider the deadlock configuration presented in Figure 3.3a. In this case, both* n *and* m *are set to* 4*, which is the number of cardinal directions and they will be encoded as* $\mathtt{North} = 0$, $\mathtt{East} = 1$, $\mathtt{South} = 2$ *and* $\mathtt{West} = 3$*. Assume at the present time that Router* 3 *initiates the checking of its east channel (*i.e., ch_7 *in Figure 3.3), which is equivalent to selecting the source node in finding a path in a single-source multiple-destination problem. Clearly,* ch_7 *has a self-reflexive path in the TC-graph presented in Figure 3.3c and it is part of a DES. The Algorithm is detailed in the following (step by step):*

*i) For all the channels in this router (*i.e., 3 *in this case), check if there is any channel that will be selected as a source node and assert its corresponding TC signal going to the neighbour router to one (*i.e., *in this case* $\mathtt{ch\_sel}[1]$ *equal to* 1 *and therefore* $\mathtt{tc\_tx}[1] = 1$*; note that* East *is encoded to* 1 *here, as mentioned before).*

ii) For all the channels in this node, compute the transitivity property for neighbour routers ($\mathtt{tc\_tx}[i]$*), i.e., if there is a signal from the neighbour (upstream) routers (*$\mathtt{tc\_rx}[j]$ *equal to* 1*) and that channel is not*

Algorithm 3.2 Pseudo code of the TC-unit computation

Definitions :-

par denotes parallel operations.
n is the number of output channels from neighbour routers.
m is the number of output channels to neighbour routers.
$tp[m]$ are temporary buffers.

Input :-

$tc\_rx[n]$ the TC-network signals from neighbour routers,
$ch\_oc[n]$ the channels occupation status from local router,
$ch\_req[n][m]$ the channels request status from local router,
$ch\_sel[m]$ the channel to check for self-reflexive loop.

Output :-

$tc\_tx[m]$ the TC-network signals to neighbours routers,
$dd\_flags[m]$ the deadlock detection flags to local routers.

1: **par for** $i = 1$ to m **do**
2: **if** $ch\_sel[i]$ **then**
3: $tc\_tx[i] \leftarrow \mathit{True}$
4: **end if**
5: $tp[i] \leftarrow \mathit{False}$
6: **par for** $j = 1$ to n **do**
7: $tp[i] \leftarrow tp[i] \vee (tc\_rx[j] \wedge ch\_oc[j] \wedge (ch\_req[j][i] = \mathit{True}))$
8: **end par for**
9: **if** $ch\_sel[i]$ **then**
10: **if** $tp[i]$ **then**
11: $dd\_flags[i] \leftarrow \mathit{True}$
12: **else**
13: $dd\_flags[i] \leftarrow \mathit{False}$
14: **end if**
15: **else**
16: $dd\_flags[i] \leftarrow \mathit{False}$
17: $tc\_tx[i] \leftarrow tp$
18: **end if**
19: **end par for**

*empty (*i.e.ch_oc[j] *not equal to* 0*) and the flit in this channel is requesting channel* i*. Returning to the particular case under consideration, to compute the* tc *neighbour signal to the East, the* tc_rx[North] *will equal to* 1 *coming from the neighbour router in the North and the* ch_oc[North] *equal to* 1*, as it contains packet* 4 *(*P_4 *in Figure 3.3a) and* ch_req[North] = East *as* P_4 *requesting the East channel, then the* tp[East] *will equal to* 1*.*

iii) The last step is to check if each tc *neighbour signal is already selected as a source node. If this is the case and the signal itself is* 1*, then a self-reflexive path exists in this particular channel and the corresponding deadlock flag is set (*dd_flag[East] *equal to* 1*). If the channel is not the selected one for checking, then the computed signals are forwarded to the neighbour routers and the deadlock flags are reset.*

*After deadlock recovery, the variables of the particular channel that detected the deadlock (*tp[i]*) will be re-computed to* 0*, as there will be no more channel dependencies, and then the corresponding* dd_flag *will be reset.*

3.4.3 *Hardware Implementation*

Figure 3.7 shows a schematic for the proposed router for a general NoC topology, with n input channels and m output channels, implementing a deadlock detection method based on TC-computation and TC-interconnects. As shown, the top level scheme of the router consists of a set of input buffers. The schematic does not show the input buffer for the traffic locally generated by IP core attached to the router because it is not relevant to this study's particular deadlock detection method, assuming a finite time is required to consume a packet at the destination. The input buffers are connected to a block implementing the routing and arbitration algorithms. A fully adaptive routing algorithm with minimal paths is used. The routing algorithm used in this work sets no restrictions on the routing turns in the network and does not use any virtual channels ordering; hence, deadlocks may occur. The TC-interconnects and the TC-unit are implemented to efficiently detect all true deadlocks. Figure 3.7 illustrates the router architecture with n input channels and m output channels which can be used for different network topologies.

The content of a TC-unit block is shown in the central part of Figure 3.7 in more detail. It consists of three components: a first block (TC-computation) implementing the transitive property (described in Eq. 3.2 and Eq. 3.3), a second block (channel selection circuit) implementing a channel sequence selector with a time delay based on the worst case delay scenario (presented in Section 3.5.2.4) and a token-ring protocol to initiate checking of the router channels attached to it.

The TC-computation block exploits the TC-interconnects, tc_rx[], provided by adjacent neighbours (upstream routers); the local channels'

occupation, ch_oc[], and request, ch_req[][], provided by the local router and the selected channel, ch_sel[], produced by the channel selection circuit. It generates the TC-interconnects, tc_tx[], to adjacent neighbours (downstream routers) and deadlock detection flags, dd_flags[], to the local router. The size of these signals depends on the NoC topology; for instance, for a 2D mesh network $n, m \in \{North, East, South, West\}$. The description of the TC-computation can be found in Algorithm 3.2 and the circuit is shown in Figure 3.7 (top-right).

The token-ring protocol is used to change the destination node and initiate the checking of channels at that router node. A possible implementation of the asynchronous token-ring protocol is to use the David cell (DC) circuit [158], shown in Fig 3.8a. The DC consists of three NOR gates of which two are cross-coupled (1&2) and the third acts as a gating mechanism for the output r. A one hot sequence structure can be formed using a chain of DCs as shown in Figure 3.8b. In order to circulate the token, the DC-chain should have one DC in a set condition, i.e., output $e = 0$ and $f = 1$, and the rest of the DCs in the reset condition, i.e., output $e = 1$ and $f = 0$. The output of each DC in the chain could be broken and used as the *start* and *done* signals of the channel selector circuit. However, the *start* signal should be synchronized with the local router clock, when a synchronous router is used in the NoC. A typical synchronizer circuit can be found in [97] which uses two edge-triggered flip-flops clocked by the receiving circuit's clock with a time delay between the clocking of the first and the second flip-flops.

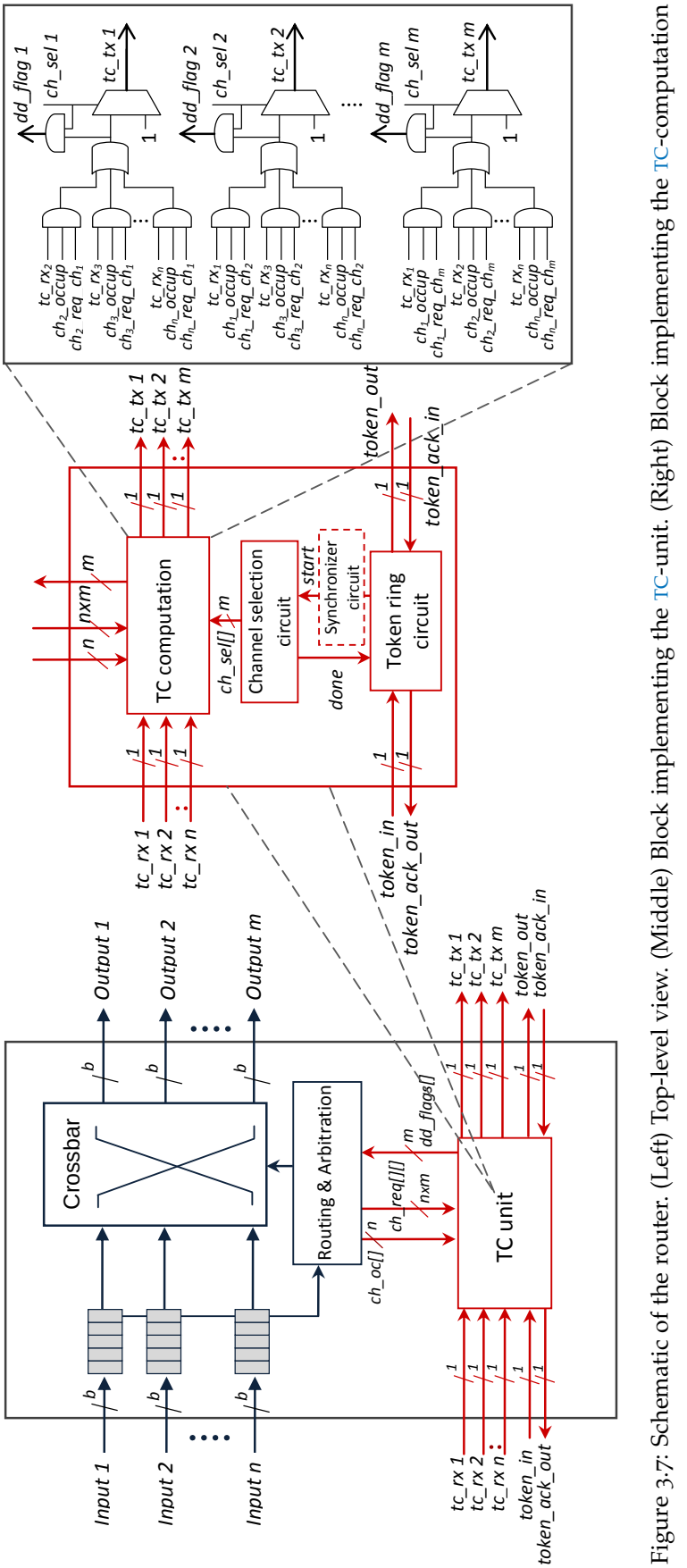
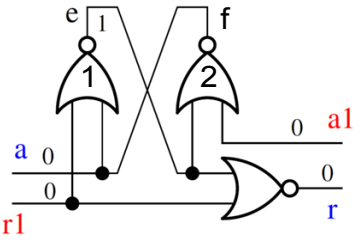
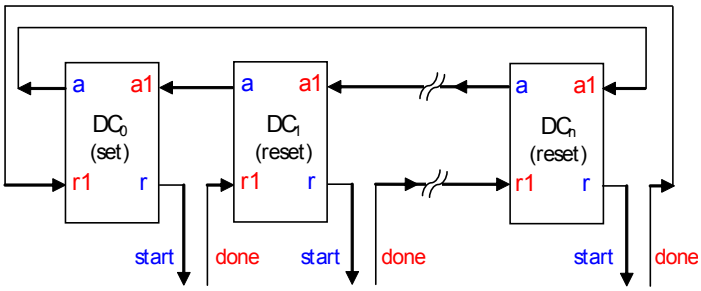


Figure 3.7: Schematic of the router. (Left) Top-level view. (Middle) Block implementing the TC-unit. (Right) Block implementing the TC-computation



(a) David-Cell circuit (in reset condition) [158]



(b) Chain of David-Cells (DC)

Figure 3.8: An example of token-ring protocol implementation

The asynchronous implementation of the token-ring requires two wires from the previous router in the Hamiltonian chain and two wires to the next router in the chain. Once a TC-unit seizes the token, it will initiate a *start* signal to the channel selection circuit. After checking all channels, a *done* signal is sent back to the token-ring circuit to pass the token to the next neighbour router. The implementation presented in Figure 3.7 illustrates that the router requires some additional circuits and interconnects in order to perform the proposed deadlock detection. These requirements are evaluated and compared, in Section 3.5.2.3, with similar router architecture using the state-of-the-art timeout detection mechanism.

The architecture introduced in Figure 3.7 is only one of the possible router implementations. Different optimizations could be applied, depending on the particular case under consideration. A point of interest could be the generalization of the approach in order to check more than one router channel simultaneously. This can be accomplished by circulating more than one token in the token-ring protocol chain, thus activating more than one router in the network to perform the checking of its channels. It should be noted that a further TC-unit block and TC-interconnect would be required, thus leading to additional area/power increase. However, for the investigated traffic patterns and network topology the results are not substantially different.

3.5 RESULTS AND DISCUSSION

3.5.1 *Evaluation Methodology*

The TC-network is evaluated in this section by comparing the percentage of detected deadlocks with the heuristic timeout mechanism [22] for different network traffic and different flits injection rate. The percentage of detected deadlocks is obtained as the ratio between the number of packets detected as deadlock over the total number of packets (received and detected). Once a packet is presumed as deadlocked, it is removed from the network. The consumed network energy caused by dropping detected packets is computed. In general, the energy dissipated by a network is divided into the following categories: 1) Routing energy, which depends on the routing type; 2) Selection energy, which refers to the type of selection if the routing algorithm returns more than one option; 3) Forwarding energy, which is used in sending a flit; 4) Receiving energy, which is used in receiving a flit and 5) Waiting energy, which is related to the time the header flit remains waiting until a successful routing takes place. The packet dropping energy is defined as:

$$\begin{aligned} E_{dd\_resolving} = h \times \left(E_{forward} + E_{receiving}\right) + Flit_{age} \\ \times \left[E_{wait} + f \times \left(E_{routing} + E_{selection}\right)\right] \end{aligned} \quad (3.4)$$

where E denotes energy, $E_{dd\_resolving}$ is the energy consumed to resolve any detected deadlock, h is the number of hops the flit passed before being aborted, and $Flit_{age}$ is the number of clock cycles the flit existed in the network (either moving or waiting for resources to be freed) and f is a Boolean flag that adds the last term to the equation if the flit type is head (if the blocked flit is head it will continue consuming energy by trying to reserve an output channel in each clock cycle).

The performance evaluation was carried out using a modified version of Noxim [62]. In particular, Noxim is modified by introducing the TC-network and the timing-based deadlock detection techniques [22, 124, 140]. Without any loss of generality, a NoC with the following specification is chosen for the evaluation: a mesh topology with five ports router architecture, fully adaptive routing with random selection function, no virtual channels, and a crossbar switch - these can be used in a wide range of NoC configurations. Each input channel consists of four flit buffers and one clock cycle is assumed for routing and transmission time across the crossbar and a channel. The results are captured after a warm up period of 10,000 clock cycles. The overall simulation time is set to 300,000 clock cycles. To ensure the accuracy of results captured, with a higher degree of confidence, the simulation

at each injection rate is repeated five times with different seeds and their mean values taken.

3.5.2 *Evaluation Results*

3.5.2.1 *Synthetic Traffic*

Figures 3.9, 3.10 and 3.11 show the performance results for a 4×4 2D mesh NoC and a uniform distribution of packet destinations[1] with different injection rate. The packet lengths are randomly generated between 2 and 16 flits. Examining Figure 3.9, the majority of detected deadlocks using the timeout mechanism [22] are false alarms. For instance, 22% of the packets injected in the network at higher injection rate are detected as part of deadlocks with the threshold value set to 32. The TC-network instead detected that less than 1% of packets are in true deadlocks, consistent with literature [162] which stated that deadlock is an infrequent event.

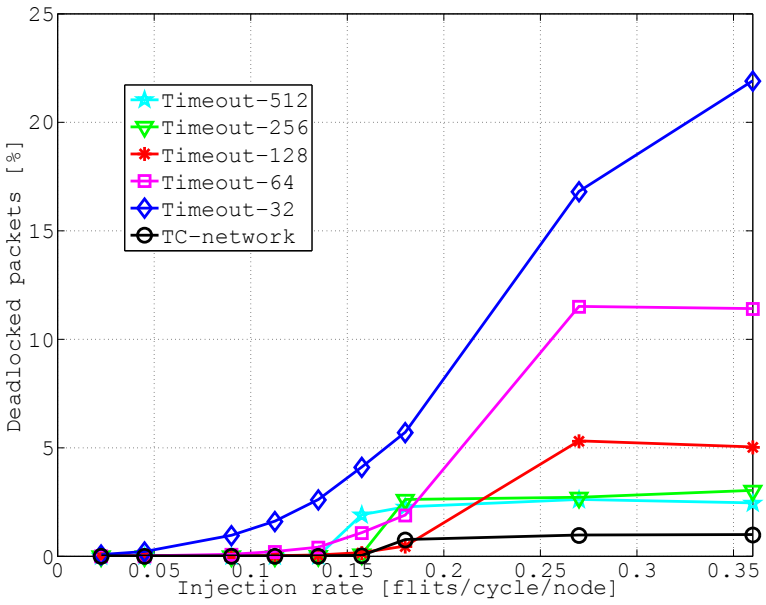


Figure 3.9: Percentage of detected deadlocked packets to the total received packets using the TC-network and the timeout mechanism under *uniform* traffic scenario

Figure 3.10 shows the network average delay versus the throughput for full load range. It shows that a smaller threshold value used in the timeout mechanism improves these two important network metrics

1 This kind of traffic pattern is the most commonly used traffic in network evaluation [52] although it is very gentle because it naturally balances the load all over the network.

because it detects more false deadlocks and, by dropping them, will alleviate network contention that may turn into congestion.

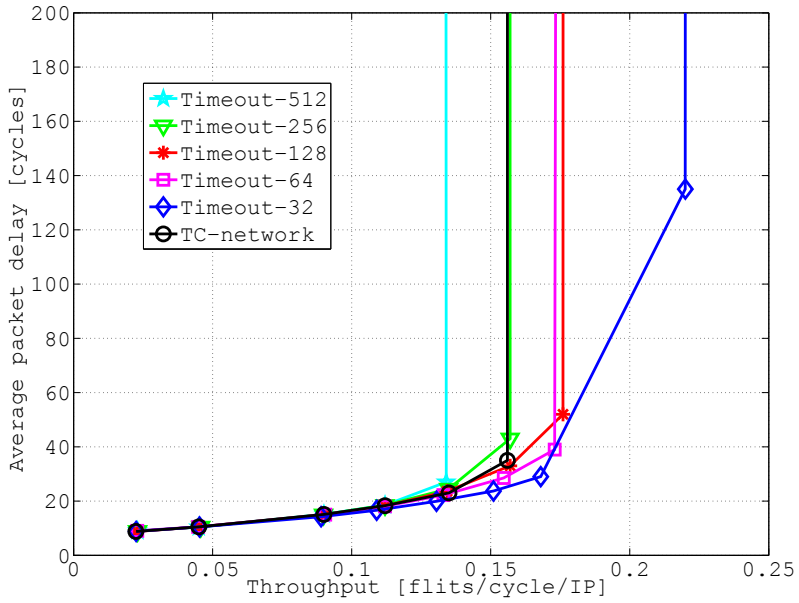


Figure 3.10: Performance of the TC-network and the timeout mechanism under *uniform* traffic scenario with different injection rates

Examining figures 3.9 and 3.10, the threshold value of 256 could be selected as the best value for such a network setting as it produces a minimum detection percentage of 2.7% with good throughput and latency. The selection of the best threshold value for different network settings (packets' length, buffer size and traffic type) was the goal of several studies [112, 124, 140, 105]. However, the TC-network method detects true deadlocks and produces similar throughput and latency figures to the timeout mechanism with a 256 threshold value and without the need for any parameters tuning.

Figure 3.11 shows the percentage of energy consumed to resolve detected deadlocks to the total network consumed energy. The figure is similar to, but not linearly proportional to the detection percentage figure (Figure 3.9). This is because significant energy consumption is caused by the routing function [62] which repeatedly tries to route the head flit until the time out value has elapsed in the case of the time out method (see Eq.3.4). For instance, the timeout-128 detects 5.3% at saturation and it wastes 8.8% of the total consumed energy by aborting these packets, while the Timeout-512 detects 2.6% at saturation and wastes 9.1% energy. There are two underlying reasons for this: the first is that the network with a bigger threshold value is delivering fewer flits at the given simulation time (Figure 3.10); the second reason is that the Flit_{age} in Eq.3.4 is directly proportional to the threshold value in the case when a packet is detected as deadlocked.

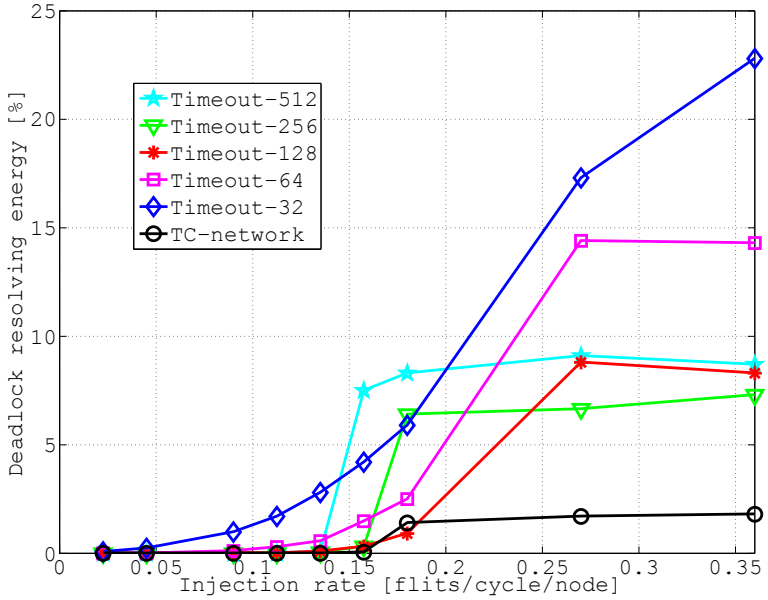


Figure 3.11: Percentage of total energy consumed to resolve detected deadlocks to the total NoC energy under *uniform* traffic scenario

To investigate different traffic scenarios and network sizes, figures 3.12, 3.13 and 3.14 show the performance results with the *bit-reversed* traffic[2]. The network size is 8×8 mesh and the packet sizes are randomly chosen between 32 and 64 flits. The results, in general, show a similar trend to the previous example. Here, the threshold value of 1024 could be selected as the best threshold. The TC-network method detects around 0.07% of packets as deadlocked and dropping them consumed energy of less than 0.2% compared to 4% detected using Timeout-1024 and a waste of energy of around 10%.

Moreover, this study implemented the deadlock detection techniques proposed in [124] and [140], which were proposed to enhance the accuracy of deadlock detection over the crude timeout method [22], i.e., to reduce the false-positive-deadlock alarms. The timeout threshold for each simulated technique is selected as 16, 32, 64, 128, 256, 512, and 1024 clock cycles, but only the results for 64 and 256 cycles are presented, since the only major difference is the measures scaling up or down. In the following figures, the deadlock detection methods proposed in [22, 124] and [140] will be labelled as *crude*, *sw* and *fc3d* respectively, followed by the used timeout threshold value.

For the rest of the synthetic traffic patterns, the results are summarized in Table 3.2. The results are presented for a single injection rate

2 Each node with binary address $\{b_{n-1}, b_{n-2}, \ldots, b_0\}$ sends a packet to the node with address $\{b_0, b_1, \ldots, b_{n-1}\}$.

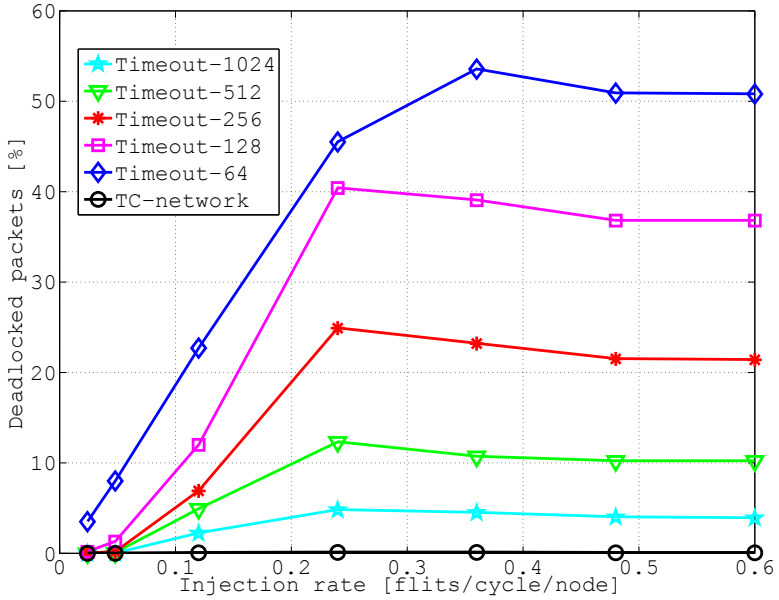


Figure 3.12: Percentage of detected deadlocked packets to the total received packets using the TC-network and the timeout mechanism under *bit-reversed* traffic scenario

which is labelled as the saturation packet injection rate (PIR)[3], since the majority of detections occur after the network saturation point. Table 3.2 illustrates the efficacy of the proposed run-time deadlock detection method over the three different existing timing based methods [22, 124, 140]. The TC-network method significantly surpasses timing-based deadlock detection mechanisms by avoiding false detections and, as a result, reducing energy wastage to resolve all detected deadlocks for all synthetic traffic scenarios presented in the table. It should be noted that the quantitative analysis presented in the last row of Table 3.2 as "TC-improvement" would vary according to the injection rate and it is used here to summarize, on average, by what factor the TC-network reduces the detected deadlocked packets compared to the timing-based schemes [22, 124, 140]. Therefore, the TC-improvement magnitude is not necessarily the same in the case where the evaluation incorporated the entire load range. The results presented are for 8×8 mesh with packet sizes randomly chosen between 32 and 128.

3 A network starts saturating when an increase in injection rate does not result in a linear increase in throughput [26].

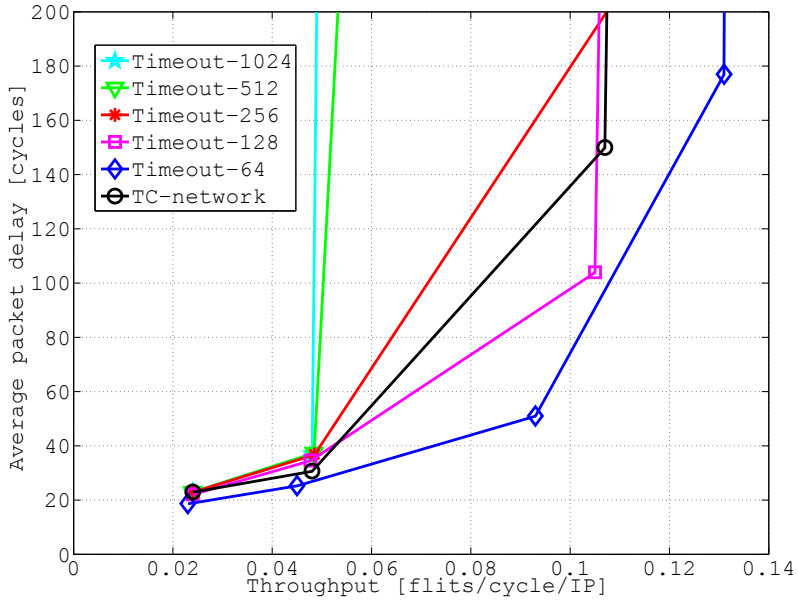


Figure 3.13: Performance of the TC-network and the timeout mechanism under *bit-reversed* traffic scenario with different injection rates

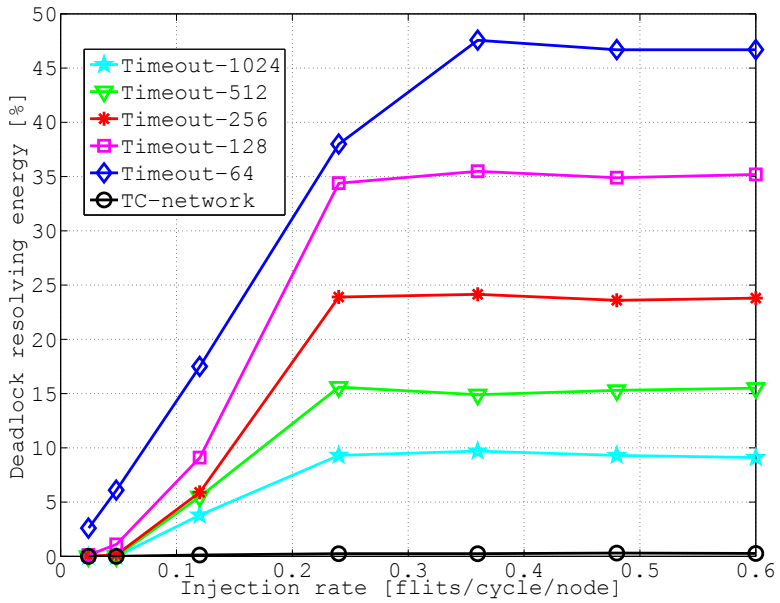


Figure 3.14: Percentage of total energy consumed to resolve detected deadlocks to the total NoC energy under *bit-reversed* traffic scenario

		Timeout-64[22]		Timeout-256[22]		Timeout-64[124]		Timeout-256[124]		Timeout-64[140]		Timeout-256[140]		TC-network	
Traffic	PIR(Sat.)	DD%	Er%	DD%	Er%	DD%	Er%	DD%	Er%	DD%	Er%	DD%	Er%	DD%	Er%
Shuffle	0.0035	49.5	39.8	32.8	29.6	33.1	25.1	19.9	17.0	24.8	16.9	15.8	12.2	0.10	0.29
Transpose	0.015	35.2	27.6	12.6	11.1	10.8	7.30	1.60	1.30	3.11	1.93	0.69	0.43	0	0
Butterfly	0.005	28.9	33.8	11.7	16.7	16.7	17.3	3.70	4.60	9.61	10.3	2.07	2.46	0	0
Random[1]	0.001	18.1	16.3	2.82	2.70	7.40	5.60	1.80	1.30	6.26	4.50	2.14	1.46	0.07	0.11
Random[2]	0.001	20.2	15.5	4.90	4.60	8.63	5.53	3.90	3.00	7.10	4.87	3.22	2.21	0.69	1.22
TC Improvement		176x	83x	75x	41x	89x	70x	36x	31x	59x	45x	28x	22x	-	-

DD%: Percentage of detected deadlocks, Er%: Percentage of total energy consumed to resolve detected deadlocks, Random[1]: Uniformly distributed traffic with four hot spots located at the corners, Random[2]: Uniformly distributed traffic with four hot spots located at the centre

Table 3.2: TC-network improvement compared to timing based methods for different threshold values and traffic scenarios.

3.5.2.2 *Real Application Traffic*

As a more realistic communication scenario, a generic multi-media system (MMS) [80], which includes an *h263* video encoder, an *h263* video decoder, an *MP3* audio encoder, and an *MP3* audio decoder, is evaluated. In [80], the application is partitioned into 40 distinct tasks, which are then assigned and scheduled onto 25 selected IP cores. The topological mapping of IP cores into tiles of a 5×5 mesh-based NoC architecture has been obtained by using the approach presented in [25]. As can be observed from Figures 3.15 and 3.17, again the TC-network method outperforms the timing based deadlock detection methods with various threshold values. Interestingly the TC-network detects zero deadlocks and thus does not consume energy for retransmission for this particular traffic, while the crude timeout method [22] detects a considerable number of false deadlocks and wastes a considerable amount of energy in recovering from these false deadlocks. For instance, up to 27% of the MMS traffics are detected as deadlocks with a 64 threshold value. The other timing-based deadlock detection methods [124, 140] reduce the false deadlock alarms, compared to the crude timeout method [22], but they do not eliminate them.

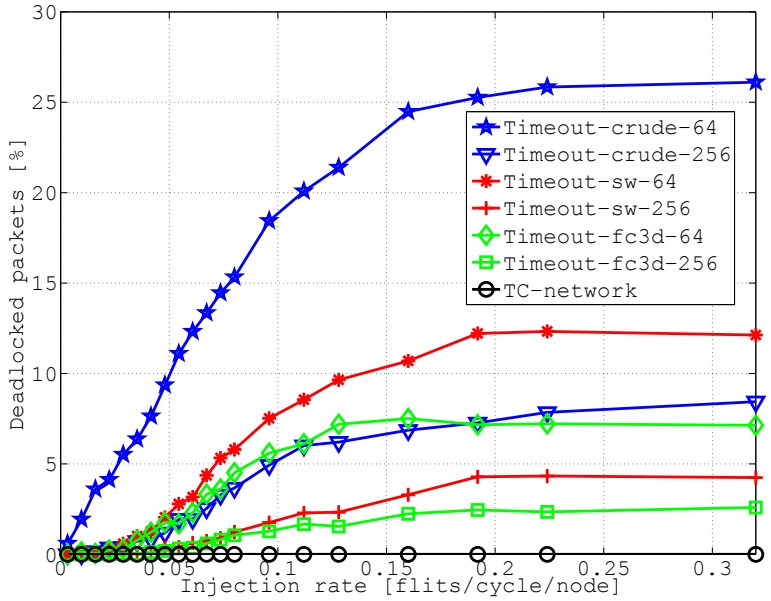


Figure 3.15: Percentage of detected deadlocked packets to the total received packets using different methods under MMS

Figure 3.16 shows the network average delay versus the throughput for full MMS load range. Once again, the result suggests that a smaller threshold value used in the timeout mechanism improves these two network metrics. This is because it detects more false deadlocks and, by dropping them, alleviates network contention that may lead to

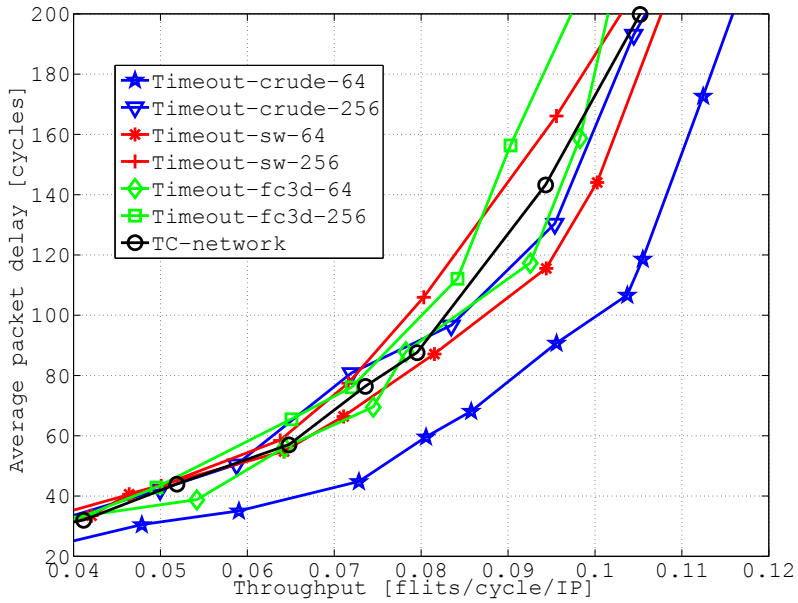


Figure 3.16: Performance of the TC-network and the timeout mechanism under MMS with different injection rates

congestion. However, timing-based detection methods with smaller threshold values increase the amount of wasted energy for retransmission. Therefore, another experiment was conducted to evaluate the total energy (transmission + retransmission) required to drain a fixed amount of data (2 Mbytes MMS traffic) for different network loads. Table 3.3 shows the result of this experiment using different deadlock detection techniques. The last column in the table shows the accumulated energy saving using the proposed detection method for the three injection rates presented in the same table. The energy saving is different for the different detection methods used and for different threshold values; for instance, the TC-network can save up to 5% of energy to run the 2 Mbytes MMS traffic compared to the timing based method [22] with a threshold value equal to 64.

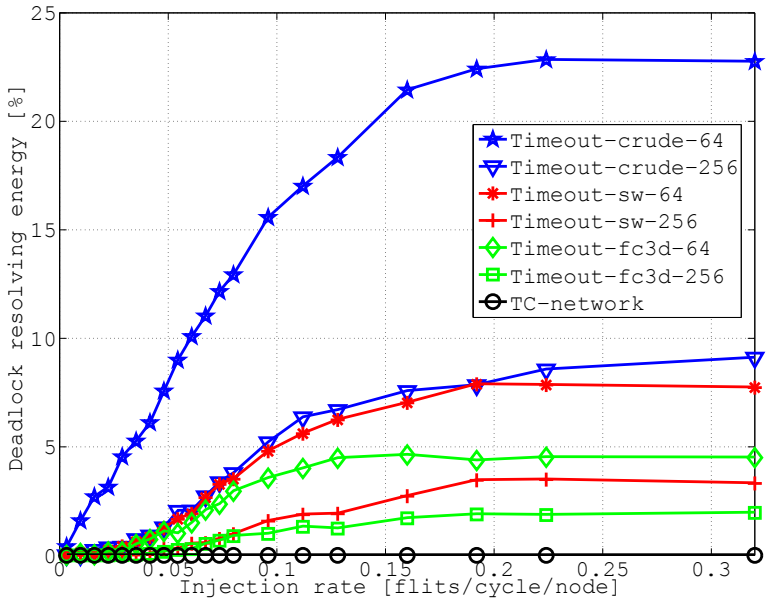


Figure 3.17: Percentage of energy consumed to resolve detected deadlocks (retransmission) to the total NoC energy (transmission + retransmission) under MMS

Method Used	Threshold Value	Injection rate = 0.05			Injection rate = 0.15			Injection rate = 0.2			Accumulated Energy Saving (%)
		$E_{trans.}$	$E_{retrans.}$	E_{total}	$E_{trans.}$	$E_{retrans.}$	E_{total}	$E_{trans.}$	$E_{retrans.}$	E_{total}	
TC-network	-	0.7152	0.0	0.7152	0.8040	0.0	0.8040	0.8128	0.0	0.8128	-
Timeout [22]	64	0.7002	0.0366	0.7368	0.6736	0.1696	0.8432	0.6693	0.2068	0.8761	5.05
Timeout [22]	256	0.7204	0.0035	0.7239	0.7548	0.0566	0.8114	0.7530	0.0787	0.8317	1.48
Timeout [124]	64	0.7181	0.0046	0.7226	0.7497	0.0621	0.8117	0.7653	0.0671	0.8324	1.47
Timeout [124]	256	0.7247	0.0004	0.7230	0.7911	0.0223	0.8134	0.7987	0.0301	0.8288	1.40
Timeout [140]	64	0.7136	0.0036	0.7171	0.7864	0.0362	0.8226	0.7873	0.0381	0.8244	1.36
Timeout [140]	256	0.7182	0.0002	0.7184	0.8090	0.0127	0.8218	0.7978	0.0155	0.8133	0.91

Table 3.3: Energy consumption (*mJ*) to drain 2 Mbytes of multi-media system data for different network loads and different deadlock detection methods

3.5.2.3 *Area and Power Estimation*

It is crucial in designing NoCs that routers should not consume a large percentage of silicon area compared to the IP core blocks. For this study, two fully adaptive routers based on the timeout and the TC-network methods were designed in Verilog. These were then synthesized using Synopsys Design Compiler and mapped onto the UMC 90nm technology library. Confirming well-known findings from, for instance, [26] and [28], the hardware synthesis result shows that the first in, first out (FIFO) buffer area significantly dominates the logic of the router. The buffer area mainly depends on the flit size. For a flit size of 64 bits and FIFO buffers with a capacity of four flits, it was found that the TC circuit adds only 0.71% area overhead to the total router area compared to 2.9% for the timeout [22] implementation with 10 bit threshold counter. Thus the proposed method yields an area gain of more than 2% in each network router circuit compared to the timeout implementation.

Power consumption is also an important system performance metric. The power dissipated in each router design was determined by running Synopsys Design Power on the gate-level netlist of the router with different random input data streams as test stimuli. It was found that the power dissipated by the TC-unit is 0.44% of the entire router power compared to 0.97% dissipated by the crude timeout circuit, with 10 bit threshold counter. This gives a power saving of 0.53%. In the Intel TeraFlop 80-tiles chip implementation [157] the communication power (Router + Links) estimation is 28% of the tile power profile. The Router power, however, is 83% of the communication power. Taking these numbers into consideration, and the power result from the proposed router synthesis, suggests that the TC-unit will dissipate less than 0.01% of the total tile power in similar NoC implementations.

Moreover, an investigation was carried out into the area and power contributions of different timing-based deadlock detection methods' circuits with different time-threshold values and they were compared to the TC-unit circuit implementation. The area gain and the power saving with the TC-unit implementation compared to different timeout implementation can be observed in Table 3.4. The table clearly shows that the techniques used in [124] and [140] introduce more area and power overheads compared to the crude timeout implementation and this is due to the extra hardware requirements required to implement these techniques. The TC-network implementation, however, not only improves the performance of the deadlock detection method, but also minimizes the area and power overheads, as can be seen in Table 3.4.

However, the implementation of the TC-network requires some extra control wires, as shown in Figure 3.7. The tc_rx[n] input wires and tc_tx[m] output wires to/from each TC-unit are coming/going to TC-units in neighbour router nodes. The *token_in*, *token_ack_out*, *token_out* and *token_ack_in* signals are used to propagate asynchronously the

token, i.e., changing the destination node, in the token-ring protocol implemented in this work. Overall, the total number of wires required to support the deadlock detection based on TC-network is

$$TC\_Network_{wires} = n + m + 4, \tag{3.5}$$

where n is equal to the number of input channels and m is equal to the number of output channels in each of the router nodes. Considering the data used in the experiments (2D mesh network where $n, m \in \{North, East, South, West\}$ and flit size of 64 bits), the TC-Network wires are 12 in this case, which is less than 2% of the total wiring cost. This number is small and will become smaller with a bigger flit size, as well as being independent of the capacity of the input and output buffers.

Module name	Proposed in	Module area to total router area	Module power to total router power
TC-unit	This work	0.71%	0.44%
Timeout-1024	[22]	2.93%	0.97%
Timeout-256	[22]	2.33%	0.81%
Timeout-64	[22]	1.56%	0.70%
Timeout-1024	[124]	3.12%	1.16%
Timeout-256	[124]	2.54%	0.98%
Timeout-64	[124]	1.76%	0.91%
Timeout-1024	[140]	3.53%	1.51%
Timeout-256	[140]	2.94%	1.38%
Timeout-64	[140]	2.12%	1.24%

Table 3.4: Area and power contributions of the TC-unit and different timeout circuits to the total router area and power

3.5.2.4 *Delay Estimation*

In order to study the operation delays, first the TC-unit's critical path gates delay is calculated using the *sdf* file generated after synthesizing the circuit using worst case library. Secondly, the interconnect delay calculation assumes the tiles are arranged in a regular manner on the floorplan with 2mm × 1.5mm tile size, similar to the Intel TeraFLOPS chip [157]. The maximum interconnect length between routers is 2 mm. The load wire capacitance and resistance are estimated, using the Predictive Technology Model (PTM) [7], to be 0.146fF and 1.099 Ohm per micron respectively. The wire delay between TC units, TC-interconnect, can be readily calculated based on the distributed RC model [139]. In

reference to Table 3.1, the worst and average convergence times of the TC-network for different network topologies can be estimated. Figure 3.18 shows the worst and average times to discover a deadlock for different network topologies and sizes. It is expected that the delay required by the TC-network to converge to the desired output will depend on the network topology and the size of the DES.

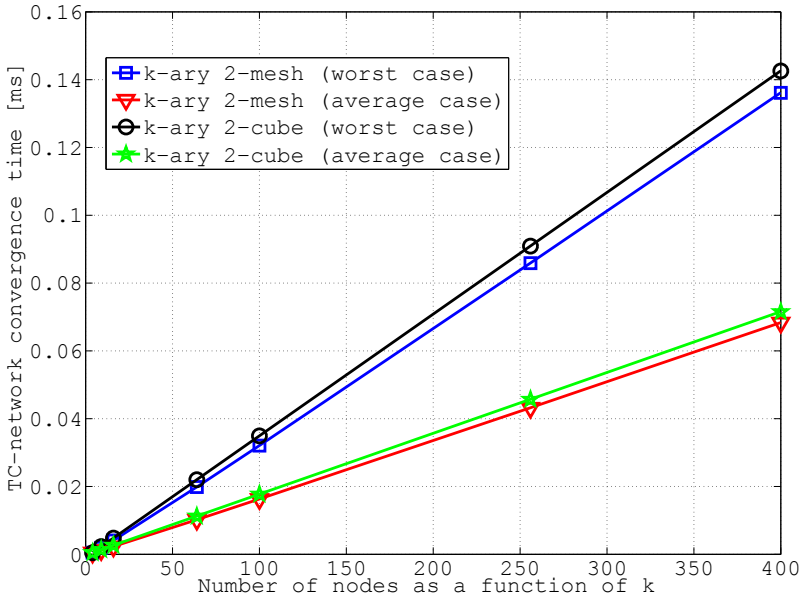


Figure 3.18: TC-network convergence time for different network topologies and sizes

3.6 DEADLOCK RECOVERY SYSTEM

The evaluation results presented in the previous sections show that the TC-network is an effective method to detect true deadlocks. This section proposes a new regressive recovery technique that augments the well-known end-to-end communication protocol to handle detected deadlocked packets. This technique then evaluated with the proposed deadlock detection method, TC-network, and compared with other timing-based detection methods and avoidance methods.

3.6.1 *End-to-End Recovery Protocol*

The *end-to-end* communication protocol is a standard transport layer protocol used to provide a reliable communication between any source-destination pairs in a network by retransmitting corrupted or lost packets [52, 56]. There are several variations of the end-to-end protocol, but in general the main idea behind it is quite simple: the source sends

data, which could be a single packet or a block of packets, and retains a copy of this data in a buffer waiting for the target destination (receiver) to acknowledge (send *Ack* packet) before releasing the data in the buffer and proceeding to send a new data. If the source (sender) receives a negative acknowledgment (*Nack*) instead, which means the destination received a corrupted data, then the source will start resending the same data again. In a case where the data is lost in the network because of any kind of fault and never reached the intended destination, the sender will wait for a prior specified time (timeout) and, if during that time no *Ack* or *Nack* is received, the sender will initiate sending a copy of the data in the buffer assuming that the first data transmitted is lost in the network. The last scenario may produce several copies received by the destination and this can be solved in the transport layer by ignoring the duplications [52].

There is a similar communication protocol known as the *switch-to-switch* communication protocol, except that flits rather than packets are transmitted and retransmitted at the link level. The data (flit) integrity is checked at each switch (a.k.a. router) in the network and it requests flit retransmission from a previous switch if the received data is corrupted. This switch level retransmission protocol requires extra buffers in each switch and thus produces higher area and power overheads in NoCs [35] compared to the end-to-end protocol. Therefore, the latter tends to consume less power, but the transmitted data undergoes high delay and thus NoCs can suffer from early saturation [91, 165].

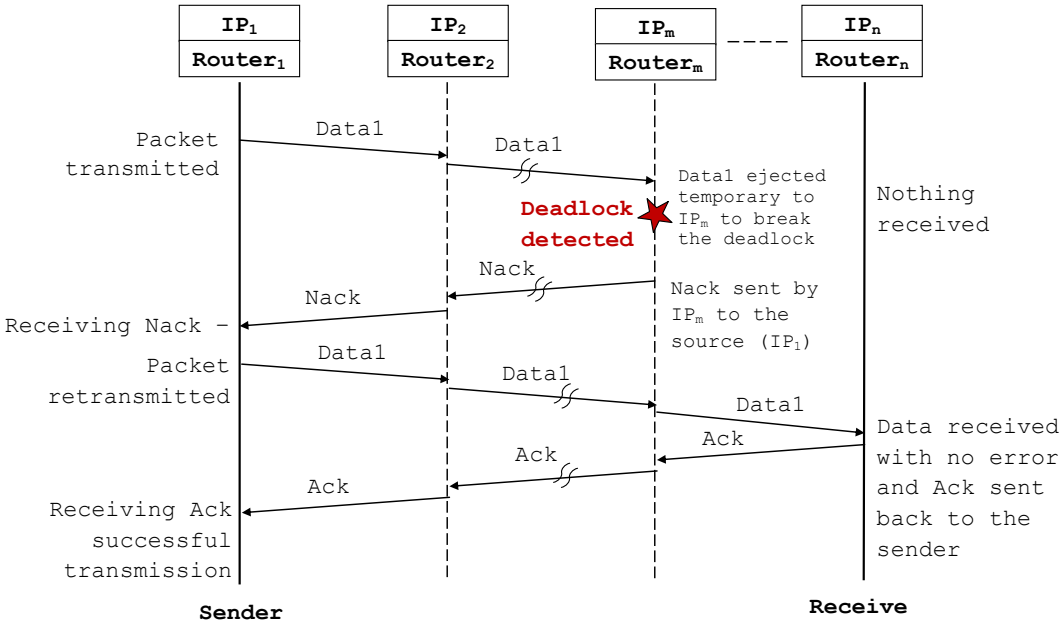


Figure 3.19: Transmission scenario 1, which illustrates how to resolve the deadlock if it forms part of the *data* packet.

In this section, the objective is to provide a recovery mechanism (protocol) to recover the detected deadlocked packets and evaluate the proposed new NoC-based deadlock detection method (TC-network) in a complete deadlock recovery system, i.e., consisting of deadlock detection and deadlock recovery mechanisms. The assumption is made in this section that the target NoC infrastructure is already utilizing the end-to-end communication protocol to ensure the reliability of data transmitted over the network. This, by itself, can be used to support a regressive deadlock recovery method because any killed deadlocked packets will be retransmitted after timeout specified in the source IP. However, choosing the best timeout value is challenging due to high correlation with network status (load, message length, traffic distribution, etc.). Thus, a modification is made to the original end-to-end protocol to avoid using any timeout counter in recovering detected deadlocked packets. The recovery process can be stated as follows: whenever a packet is detected as being deadlocked, the destination address of that packet is changed to the local IP and a flag is set to indicate a premature ejection. This work assumes at each IP there is enough storage available in order to prevent any dependency cycle. At the transport layer, the local IP will take one of three possible actions based on the type of the received packet. In such a NoC any deadlocked packets can be one of three different possible types at any time. These are: 1) the transmitted data packet; 2) the acknowledgment packet (*Ack*) and/or 3) the negative acknowledgment packet (*Nack*). It is important to note that *Ack* and *Nack* packets each consist of a single flit (wormhole switching).

To illustrate how the local IP will respond in the case of a premature packet being received:

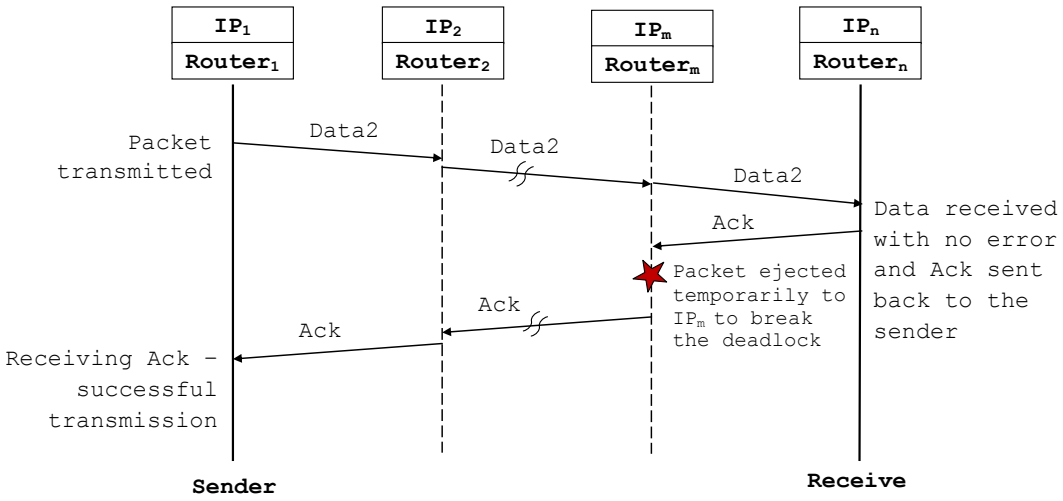


Figure 3.20: Transmission scenario 2, which illustrates how to resolve the deadlock if it forms part of the *Ack* packet.

Figure 3.19 illustrates the first scenario where the data packet (*Data1*) becomes part of a detected deadlock at Router_m and this triggers the recovery process. The packet is ejected to the local IP. The IP will extract the source information from the head flit and ignore the rest and schedules to send *Nack* to the source (*IP1* in Figure 3.19). Eventually, the sender receives the *Nack* and retransmits *Date1* again. Figure 3.20 shows the case where *Data2* is received by the destination intact and an *Ack* packet is sent back to the source to acknowledge *Data2*. However, the *Ack* packet is detected as part of a deadlock at Router_m. The *Ack* is ejected temporarily to the local IP to break the deadlock dependency cycle and a new *Ack* is sent to the source. The third scenario is illustrated in Figure 3.21, where *Date3* is received tampered by the destination and a *Nack* packet is sent back to the source to indicate that. However, the *Nack* packet is detected as part of deadlock at Router_m. The *Nack* is ejected temporarily to the local IP to break the deadlock dependency cycle and a new Nack is sent to the source of *Date3*, requesting retransmission. These three scenarios, presented in Figures 3.19, 3.20 and 3.21, are the only possible unique situations and any other situations could be a mixture of these three. As presented earlier in this work, the proposed protocol handles these situations efficiently. In the next section, the proposed augmented end-to-end communication protocol is evaluated in a NoC with a complete deadlock recovery system.

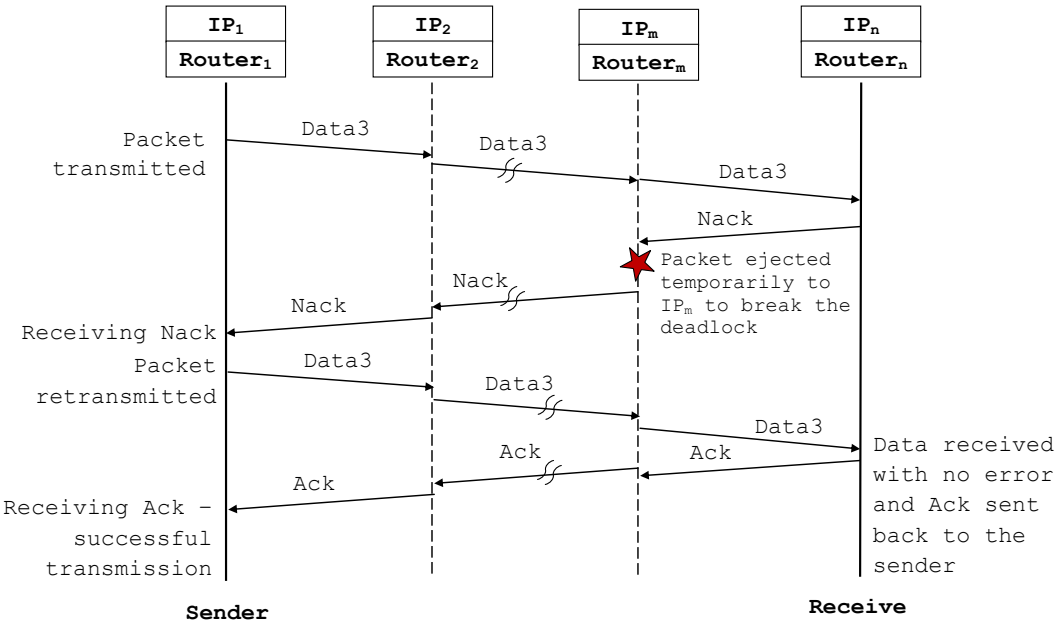


Figure 3.21: Transmission scenario 3, which illustrates how to resolve the deadlock if it forms part of the *Nack* packet.

3.6.2 *Deadlock Recovery System: Evaluation Results*

The aim in this section is to evaluate the proposed method in a full deadlock recovery system. Several experiments are conducted for the previously described reliable *end-to-end* transmission protocol with an 8×8 2D mesh NoC and *shuffle* distribution traffic. The packet length consists of 32 flits. The TC-network is evaluated by comparing the percentage of detected deadlocks, percentage of energy consumed for retransmission, throughput and latency with the heuristic timeout mechanisms, namely the crude timeout [22], the software-based timeout [124] and the flow control-based timeout [140]. Table 3.5 shows the percentage of detected deadlocks and the percentage of energy used for retransmission for each of the deadlock detection approaches mentioned earlier. A threshold time of 32 clock cycles is used for this experiment. Clearly the *crude* timeout mechanism performs the worst among the all methods by detecting up to 52% of the traffic as deadlock at high load and consuming around 32% of the communication energy to recover these false positive detections. The other two sophisticated timeout mechanisms (*sw* and *fc3d*) performed much better than the *crude* timeout mechanism by reducing the false deadlock detections and thus reducing the energy required for retransmission. The proposed TC-network, however, outperforms the three techniques by eliminating all the false deadlock alarms and thus substantially reducing the energy reburied for retransmission. The network throughput and average delay for the four deadlock detection methods are shown in Figure 3.22.

The figure clearly shows that the TC-network outperforms the timing based deadlock detections method in terms of throughput and average network delay. This can be due to the following: 1) the TC-network detects deadlocks quickly without the need to wait for a specific threshold time; 2) the timing based methods detect a lot of packets as deadlocked and recovering all these packets will exhibit a big delay and will add burden on the network performance. The timing based methods consume a considerable amount of communication energy for recovering detected packets (most of them are false alarms). Thus another experiment is conducted to measure the energy required to transmit 1 *Mbyte* of data through the network with a different injection rate. The three timing based approaches and the TC-network approach are employed for deadlock detection, while the recovery protocol used in the experiment is the *end-to-end* protocol. Figure 3.23 shows the consumed energy to drain the specified bandwidth with different injection rates. Clearly, the TC-network consumes less energy to send the data over the NoC for the full load.

To compare the performance of the deadlock recovery methodology with the deadlock avoidance methodology, a new experiment is conducted with the same settings as above. The objective here is

	Timeout-crude-crude-32 [22]		Timeout-sw-sw-32 [124]		Timeout-fc3d-fc3d-32 [140]		TC-network	
IR	DD%	Er%	DD%	Er%	DD%	Er%	DD%	Er%
0.016	3.7	1.5	0.5	0.1	0.1	0.07	0.0	0.0
0.032	10.9	4.0	1.2	0.3	0.1	0.02	0.0	0.0
0.048	19.9	8.1	2.5	0.8	0.5	0.1	0.0	0.0
0.064	26.8	11.7	5.3	1.7	1.0	0.2	0.0	0.0
0.080	37.4	18.4	8.9	3.1	1.8	0.5	0.0	0.0
0.096	47.4	25.8	15.7	6.2	4.3	1.3	0.0	0.0
0.112	51.1	29.6	23.3	10.4	8.6	2.8	0.05	0.05
0.128	53.1	31.3	25.9	12.1	9.8	3.4	0.05	0.07
0.144	52.9	31.4	27.6	13.4	9.9	3.5	0.04	0.05

DD%: Percentage of detected deadlocks, Er%: Percentage of total energy consumed to resolve detected deadlocks

Table 3.5: Percentage of detected deadlocked packets and percentage of total energy consumed to resolve them under *shuffle* traffic scenario with different deadlock detection schemes

to compare the fully adaptive routing algorithm augmented by the TC-network with different deadlock avoidance techniques. The deadlock avoidance techniques may produce non-adaptive (deterministic) or partially adaptive routing algorithms. The following routing algorithms that adopt deadlock avoidance are chosen to compare with the deadlock recovery: 1) *XY* routing [52] as a very popular deterministic routing algorithm; 2) *West-First* [65] very popular partially adaptive routing; and 3) *Odd-Even* [47] as it produces better adaptivness among all the turn-based routing algorithms [47]. Figure 3.24 shows the average network delay versus the network throughput for the four above-mentioned routing algorithms. Clearly, the fully-adaptive routing algorithm outperforms all the other routing algorithms and this is due to the maximal adaptivness provided by its routing function. The *XY* routing performs the worst, since there is always one path between each pair of communication irrelevant to the network status (congested or not).

Figure 3.25 shows the energy required to transmit 1 Mbyte of data through the network with different injection rate using the four different routing algorithms. Clearly the TC-network approach consumes less energy to send the 1 Mbyte of data for all the PIR conducted in the experiment.

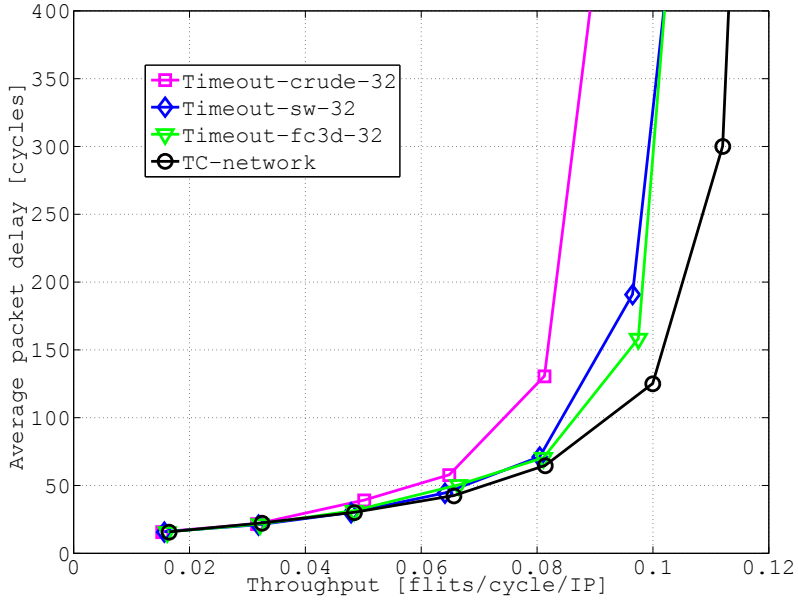


Figure 3.22: Evaluation of deadlock recovery system with *shuffle* traffic distribution and different deadlock detection techniques.

3.7 SUMMARY AND CONCLUSIONS

Interconnection networks with adaptive routing are susceptible to deadlock, which could lead to performance degradation or system failure. This chapter studies deadlock detection and recovery in networks-on-chip, as opposed to deadlock avoidance. Detecting deadlocks at run-time is challenging because of their highly distributed characteristics. This work presents a deadlock detection method that utilizes run-time transitive closure (TC) computation to discover the existence of deadlock-equivalence sets, which imply loops of requests in NoCs. This detection scheme guarantees the discovery of all true-deadlocks without false alarms, in contrast with state-of-the-art approximation and heuristic approaches. A distributed TC-network architecture, which couples with the NoC architecture, is also presented to realize the detection mechanism efficiently. Detailed hardware realization architectures and schematics are also discussed.

The proposed method is rigorously evaluated using a cycle-accurate simulator and synthesis tools to demonstrate the effectiveness of the TC-network based detection method compared to the timeout mechanism. Experimental results confirm the merits and the effectiveness of the proposed method. It drastically outperforms timing-based deadlock detection mechanisms by eliminating false detections and, thus, reduces energy wastage in retransmission for various traffic scenarios, including real-world application. The new method eliminates the need for any kind of time out mechanism and delivers true deadlock

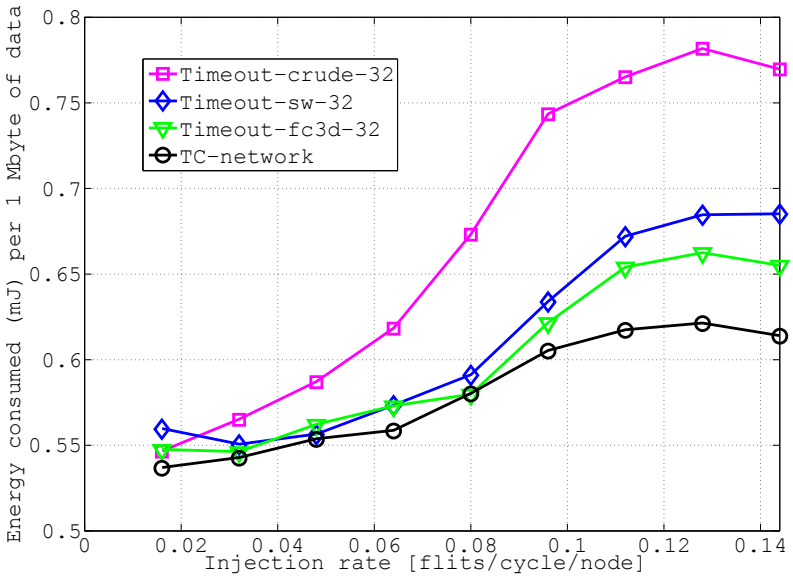


Figure 3.23: Energy required to drain 1 Mbyte of data in the deadlock recovery system for *shuffle* traffic and using different deadlock detection techniques.

detections independent of the network load and message lengths, rather than approximating with congestion estimation, as in the existing methods. It has been observed that timing based methods may produce two orders of magnitude more deadlock alarms than the TC-network method. Moreover, the hardware overhead for the TC-network has been examined. The implementations presented in this chapter demonstrate that the hardware overhead of TC-networks is insignificant. It is found that TC-unit hardware consumes less than 0.75% of the router area and dissipates less than 0.45% of the router power. These figures give an area saving of more than 2.0% and power saving of more than 0.4% when compared to the crude timeout circuit implementation with 10-bit threshold counter.

A new progressive deadlock recovery protocol is also proposed. The recovery technique augmented the well-known end-to-end transmission protocol. The full deadlock recovery system (detection + recovery) is evaluated with TC-network and the timing-based detection methods. This system then compared to popular deadlock avoidance techniques in terms of performance and energy consumed for communication. Next chapter will present the results of implementing TC-network for real-time deadlock detection in a 3D networks-on-chip.

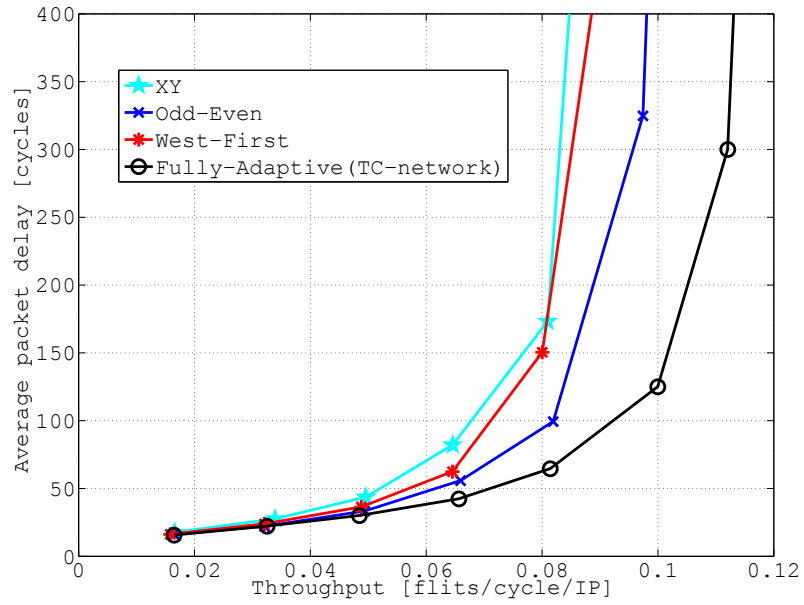


Figure 3.24: Performance evaluation of deadlock recovery and deadlock avoidance systems with *shuffle* traffic distribution.

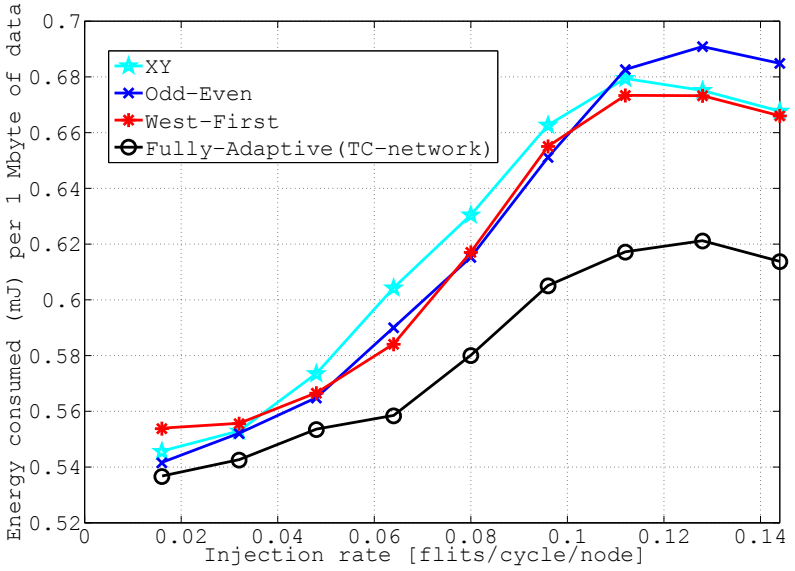


Figure 3.25: Energy required to drain 1 Mbyte of data in deadlock recovery and deadlock avoidance systems for *shuffle* traffic.

4

3D-NOCS BASED TSV INTEGRATION

4.1 INTRODUCTION

Over the past decades, shrinking transistor feature size has been the core aspect of maintaining Moore's Law [125]. The IC functionality to keep with Moore's observation needs to be doubled approximately every two years. However, global interconnects became a performance hindrance to current and future VLSI systems design because the scaling favours logics more than interconnects [3]. This brought the attention of researchers to other alternatives to maintain the trend. One important way forward is to explore the design of multi-level IC using 3D VLSI [135, 63, 106]. The possibility of 3D integration provides a new dimension to exploit novel geometric integration of silicon dies [58]. A variety of vertical cross-die interconnection techniques are being developed. In particular, the TSV [144, 20, 95] of 3D-IC connecting multiple die/wafer layers in a single chip provides opportunities to increase the integration capacity and also reduces the lengths of global interconnects. Moreover, 3D-IC chip is capable of integrating different technological scales and/or different technological compartments, such as CMOS logics, Memory, analogue sensors and even micro-electro-mechanical systems (MEMS), by implementing them over multiple die layers.

Combining NoCs with the emerging TSV-based 3D integration technology, 3D-NoCs, can produce high performance and high density SoC and CMP designs which will sustain Moore's Law. As 3D-NoCs are a relatively recent methodology, thorough investigations are required in order to evaluate the potential benefits. This chapter investigates the merits of adapting the 3D-NoC design for future SoCs and CMPs by comparing it with the 2D-NoC design. The comparisons are focused upon three aspects: the average shortest path between any source/destination pairs; network throughput and network saturation; and the frequency of deadlock occurrences. The first and the second aspect can be conceived as a measure for possible performance enhancement, while the third aspect is a useful measure for the purpose of judging whether adopting deadlock detection methodology in designing such a system can yield a better performance compared to deadlock avoidance methodology. In addition, this chapter presents a 3D-IC fabricated prototype chip and the associated testing results. A TC-network, proposed in Chapter 3, prototype is fabricated using a die-stacked 3-layer 3D architecture using TSV and 150nm CMOS technology. The main contributions of this chapter are as follows:

- Investigating the advantages of implementing future on-chip systems using TSVs-based 3D integration;
- Investigating the effect of adding an additional dimension to the 3D-NoCs in terms of deadlock formation rate;
- Introducing the design of TC-network architecture and its implementation, using three-layer 3D-IC. The vertical inter-unit communication is achieved by means of TSV. The three-layer chip prototype is fabricated using 150nm CMOS technology.
- Presenting the test results and findings of the fabricated prototype chip.

The design, fabrication, verification and test of the 3D-IC is accomplished as a part of collaboration project between Newcastle University and MIT Lincoln Lab [5]. The latter contribution is primarily in the layout, fabrication and packaging of the chip. Part of the 3D chip design test results are presented in [15, 116]. The rest of the chapter is organized as follows: Section 4.2 illustrates the advantages of using 3D-NoCs design in terms of performance and deadlock occurrences. Section 4.3 presents the prototype three-layer 3D chip and the testing apparatuses. Section 4.4 includes experimental results and discussion, while Section 4.5 summarizes and concludes this chapter.

4.2 ADVANTAGES OF 3D-NOCS OVER 2D-NOCS

This section investigates and sets comparative benchmarks between 2D-NoCs and 3D-NoCs in terms of average shortest path, network throughput, network saturation and deadlock occurrences. Mesh topology is chosen with five and seven ports router architecture for 2D and 3D-NoCs respectively, this being because NoCs interconnected as mesh are very popular for their regularity which suits chip layout [161, 157, 10]. The performance evaluation was carried out using a modified version of *Noxim* [62]. In particular, Noxim is modified to support 3D-NoCs architecture and routing (*XYZ* and *fully-adaptive* routing algorithms). Moreover *Noxim* modifications presented in Chapter 3 are also used to evaluate the deadlock occurrences. In the simulated NoCs, each input channel consists of four flit buffers. The results are captured after a warm up period of 10,000 clock cycles. The overall simulation time is set to 300,000 clock cycles.

4.2.1 *Average Shortest Path Computation*

The shortest path problem can be described as follows: Given a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ nodes and $m = |\mathcal{E}|$ edges, and a cost associated with each edge $n_i \rightarrow n_j \in \mathcal{E}$, which is denoted as

$C(n_i, n_j)$. The edge cost can be defined subject to different applications. The total cost of a path $p = \langle n_0, n_1, \ldots, n_k \rangle$ is the sum of the costs of its constituent edges: Cost(p)= $\sum_{i=1}^{k} C(n_{i-1}, n_i)$. The shortest path of $\mathcal{G}$ from n_i to n_j is then defined as any path p with a cost that is $min \sum_{i=1}^{k} C(n_{i-1}, n_i)$ for all constituent edges n_i. The computational procedure of all-pairs shortest-paths, using the Floyd-Warshall algorithm, is outlined in Algorithm 4.1 [49]. It has three nested loops containing a $O(n)$. In lines 2-14, it converts the directed graph ($\mathcal{G}$) to a Boolean Adjacency matrix (A) to reflect the existing paths and their costs in the graph. In lines 16-22, the all-pairs shortest-paths is computed. The core formula computed in the three nested loops can be seen in code line 19 which requires an addition operation and minimum operator.

Algorithm 4.1 Floyd-Warshall algorithm to compute all-pairs shortest-paths from a directed weighted graph $\mathcal{G}$

Definition: -

$C_{i,j}$ is the cost (weight) to go from edge i to edge j.

∞ is a big value $\in \mathcal{R}$.

Input: -

$\mathcal{V}$ is a set containing all vertices in $\mathcal{G}$.

$\mathcal{E}$ is a set containing all vertices in $\mathcal{G}$.

Output: - $S^{(n)}$ is an $n \times n$ matrix containing all-pairs shortest-paths.

```
1:  n = |V|
2:  for (i = 1 to n) do
3:      for (j = 1 to n) do
4:          if (i == j) then
5:              A_ij ← 0
6:          else
7:              if (e(i,j) ∈ E) then
8:                  A_ij ← C_i,j
9:              else
10:                 A_ij ← ∞
11:             end if
12:         end if
13:     end for
14: end for
15: S^(0) ← A
16: for (k = 1 to n) do
17:     for (i = 1 to n) do
18:         for (j = 1 to n) do
19:             S_ij^(k) ← min(S_ij^(k-1), S_ik^(k-1) + S_kj^(k-1))
20:         end for
21:     end for
22: end for
23: return S^(n)
```

The all-pairs shortest-paths can be extended to compute the average shortest-paths length in NoCs to compare 2D and 3D topologies. In network topology, the average shortest path length is a well-known concept which is used as a measure of the efficiency of data transport on a network. It can be defined as follows:

$$\overline{\chi} = \frac{1}{n \cdot (n-1)} \cdot \sum_{i,j,i \neq j}^{n} d(n_i, n_j), \tag{4.1}$$

where n is the number of vertices in $\mathcal{G}$ and $d(n_i, n_j)$ is the cost of the shortest path from node n_i to node n_j. Similar to Algorithm 4.1, the status of connectivity for all-pairs of nodes in the network can be determined by modifying the line 19 by replacing the $+$ operator with logical OR ($\vee$) and the $\min$ operator with logical AND ($\wedge$). Certainly the cost between any two nodes will become a binary value (1 for an existing path and 0 for a non-existing path).

To illustrate using a numerical example, assumes a network with 8 nodes interconnected in 2×4 2D-NoC (Figure 4.1) and $2 \times 2 \times 2$ 3D-NoC (Figure 4.2) with different link weights of "1" for connected and "0" for disconnected. A TC-network with TC-units can be used to find the status of connectivity between nodes. If node 1 is the reference node, then the optimal costs are $V(3) = V(5) = V(6) = V(7) = V(8) = 1$ and others are zeros for the 2D network. While for the 3D network, if node 1 is the reference node, then the optimal costs are $V(2) = V(3) = V(4) = V(8) = 1$, while others are zeros. This can also be extended to compute the average shortest path length. Therefore, instead of finding the connectivity as "0" or "1", the objective could be modified to find the shortest distance between two nodes, and then find the average of the shortest path lengths between all the connected nodes. Returning to the given example, for the 2D network the optimal costs are $V(3) = 1, V(5) = 2, V(6) = 3, V(7) = 3, V(8) = 4$ and the average shortest path length is 2.6. While for the 3D network the optimal costs are $V(2) = 1, V(3) = 1, V(4) = 2, V(8) = 3$ and the average shortest path length is 1.75, assuming node 1 is the reference node for both cases.

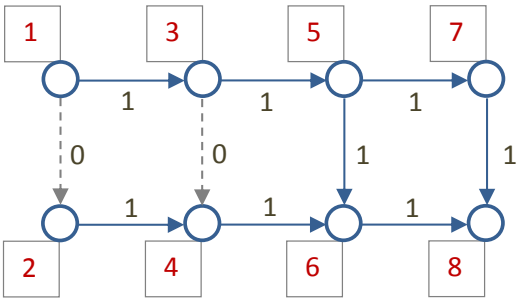


Figure 4.1: Eight nodes NoC interconnected in 2D

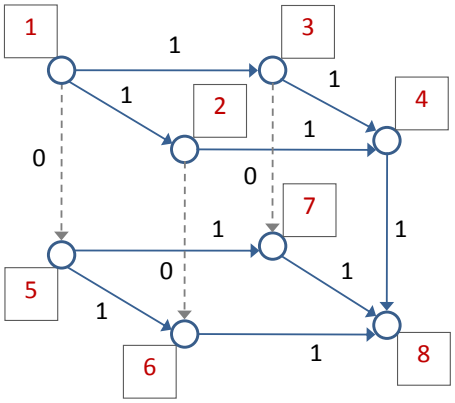


Figure 4.2: Eight nodes NoC interconnected in 3D

Different-sized networks are constructed and simulated in order to evaluate and compare the 2D and 3D average shortest path length. 2D-NoC sizes are 3×6, 4×8, 10×20, 10×30, ..., 10×120 and 3D-NoC sizes are $3 \times 3 \times 2$, $4 \times 4 \times 2$, $10 \times 10 \times 2$, $10 \times 10 \times 3$, ..., $10 \times 10 \times 12$. Therefore the number of nodes in the network varies from 18, 32, 200, 300,..., 1200. The network link weights are randomly generated in the simulation and then the average shortest path lengths ($\overline{\chi}$, see Eq. 4.1) for these networks are computed. A ratio of the 2D/3D yields is defined as β in order to compare 2D and 3D-NoCs ($\beta = \overline{\chi}_{2D}/\overline{\chi}_{3D}$).

The vertical links length (called as TSV technology factor) is also considered in the simulation. The 2D unit length is assumed 1 and the 3D via unit length is TSV, see Figure 4.3. Therefore, TSV is varied with TSV $= 0.5,\ 1,\ 1.5,\ 3$ and 5. The variation in TSV length is used to observe the impact of the TSV on the scale of the network. Figure 4.4 shows the average shortest path lengths versus grid sizes (number of nodes), together with the 2D/3D ratio β. β is close to 1 for small networks and becomes steady in the range of 2.5 to 2.9 until a grid size of around 800, then it rises rapidly for very large networks. An increasing TSV technology factor moves the 3D performance towards 2D (with a break-even when TSV is around 12 to 15). The plot clearly highlights the importance of the 3D technology especially for large networks even with TSV length assumed to be multiple of the 2D planar link length. In fact, many studies [111, 57] suggest that TSV link is much faster than normal 2D planar links which enable many proposals to serialize the vertical links to save area. The results presented in Figure 4.4 support these proposals as the 3D network can still produce a smaller average shortest path even if the TSV unit length is assumed multiple of the planar 2D links (assuming aggressive serialization is adopted).

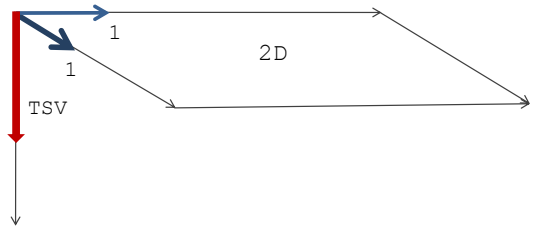


Figure 4.3: TSV unit length modelling in 3D-NoCs compared to planar 3D-NoCs links

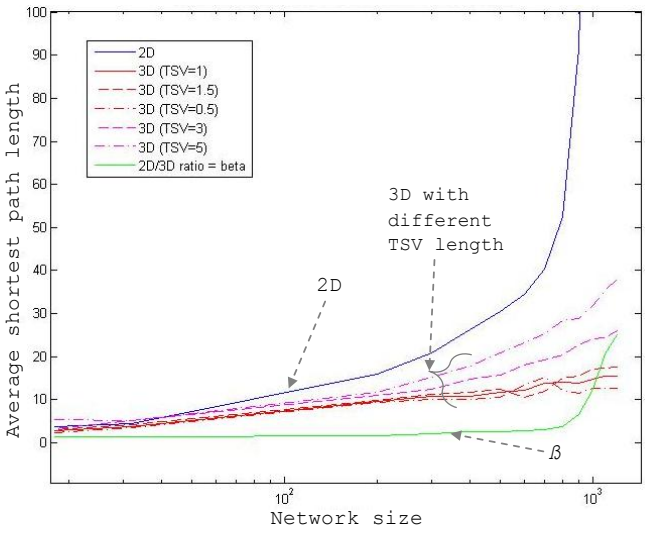


Figure 4.4: Average shortest path lengths versus grid size (log)

4.2.2 *Throughput and Network Saturation*

This section presents cycle accurate simulations for both 2D and 3D-NoCs which are used to investigate the possible performance enhancement that can be gained using 3D-NoCs. The evaluation is focused on the throughput and saturation point[1]. Figure 4.5 shows the throughput and saturation IR improvements for 128 nodes connected as 2D mesh NoC (16×8) and 3D mesh NoC ($8 \times 4 \times 4$). For this case, the throughput improvement of the 3D-NoC is 94% and the saturation load improvement is 100%. The result is captured based on a mesh NoC topology with 5 and 7 channels router architecture for the 2D and 3D-NoCs respectively and *XY* routing algorithm for 2D and *XYZ* routing for 3D-NoCs. The routers' vertical channels characteristics of the 3D-NoCs are assumed to be the same as the 2D-NoCs channels (TSV length equal 1, see Figure 4.3). Figure 4.6 shows the performance gain of the 3D-NoCs over 2D-NoCs for different grid sizes. The improvement increases rapidly when the number of nodes increases. The throughput and saturation IR improvements can be accounted for by the smaller

1 The saturation injection rate (IR) calculated at the point of the increase in applied load does not result in a linear increase in throughput [133].

hops count in 3D-NoCs when compared to 2D networks [63]. Thus using 3D-NoCs, a substantial performance improvement can be achieved over conventional 2D implementations.

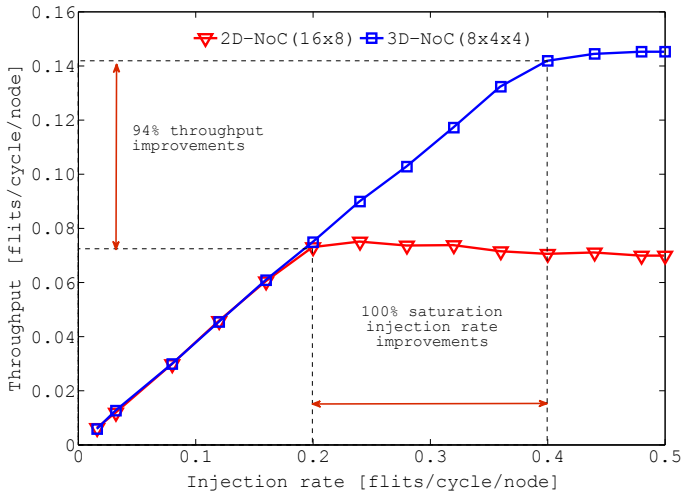


Figure 4.5: Throughput and saturation IR improvements of 3D-NoCs over 2D-NoCs for 128 nodes connected as mesh (16×8 and $8 \times 4 \times 4$).

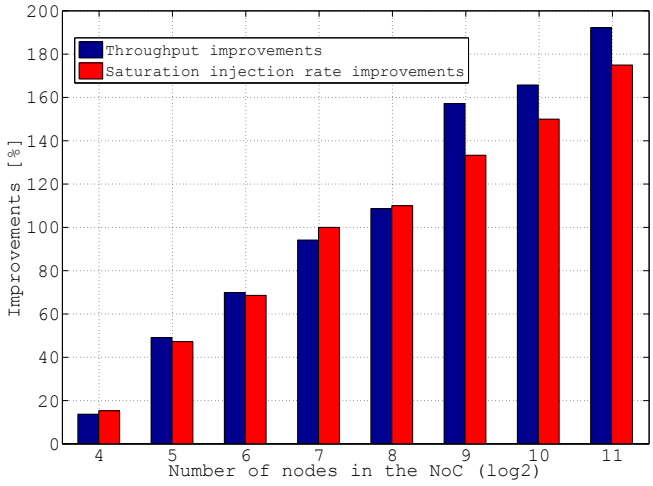


Figure 4.6: Throughput and saturation IR improvements of 3D-NoCs over 2D-NoCs for different network grid size.

4.2.3 *Deadlocks Formation Rate*

This section evaluates deadlocks formation rate on 2D and 3D-NoCs using TC-network, proposed in Chapter 3, and the state-of-the-art timing based method [22]. The simulated NoC has 64 nodes interconnected as 8×8 2D mesh and $4 \times 4 \times 4$ 3D mesh with *uniform* generated traffic. A

fully adaptive routing algorithm is used without using any deadlock avoidance techniques. Figure 4.7 shows the percentage of detected deadlocked packets to the total transmitted packets through the NoC with different IR.

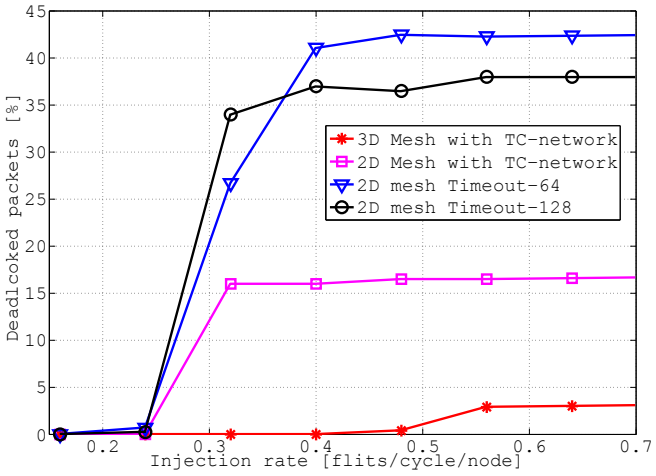


Figure 4.7: Percentage of detected deadlocks for 64 nodes NoC configured as 2D mesh and 3D mesh with fully adaptive routing algorithm.

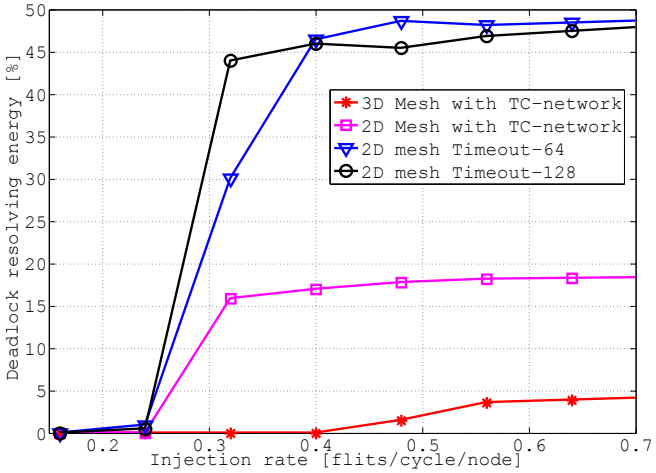


Figure 4.8: Percentage of energy consumed to resolve detected deadlocks to the total energy for NoC configured as 2D mesh and 3D mesh with fully adaptive routing algorithm.

In line with [22, 124, 140, 112], the formation rate of deadlocks increases with the increase of network load. It is clear from the figure that deadlock occurrences in the 3D-NoC are less probable than 2D-NoC as the TC-network detects less than 3% in the 3D-NoC compared to 16% detected using the TC-network in the 2D-NoC. This can be due to the extra spatial degree of freedom provided by the vertical channels

in 3D-NoC implementation. It is worth mentioning that TC-network method still detects fewer than the deadlocks detected using the time-out method [22], 47% and 48% for time-out values equal to 64 and 128 respectively. Figure 4.8 shows the percentage of energy used to resolve detected deadlocked packets to the total NoC energy. In general, the figure shows similar trends to Figure 4.7.

4.3 PROTOTYPING AND TESTING

This section presents the fabrication and testing results of the prototype 3D chip[2] which implements the 3D TC-network coupled to 3D-NoC. The 3D TC-network is used to compute at run-time the transitive closure of the NoC and identify the existence of a deadlock equivalence-set, as presented in Chapter 3. The chip layout and the pins map are shown in Figure 4.9. Table A.1 in Appendix A shows the complete list of I/O pins map.

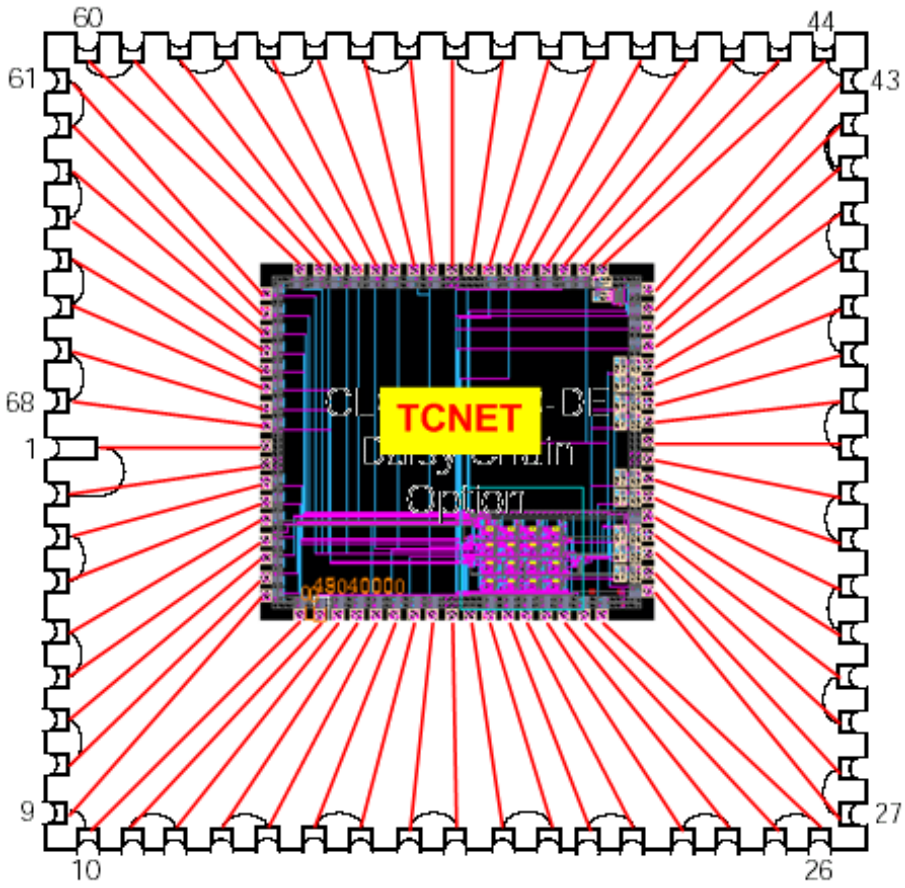


Figure 4.9: 3D stacked chip layout shows the input/output pins map

2 The chip was designed and implemented by Terrence Mak using MIT Lincoln Lab 3D foundry for fabrication. I contribute to the chip verification and testing and thus the emphasis in this section is on the testing part.

4.3.1 *3D TC-network coupled with 3D-NoC*

Figure 4.10 shows a 3D layout for the implemented TC-network coupled to 3D-NoC. It is a NoC of $4 \times 4 \times 3$ nodes and interconnected using mesh topology. The design is fabricated in a fully stacked 3-tier 3D TSV 150-nm CMOS technology through MIT Lincoln Lab [5]. Each tier (layer) consists of 4×4 tiles. The vertical inter-tiles communication is achieved by means of TSV. Each tile consists of a TC-unit, TC-interconnect and two sets of six bits register. The registers are used to mimic the channels' occupations and requisitions in each NoC router, i.e. the cardinal directions plus *Up* and *Down* directions. These internal registers can be addressed and loaded from off-chip with the test stimuli. The three tiers are manufactured in a $2mm \times 2mm$ and 150-nm CMOS technology and 3D Cu TSV.

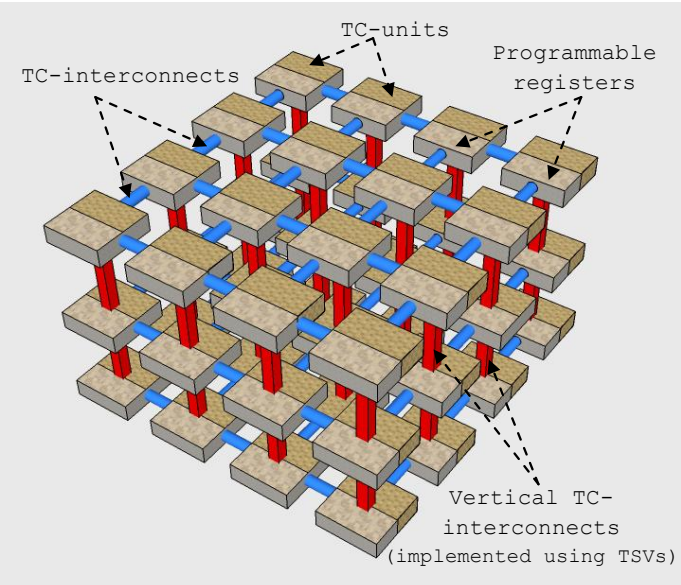


Figure 4.10: 3D layout shows the topology of the implemented chip.

4.3.2 *Three-tier Ring Oscillator*

To verify the feasibility of 3D circuits and the impact of TSV on digital signalling and circuit operation a ring oscillator (RO), as a standard digital circuit, is used. A 3D RO circuits with three inverting stages is implemented with p-type metal-oxide-semiconductor logic (PMOS) and n-type metal-oxide-semiconductor logic (NMOS) transistors width/length equal to $4\mu m/200nm$ and $2\mu m/200nm$ respectively. The inverter chain consists in total of 43 inverters and crossing 4 TSV as illustrated in Figure 4.11. The first tier contains 17 inverters and 13 inverters in each of the remaining two tiers. The number of 43 inverters is selected to allow the inverter chain delay enough time for the rising and falling time of the oscillator. According to [139]

$2NT_p \gg T_{falling} + T_{rising}$ where N is the number of inverters and T_p is the propagation delay of a single inverter. Thus N should be large enough to satisfy the equation.

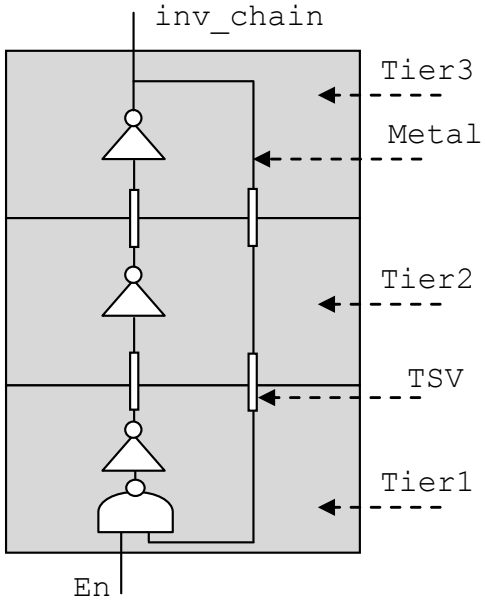


Figure 4.11: Schematic of 3D ring oscillator implemented on-chip

4.3.3 *Testing the 3D-IC chip*

The following apparatuses are used to test the fabricated chip:

1. FPGA board, Spartan-3A DSP [11] from Xilinx, as reprogrammable test environment for the chip.
2. A special socket and a printed circuit board (PCB) are used to mount the chip on top of the FPGA board using the 50 pins SAM interface. The PCB is designed to contain a programmable voltage level shifter to convert the SAM interface signals to the chip voltage level and vice versa (See Figure 4.12).
3. High performance oscilloscope and variable precise DC power supply.
4. High performance digital signal analyzer

Test patterns are prepared in Verilog and synthesised and downloaded to the FPGA board to check the functional operation of each tier in the chip separately and through tiers using the TSV connection. First, the static functional test was successfully carried out when the chip produces a stable output and draws a reasonable current (15.5mA at a VDD of 1.5V). Then a set of dynamical tests are implemented. These

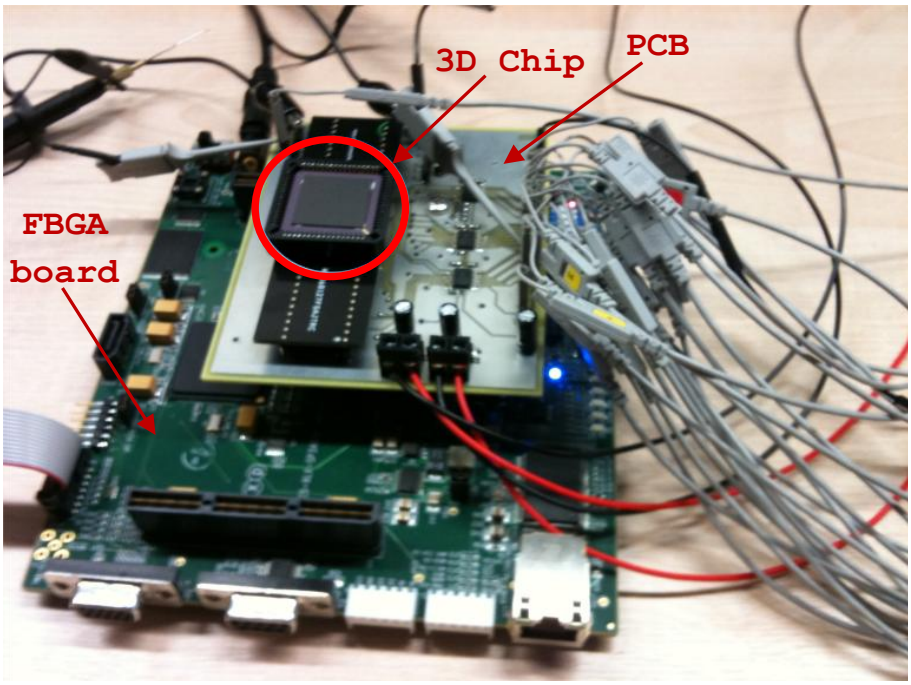


Figure 4.12: Snapshot showing the setup to test the chip. The chip is mounted in a special socket on top of a PCB then interfaced to the FPGA board using a SAM interface.

dynamical tests can be described as follows: after power on, first the chip is switched to initialization mode which resets all the internal registers. Subsequently, it starts configuring different size and type of deadlock loops by creating certain channels occupation and requisitions dependencies. The size of the deadlock equivalence-set will depend on how many nodes are participating in the loop, while the type is used to differentiate the DES in a single tier (planar deadlock) or through tiers (3D deadlocks), see Figure 4.13. The selection of the size and the type is a function of 8 bits input port on the FPGA board. Finally, the TC-network is activated by selecting the source node to check if it is part of a DES.

The internal registers that represent channels occupations and requisitions of each router in the NoC can be addressed using 4-bits addressing signals $(\text{RowEnable}\langle 1:0\rangle, \text{ColEnable}\langle 1:0\rangle)$ plus the layer enable signals $(\text{LayerEN}\langle 2:0\rangle)$. The register data $(\text{RegBus}\langle 0:5\rangle)$ are written with respect to the FPGA clock, 25.175 MHz, which is fed to the chip. A proper delay is applied to the FPGA clock to ensure that the arrival of the clock will be after the settling of the registers address and data signals. After configuring, deliberately, different size and type of DES, the TC-network computation can be initiated by selecting a node in the NoC to check if it is part of a DES. In principle, this represents the transitive closure computation of a single source (selected node) and multi-destination (the reset of the nodes). Source node selection is achieved by means of addressing signals

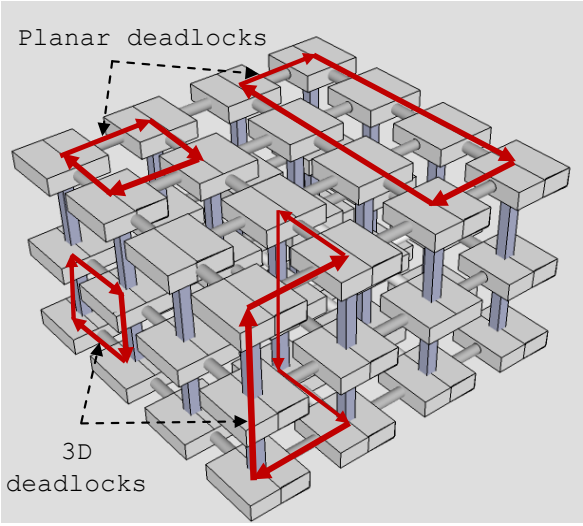


Figure 4.13: Examples of planar deadlock and 3D deadlock configurations with different sizes in the 3-tiers chip.

$(\text{RowisSource}\langle 1:0\rangle, \text{ColisSource}\langle 1:0\rangle)$ and layer source enable signals $(\text{IsSourceLayerEN}\langle 2:0\rangle)$. After configuring the 3 tiers' internal registers the TC-network output $(\text{LxRoOUT}\langle 1:0\rangle, \text{LxCoOUT}\langle 1:0\rangle,$ LxCoRoOUT_isZero, where x represent the layer number) is sampled and read through the FPGA SAM interface, digital signal analyzer and the oscilloscope. Figure 4.14 shows the input output signals of the 3D chip with 12 nodes DES configured in the first layer of the chip and detected by the TC-network. The blue signal is the triggering signal to compute the transitive closure of the network and the red signal shows the detection of a self reflexive loop (DES). The delay between the raise times of the two signals is 8.12 ns.

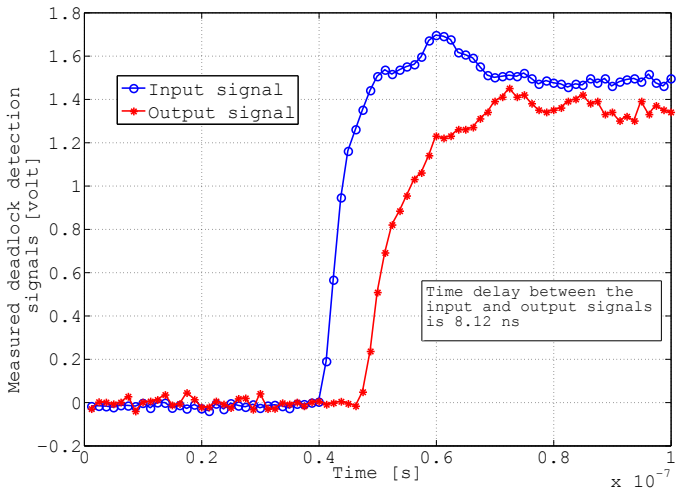


Figure 4.14: The start of TC computation and the deadlock detection signals

4.4 RESULTS AND DISCUSSION

4.4.1 *Ring Oscillator*

This section present the test results of the 3D RO with 43 stages of inverters connected together across three different tiers. Figure 4.15 shows the measured output of the 3D RO operation at VDD equal 1.5V. It shows a signal oscillating at 448.91MHz. The plot demonstrates that the three-tier ring oscillator chip is a successful 3D circuit. Figure 4.16 demonstrates the RO operation with different values of the power supply (from 1.5V to 0.4V). The figure illustrates the simulation results (frequency and period) with the actual measured result from the 3D-IC. Here, because the wiring length is very short and capacitance induced by 3D via is very small, the simulation neglects the TSVs delay. By comparing the actual measured frequency (448.2 MHz, T=2.227ns) with the simulated one (473.6MHz, T=2.112ns) at VDD of 1.5 volts, and considering only the delay induced by TSV vias (totally 4 in the ring oscillator), it is possible to obtain an estimation of the delay on each TSV as $(2227 - 2112)/2/4 = 14.375$ps.

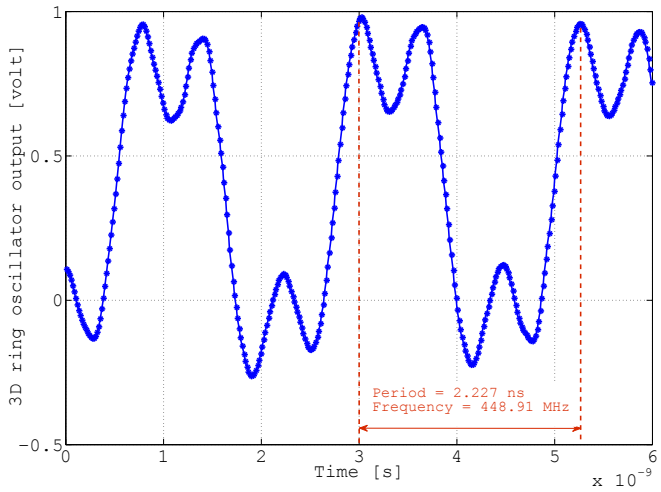


Figure 4.15: Ring oscillator output captured showing an oscillation of 448.91 MHz

4.4.2 *TC-network for Deadlock Detection*

Different-sized planar deadlock loops have been configured in tier 1 and tier 2 of the chip. The deadlock detection signals for these loops have been measured and plotted in Figure 4.17. The graph shows a logical trend in the detection time to detect the deadlock when the size of deadlock increases, i.e. covers more nodes, and the detection time increases as well. However, the delay figures for tier 2 are bigger

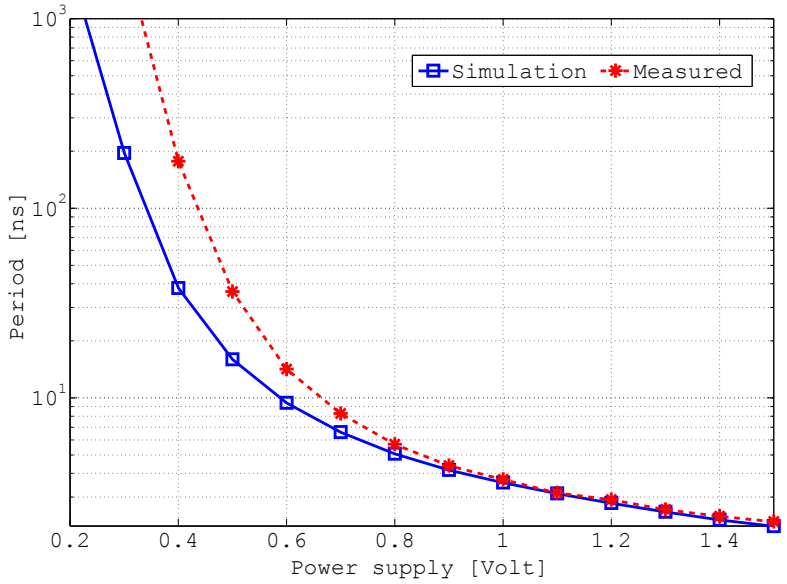
than tier 1 and this could be due to different interconnect routing for the two layers. Appendix A shows the individual oscilloscope capture of the presented test.

Different deadlock loops have been configured in order to compare the delay of the TC-network in detecting deadlocks in 2D and 3D. The detection delay is measured with different power supply voltages and plotted in Figure 4.18. The figure shows a slightly larger delay for detecting deadlock in 2D and thus suggests the planar interconnect delay is comparable to the vertical interconnect delay (TSV delay). Here, it should also be emphasized that the TC-network is a pure combinational logic and its computation speed is proportional with the power supply voltage supplied to the chip.

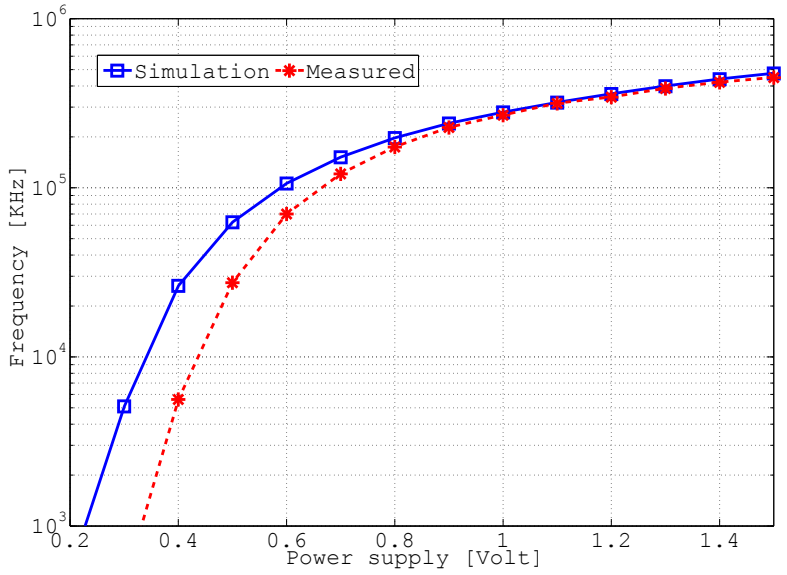
4.5 SUMMARY AND CONCLUSIONS

The first part of this chapter demonstrated the effectiveness of using 3D-NoCs compared to 3D-NoCs using different design measures for different network sizes. The average shortest paths length, network performance (throughput and saturation injection rate) and deadlock formation rate all show a considerable improvement when adopting 2D-NoCs compared to 3D-NoCs. In this study, the average shortest paths length for 3D-NoCs was found to be 2x to 3x smaller than a 2D version of NoCs with the same number of nodes. The 3D-NoCs design continues to show improvements even when the vertical network links are assumed multiple of the planar 2D links. This is to quantify the efficiency of mass transport of 2D-NoCs compared to 3D-NoCs. Also, performance evaluation of different sizes of 2D and 3D-NoCs revealed that 3D-NoCs can greatly improve the network throughput and the saturation load. The results show that around 100% throughput and saturation IR improvements are achieved with NoC size 128 interconnected as $8 \times 4 \times 4$ 3D mesh compared to 16×8 2D mesh and this improvement increases when the network size increases. Moreover, this chapter shows that deadlock is less probable in 3D-NoC due to the extra spatial degree of freedom provided by the vertical channels.

The second part of this chapter presented the 3D TC-network chip prototype and fabrication results. The prototype consisted of three tiers combined in a single chip and interconnected using TSV technology. Prototype testing results demonstrated the TC-network is effective and detects deadlock rapidly in a NoC. TC-network deadlock detection time was investigated using deadlock loops of various lengths. A set of comparisons between the delay time to detect planar deadlocks and the 3D deadlocks suggested that the vertical interconnect of the TC-network, using TSV, has a similar delay time to the normal planar interconnects. A 3D ring oscillator circuit realization in 3D using the TSV was presented and tested. This work provided encouraging results for future NoCs application using 3D embedded TC-network.



(a)



(b)

Figure 4.16: (a) Frequency (log); (b) period (log) of the 3D-IC ring oscillator operation with different power supply values

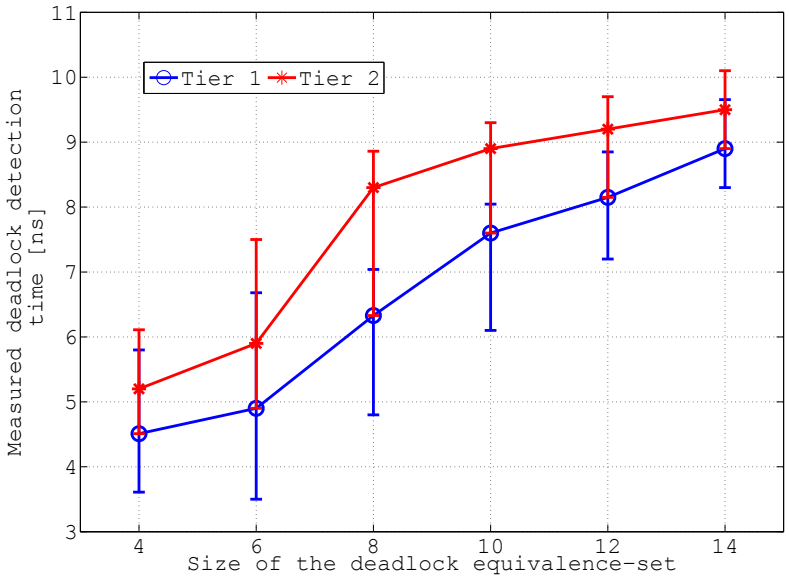


Figure 4.17: The time delay to detect different-sized deadlocks in tier 1 and 2 in the 3D-NoC chip (error bars represent the maximum and minimum measurements).

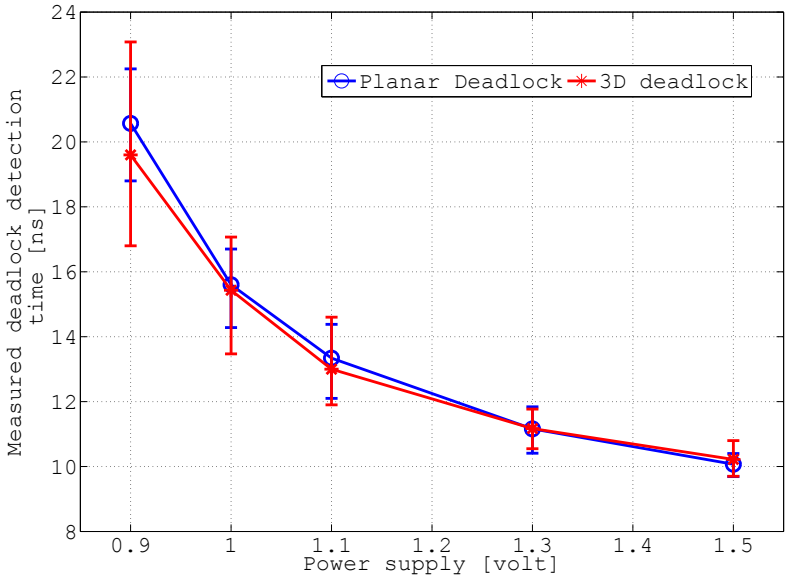


Figure 4.18: Deadlock detection delay with scaling down the power supply voltage (error bars represent the maximum and minimum measurements).

5 RUN-TIME THERMAL ADAPTATION IN 3D-NOCS

5.1 INTRODUCTION

The continuing advances in technology scaling over the past decades has led to very important improvements in the portability, size and performance of electronic products, but with some real challenges. Network-on-chip (NoC) [55, 33, 99] emerged as a new on-chip communication paradigm for SoC and CMP design to tackle many of these challenges. This new paradigm transmits packets between on-chip cores through a network of interconnected routers instead of using the classical point-to-point or bus-based interconnection.

Semiconductor manufacturing processes are approaching the physical limits of the semiconductors' materials. This brought the attention of researchers to other alternatives to continue deriving the benefits offered by scaling. One important way forward is to explore the design of multi-level IC using 3D VLSI [135, 63, 106]. In particular, the TSV [144, 20, 95] of 3D-IC connecting multiple die/wafer layers in a single chip provides opportunities to increase the integration capacity, reduce the global interconnects lengths, reduce the form factor and enable heterogeneous integration. However, increasing the integration capacity offered by technology scaling and 3D-IC integration led to several chip manufacturing thermal challenges: 1) scaling decreases the maximum junction temperature of transistors [152]; 2) 3D integration exacerbates the spatial temperature gradients over different chip stratum [41]; 3) both scaling and 3D integration increase the density of devices which means higher power dissipation and higher heat generation; 4) leading to designing cooling system based on a worst-case becomes infeasible for commercial products [85]. These thermal challenges are increasingly affecting the performance and the reliability of microelectronic products. It has been reported in [148] that over 50% of electronic product failures can be accounted for by thermal issues and hotspots formations. Furthermore, higher temperatures can cause slower devices, increasing leakage current, increase the interconnect delay due to increase in the resistivity and thus increase the possibility of timing closure failures in the communication fabric.

NoCs are repeatedly reported to consume a large portion of the dies power, *e.g.*, 40% of tile power dissipated by each router in MIT RAW chip [155, 8] and 28% of tile power goes to the router plus links in the Intel 80-tiles TeraFLOPS chip [157, 77]. It also has been reported in [28] that links and router power are comparable to computation unit power in each tile. Moreover, the power density of the router tends to be

higher than the rest of the tile which consequently makes it generate higher heat than the rest of the tile. This work argues that routing in NoCs based systems determine significant parts of the thermal gradient and by better control of routing, a better thermal distribution over the entire chip can be achieved. Hence, this chapter introduces an adaptive strategy to effectively diffuse heat throughout the 3D geometry. This strategy employs a dynamic programming network to optimize the direction of routing in NoCs. The main contributions of this chapter are as follows:

- Proposes a new adaptive routing strategy to effectively diffuse heat throughout the NoCs based CMPs via routing packets through the coolest and less utilized resources (routers and links).
- Introduces a distributed architecture, DP-network, to implement the proposed adaptive routing strategy in a 2D and 3D NoCs.
- Develops a tool, which is based on the accurate *HotSpot* thermal model and *SystemC* cycle accurate model, to simulate the thermal system and evaluate the proposed approach.
- Evaluates the proposed methodology through experimental studies and comparisons with the state-of-the-art adaptive routing algorithms using various traffic scenarios.

Part of the results of the proposed adaptive routing strategy is presented in [18, 17]. This chapter is organized as follows: Section 5.2, summarizes the related work for thermal modelling and mitigation of the ICs. Section 5.3 illustrates the methodology behind the proposed run-time thermal adaptation strategy. Section 5.4 presents the on-chip communication, power and thermal models adopted in this work. Section 5.6 summarizes and concludes this chapter.

5.2 RELATED WORK

The complex thermal behaviours prohibit the advancement of 3D VLSI systems. Thus thermal modelling of the chip (2D and 3D) and thermal mitigation become very important research topics and researchers [148, 152, 60, 85, 113, 167, 109, 45] have given a lot of attention to these topics. In terms of modelling, Skadron [152] and his research group made valuable efforts in modelling the thermal behaviour of Microprocessor chips. They started from a basic resistance-capacitance (RC) dynamic compact model to model the main heat transfer path with typical package settings. Their thermal model solver software is called *HotSpot*. The software evolved through several versions (from version 1.0 [82] up to version 5.02 [85]) with more accurate modelling of different heat transfer paths, materials, chip package and also enabled multi-layer

chip modelling. Currently, *HotSpot* does not support multiple materials within one layer in the 3D modelling, which could be necessary for modelling TSVs. However, in [50] the software is extended to enable modelling heterogeneous layers.

Thermal mitigation methods have also been studied by different researchers. For example, DVFS to avoid exceeding the emergency temperature are used in [60]. On-line task allocation to avoid exceeding the maximum temperature and hotspot formation is proposed in [113]. Others propose a run-time thermal management scheme for on-chip networks that uses mainly traffic throttling to avoid thermal emergencies [152, 148, 109]. All these techniques proposed for heat mitigation require sacrificing performance to avoid maximum temperature violation.

Exploring NoCs routing capabilities to better control the chip temperature and distribution has been partially explored by a recent work in [45], where a non-minimal 3D routing algorithm that routes packets to the layer closest to the heat sink then towards the destination has been proposed. It is basically a modified version of XYZ routing. This has been combined with a different type of traffic throttling to prevent reaching the emergency temperature. This work, however, introduces an adaptive strategy to effectively diffuse heat throughout the 3D geometry. The proposed approach would significantly diffuse the hotspots from a 3D geometry and the overall temperature can be significantly reduced and all the thermal mitigation from previous works can be used as integral part of this work, e.g., DVFS, traffic-throttling, etc.

5.3 METHODOLOGY FOR RUN-TIME THERMAL ADAPTATION

5.3.1 *Dynamic Programming*

Dynamic Programming (DP) is an optimization method which was first introduced by Richard Bellman [32] in the 1940s and developed since that through a large number of research papers [31, 64, 37, 24] and books [30, 66, 49, 38]. It has proved its effectiveness in a variety of practical problems in many disciplines [145, 136] where the main complex problem can be broken down into simpler subproblems and many of these subproblems are often the same. DP is based on a fairly simple idea which can be stated as: in general, to solve a given complex problem then one should combine the solutions of different parts of the problem (subproblems). In other words, dynamic programming (DP) refers to nesting smaller decision problems inside larger decisions. In contrast to *divide-and-conquer* method which shares the same definition of DP so far, the latter can yield a solution if the subproblems are not independent. DP method takes far less time than naive methods (if the repeating subproblems are exponentially large) as it aims to solve each subproblem only once. DP conveys the same concept in both

mathematical optimization and computer programming contexts. It is important to notice that not every decision problem can be broken down into subproblems. However, frequently decision problems that extend a number of points in time can be broken apart recursively. Bellman describes this as the *"Principle of Optimality"* or what was later called *"Bellman equation"* which is a necessary condition to apply DP. The Bellman equation states the value of a decision problem at a definite point in time in relation to reward from some initial choice plus the remaining value of the decision problem after making the initial choice. The decision-making problems can be expressed in a recursive form as:

$$V_i(t) = \max_{\forall k}\{R_{i,k}(t) + V_k(t)\}, \forall i, \tag{5.1}$$

where $V_i(t)$ is the expected reward of the i-th state and $R_{i,k}(t)$ is the reward of transition from state i to state k. The optimal solution to such an equation requires listing of all the possible states which can be computationally excessive because of the *"curse of dimensionality"* [154, 30].

5.3.2 *Coolest Path as Shortest Path*

The shortest path problem is a classical problem in graph theory. The shortest path between two nodes (vertices) in a weighted graph is a path such that the summation of the costs (weights) of its edges is minimal. Figure 5.1 shows the shortest path as a bold line between source node (S) and destination node (D). It also highlights that the shortest path problem exhibits the *optimal substructure* attribute which means combining the optimal solutions of the subproblems yields the solution to the original optimization problem. Any problem must show such a property in order for DP method to be applicable.

A range of run-time issues, such as dynamic routing, fault tolerance design and thermal aware-routing, can be formulated as the shortest path problem and resolved using a DP. Given a directed weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ with $n = |\mathcal{V}|$ nodes and $m = |\mathcal{A}|$ edges, a cost is associated with each edge $s \to d \in \mathcal{A}$, which is denoted as $T_{s,d}$. The edge cost can be defined subject to different applications and in this work is defined as the local router temperature. The nodes are the NoC's routers and the edges are the links. The total cost of a path $p = \langle r_0, r_1, \ldots, r_k \rangle$ is the sum of the costs of its constituent edges: Cost(p)= $\sum_{i=1}^{k} T_{i-1,i}$. The shortest (coolest) path of $\mathcal{G}$ from r_i to r_j is then defined as any path p with a cost (temperature) that is $\min \sum_{i=1}^{k} T_{i-1,i}$ for all constituent edges r_i.

The shortest path problem can be readily formulated and resolved as linear programming (LP) problem [75]. Suppose node (router) r_d is the destination node and the target is to compute the coolest path cost

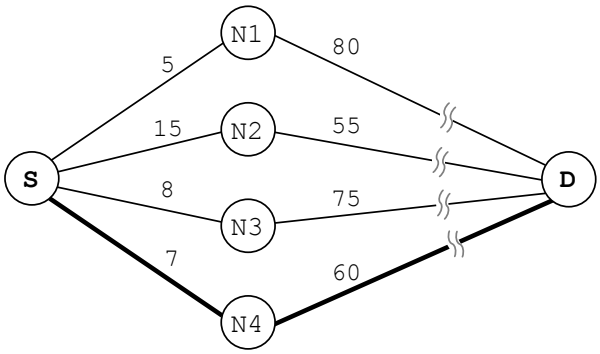


Figure 5.1: The shortest path in a weighted graph (bold line from S to D). It is clear this problem exhibits optimal substructure; a straight line indicates a single edge; a broken line indicates a shortest path between the two nodes it connects (other nodes on these paths are not shown).

$t(s,d), \forall s \in \mathcal{V}$, i.e multi-source single destination problem. To express it as LP, the constraint becomes $t(s,d) \leqslant t(u,d) + T_{s,u}$ to indicate that the cost of the coolest path from any node r_s to destination r_d is less than or equal to the coolest path from node r_u plus the cost of a direct path from node r_u to node r_s. Intuitively, the cost to go from the destination node r_d to itself is zero, thus $t(d,d) = 0$. Therefore, the following LP formulation can be obtained:

$$\begin{aligned} \text{Minimize} \quad & \sum_{\forall s \in \mathcal{V}} t(s,d) \\ \text{Subject to} \quad & t(s,d) \leqslant t(u,d) + T_{s,u}, \forall s,u \in \mathcal{V}, \\ & t(d,d) = 0. \end{aligned}$$

Otherwise, the problem can be declared in the form of the Bellman equation, which describes a recursive formula in step k. This formulation can lead to a simple parallel construction to accelerate the calculation. To find the cost of the coolest path from a source node (r_s) to a destination node (r_d) will require the *cost-to-go* function (or DP-value), which is the expected cost from r_s to r_d. This expected cost will be recursively updated based on previous estimates until it reaches the optimal value, i.e. the minimum cost in term of temperature which will represent the coolest path from r_s to r_d. This algorithm is known as *dynamic programming*. In its discrete form, the k-th symbol refers to the iteration number and the $*$ symbol refers to the optimal cost. The Bellman equation becomes,

$$V^{(k)}(s,d) = \min_{\forall u \in \mathcal{V}} \left\{ V^{(k-1)}(u,d) + T_{s,u} \right\} \tag{5.2}$$

where $V(d,d) = 0$. If the recursion is expanded from r_0 to r_k, the *cost-to-go* can be expressed as the total cost of the path from node r_0 to node r_k,

$$V_k^*(r_0, r_k) = \min_{\{r_0, r_1, \ldots, r_k\} \in P_{r_0, r_k}^k} \left\{ \sum_{i=1}^{k} T_{i-1,i} \right\} \tag{5.3}$$

where destination node $r_d = r_k$ and $P_{i,j}^k$ is the set of paths from r_i to r_j , all of which have k edges. The optimal decisions at each node r_i that direct to the coolest path can be obtained from the argument of the minimum operator at Eq. 5.2 as follows:

$$n_s = \arg \min_{\forall u \in \mathcal{V}} \{V^*(u,d) + T_{s,u}\} \tag{5.4}$$

where the optimal decision becomes $\mu(s,d)$.

5.3.3 *Dynamic Programming Network*

LP and DP approaches can yield the optimal solution for the coolest path optimization problem which was formulated as shortest path problem in the previous section. However, the DP approach presents an opportunity to greatly improve the computational speed by solving the problem using a parallel architecture. A DP-network architecture can be introduced to solve the Bellman recursive equation (Eq. 5.4) in a parallel computational platform. Such a network has a parallel architecture, and can be used to compute the coolest path solution through the simultaneous propagation of successive inferences.

From the literature, Lam and Tong [101] introduced a DP-network to solve a set of graph optimization problems with an asynchronous and continuous-time computational framework. This inference network has been shown to be inherently stable in all cases and robust with an arbitrarily fast convergence rate [101]. A parallel computational network for solving the dynamic routing problem for NoCs is also proposed in [117]. This chapter, however, proposes an adaptive strategy to effectively diffuse heat throughout the 3D chip geometry by selecting and optimizing the direction of data manoeuvre in a NoC. It also presents the implementation of this strategy using a distributed architecture, DP-network, which closely couples with the NoC infrastructure.

The DP-network is constructed by the interconnection of autonomous computational units. The structure of a *unit* U and *links* in a general inference network can be described by the following equations:

$$\eta_k(i,j) = \mu(i,k) + \mu(k,j) \tag{5.5}$$

$$\mu(i,j) = \min_{\forall u \in \mathcal{N}(i)} \eta_k(i,j) \tag{5.6}$$

where each *unit* represents a binary relation $U(i,j)$ between two objects and there are R sites, neighbouring units, to perform the inference action as defined in the site function (addition in this problem formulation). The value of the corresponding relation between i and j is then determined by resolving the conflict among all of the site outputs (minimum operator). Simply, $\eta_k(i,j)$ represents the site output at the k-th site and $\mu(i,j)$ stands for the unit $U(i,j)$ output. The units are linked to resemble the coolest path problem structure where each unit represents a node and a link represents an edge. A distributed network can readily be implemented using such realization.

5.3.4 *Coupling DP-network with 3D-NoC*

The DP network approach is illustrated in Figure 5.2. It consists of a network of distributed computational units and links between the units. The topology of the network resembles the defined graph topology, which is the communication structure of a NoC. At each node, there is a computation unit, which implements the DP coolest path calculations. The numerical solution of the unit will be propagated to the neighbour units via the neighbourhood interconnects. The DP network is tightly coupled with the NoC and each computational unit locally exchanges information with the corresponding NoC node. In particular, each unit receives the local router temperature and sends the optimal direction to route packets based on the coolest path to the target destinations. Figure 5.2 also shows temperature sensor(s) embedded in each tile which can be similar to any of the proposed on-chip sensors in [150, 74, 48].

5.3.5 *NoC Routing with DP-network*

Algorithm 5.1 presents the operations required for updating the routing directions with a DP-network. At each node unit, there are R inputs from the R neighbour nodes for the expected costs. The local router temperature, which is used as the cost in the DP-unit computation, could come from local embedded sensors in the chip or a core running the thermal model simulator. The output of the unit at node r_i is the updated expected cost $V(i,j)$ and is sent to all adjacent nodes. The main algorithm is outlined in line $1-7$. For each destination j and direction k, the expected cost will be computed and the minimum cost will be selected, as stated in line 5. The optimal direction for routing is selected and used to update the routing directions, as stated in line 6. Although the algorithm consists of two *for-loops*, this can be realized

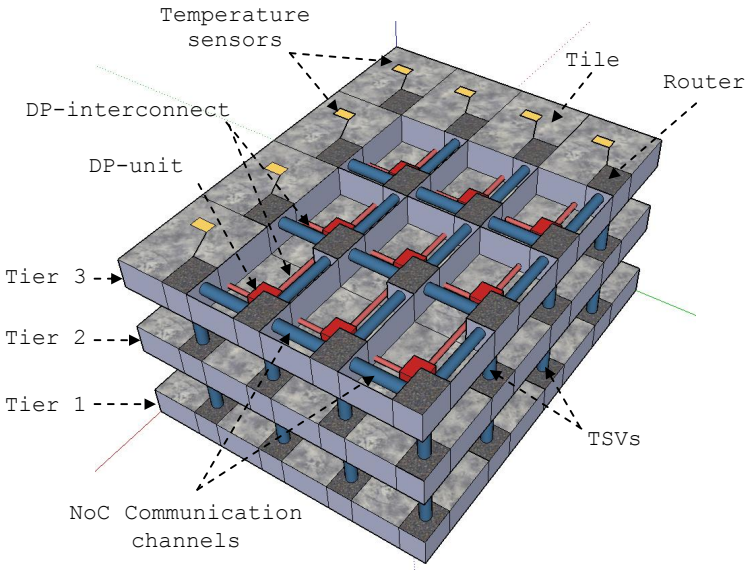


Figure 5.2: A DP-network coupled to a 3D mesh NoC, drawing not to scale

in hardware with a parallel architecture and the computational delay complexity can be reduced to linear. To avoid deadlock, the proposed strategy adopts the deadlock-free turn model [65]. The *Negative-first* turn model is used. It prohibits all turns to the *south-west* and *east-north* in a 2D mesh network plus the *south-down*, *east-down*, *up-west* and *up-north* turns in a 3D mesh network.

5.3.6 *DP-network Convergence Time and Complexity*

The delay of DP-network converges to an optimal routing solution depending on the network topology, which determines the delay of information propagates within the network, and the delay of each computational unit. It can be seen that each unit involves $\mathcal{O}(|\mathcal{A}|)$ additions and comparisons where $|\mathcal{A}|)$ is the number of edges. Note that the number of addition corresponds to the number of adjacent nodes and $|\mathcal{A}|)$ is an upper bound, which corresponds to the configuration of a fully connected network. Hence, the worst case solution time is $\mathcal{O}(k|\mathcal{A}|)$, where k is the number of iterations evaluated by each unit. In software computation, the k equals to the number of nodes in the network, thus $k = |V|$, which guarantees that all nodes have been updated [49]. However, in the hardware implementation with parallel execution, k is determined by the network structure and $\mathcal{A}$ additions can be executed in parallel. Each computational unit can simultaneously compute the new expected cost for all neighbouring nodes. Therefore, the solution time becomes the time for the updated value to be distributed to every other node and the computational complexity becomes $\mathcal{O}(1)$. The network convergence time is proportional

Algorithm 5.1 Pseudo code of the proposed dynamic thermal optimization routing using DP-network

Definitions:-

par: denotes parallel operations,
r_i: the current node (router),

Input:-

$V(i,j), i \in \mathcal{R}(i)$ where $\mathcal{R}(i)$ returns all neighbour nodes of r_i and $i = 1, 2, \ldots, R$
T_i router local temperature from thermal model or local sensor(s),

Output:-

$\mu(i,j)$ is the optimal direction from node i to j.
$V^*(i,j)$ is the optimal cost from node i to j.

1: **for all** i such that $r_i \in \mathcal{V}$ **do**
2: **par for all** k directions such that $k \in \mathcal{R}(i)$ **do**
3: $V'(i,k,j) = V(k,j) + T_i$
4: **end par for all**
5: $V^*(i,j) = min_{\forall k} V'(i,k,j)$
6: $\mu(i,j) = \arg\min_{\forall k} V'(i,k,j)$ {Update routing direction}
7: **end for all**

to the network diameter, which is the longest path in the network. To determine the minimum rate at which to clock each of the DP-units and still guarantee the convergence, the formula equation can yield that:

$$N_{dim} N_{dst} (t_{link} + t_{unit}) < t_{temp\_sampling}, \tag{5.7}$$

where N_{dim} is the NoC diameter, N_{dst} is the number of destinations (number of NoC nodes), t_{link} is the delay time of the DP-interconnect between each neighbour nodes, t_{unit} is the delay time of the DP-unit computation and finally $t_{temp\_sampling}$ is the time to sample the temperature of the individual router. To estimate t_{link}, the predictive technology model (PTM) [7] is used to evaluate the load wire resistance, inductance and capacitance which was found to be 448Ω, 1.9nH and 169fF per mm respectively for maximum interconnect length between routers equal 2mm and technology size of 65nm. The wire delay between DP-unit and DP-interconnect can be readily calculated based on the distributed resistance-inductance-capacitance (RLC) model [139]. For the temperature sampling time, it has been reported in [152] that a sampling period of 3 to 10µs gives excellent precision. Knowing NoC's size and diameter, t_{unit} can be easily computed to satisfy the formula above. Then the frequency to operate each DP-unit can be readily computed.

5.4 ON-CHIP COMMUNICATION, POWER AND THERMAL MODELS

5.4.1 *Communication Model*

The network of concern in this work is a collection of router nodes connected by channels. Each router is connected to a single IP core that can inject/consume packets via the router. An adaptive routing algorithm with minimal paths is used. A wormhole flow control technique [54] is employed which has been widely used in NoCs.

The performance evaluation was carried out using a modified version of *Noxim* [62]. In particular, the router architecture was modified by adding additional ports for communicating the DP values and introducing a routing-table updating scheme. Moreover, *Noxim* was extended to support 3D-NoCs topologies and routing. The simulator was also back annotated with all energy estimations extracted from the hardware synthesis tools and NoCs power simulator (see next section). The traffic patterns embedded in *Noxim* were used for the performance evaluation.

5.4.2 *Area and Power Model*

The router's area and consumed energy are evaluated using *Orion 2.0* [94] NoCs power simulator with router configuration and floorplan similar to the Intel 80-tiles chip [157] (see Figure 5.3), i.e., 3GHz operating frequency, 1V supply voltage, 39 bits channel width, etc. In general, the energy dissipated by a NoC is divided into the following categories: 1) Routing and arbitration energy which depends on the routing type; 2) Forwarding energy which is used in sending a flit to the next router or IP; 3) Receiving energy which is used in receiving a flit; 4) Clock energy which is the energy dissipated by the clock and 5) Waiting energy which is related to the time the header flit remains waiting until a successful routing takes place (leakage energy). The forwarding energy, which is the dominant energy on the router, can be divided into the following:

$$E_{forward} = E_{buffer} + E_{crossbar} + E_{link\_traversal} \tag{5.8}$$

where E denotes energy, $E_{forward}$ is the flit forward energy and E_{buffer} is accounted only for buffer reading energy, while the buffer writing energy is computed as part of the flit receiving energy.

5.4.3 *Thermal Model*

A typical modern chip package can be seen in Figure 5.4. The schematic shows several heat conductive layers between the PCB, on which the chip is usually installed, up to the ambient temperature represented

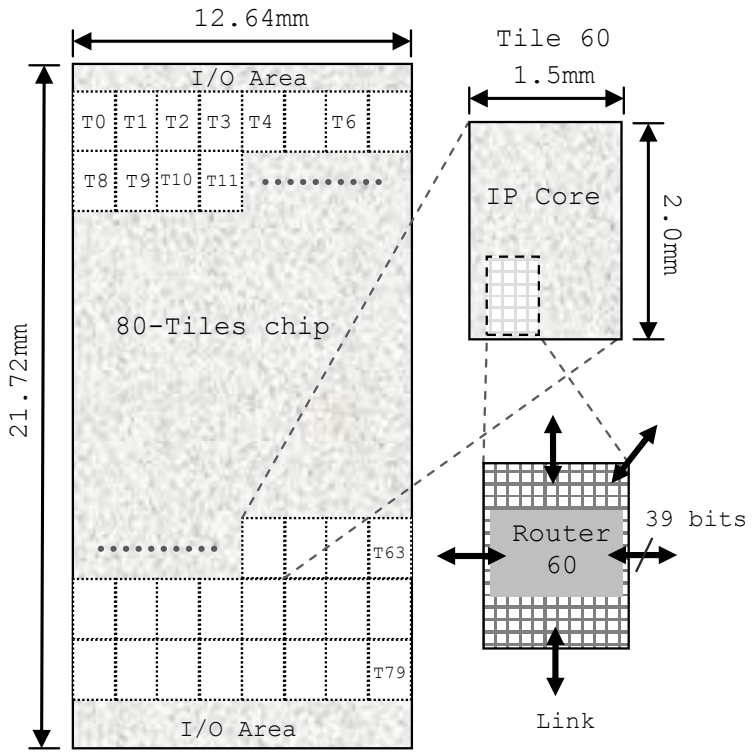


Figure 5.3: The floorplan of the Intel 80-tiles chip fabricated at 65nm technology.

by the air flow. The schematic is very popular in the academic community dealing with thermal aware architecture design, thermal package design and dynamic-thermal-management [148, 84, 85, 113]. The package is designed in such a way as to maximize the heat-flow from the active layer(s), in this case the die(s), to the ambient temperature and thus to remove the heat generated by the actual chip activity. This path represents the primary heat-flow in the package. The layers that constitute this path are in sequence; the die(s), thermal interface material, heat-spreader and heat-sink which is usually cooled by a fan. These layers are often made of aluminium, copper, or some other highly conductive material to maximize the heat-flow. Also there is a secondary heat-flow path from the die(s) to the PCB and the package is designed in a way to minimize the heat-flow through it in order to protect the board and the other devices installed on it from excessive heat accumulation. This path is from the die(s) to the PCB and it consists of the following: the die(s), C4 pads and underfill, IC carrier (usually ceramic), solder balls and the PCB. Moreover, there are other heat-flow paths which represent the lateral heat transfer between each layer in Figure 5.4 and the ambient. These heat transfer paths are included in the adopted thermal RC model [86].

The model is built on the basis of a well-known duality between thermal and electrical phenomena, as both are described by the same differential equations. The heat transfer through material is similar to a

current passing through a resistor and capacitor. The generated voltage difference (potential difference) is analogous to temperature difference (potential difference) and the electrical resistance that caused the difference is analogous to thermal resistance, while the delay in voltage change (analogous temperature change) is due to the existence of electrical capacitance (analogous thermal capacitance). As in electrical circuits, thermal capacitance is necessary for modelling transient heat transfer behaviours and both thermal resistance and capacitance will determine the time constant of the transient change (RC time constant).

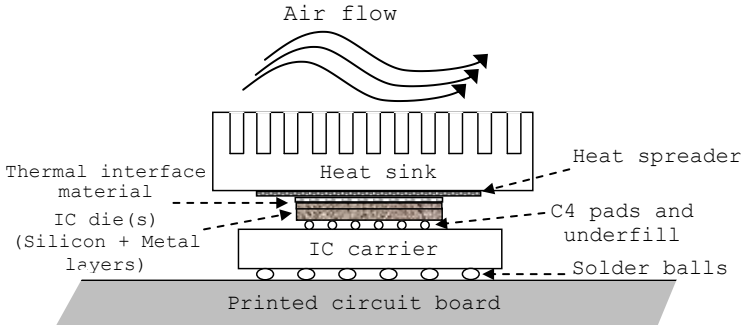


Figure 5.4: Side view of a typical IC package installed on a PCB showing the constituent layers [84].

Figure 5.5 shows a simplified RC-thermal model of the studied IC. Figure 5.5a demonstrates the primary and the secondary heat transfer paths to ambient which are represented by a potential difference circle. $R_{upper\_convection}$ and $R_{lower\_convection}$ represent the convection thermal resistors to the ambient, while $R_{upper\_conduction}$, $C_{upper\_conduction}$, $R_{lower\_conduction}$ and $C_{lower\_conduction}$ represent the combined thermal conduction resistors and capacitors of the primary and secondary heat transfer paths (these are the layers shown in Figure 5.4 which are combined here to avoid cluttering). Also, for the sake of clarity, the die is modelled in the figure as a single current source (node) flow through die thermal resistor (R_{die}) to the heat transfer paths. In fact, in the actual model, each die is divided into several nodes to represent the architectural blocks consisting of (i.e. router and core). Between each of the architecture blocks a lateral thermal resistance is modelled to reflect the heat diffusion between adjacent blocks within the die. Figure 5.5b shows an example of a die divided into two units and the lateral thermal resistance between them is labeled as $R_{lateral\_u1\&u2}$. Likewise, the chip may consist of several dies (3D IC) and these could be readily added to the described RC model with their corresponding heat-sources. Furthermore, to avoid cluttering, the lateral heat transfer is modelled in each layer of the chip presented in Figure 5.4, but not shown in the figures presented so far. Figure 5.5c illustrates the lateral heat transfer for the heat spreader layer. In addition to the thermal resistance of the heat spreader ($R_{spreader}$) to the thermal resistance of the heat-sink (R_{sink}) and eventually to the convection thermal resis-

tance (not shown in Figure 5.5c), it consists of four lateral thermal resistance ($R_{lateral\_spreader}$) connected directly to thermal convection resistance then to the ambient. This applies precisely for all the layers shown in Figure 5.4. The values of thermal resistance (R) and the thermal capacitance (C) can be computed from the following well-known equations:

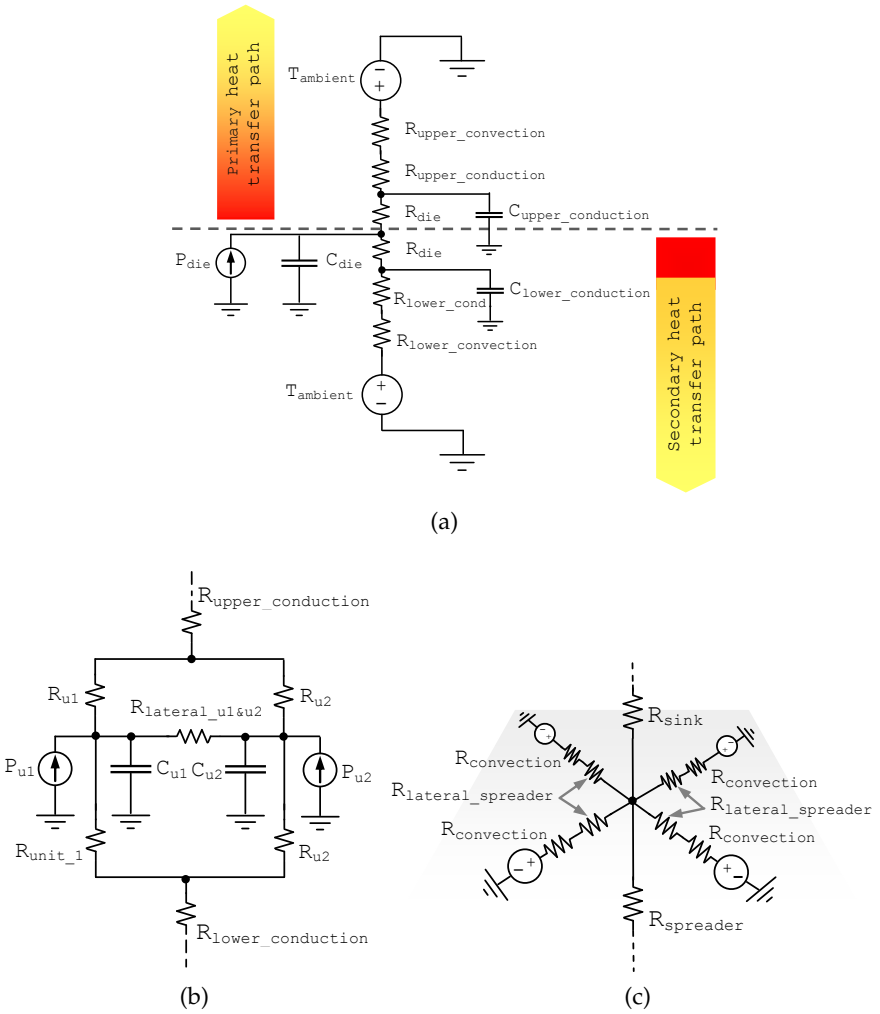


Figure 5.5: Dynamic compact model illustration: (a) RC model for Figure 5.4; (b) RC model for an example die with two architectural units (u1 & u2) and their connection to the upper and lower layers; (c) RC model for the heat-spreader layer, illustrating the four cardinal direction's lateral heat transfer paths.

$$R = \frac{t}{k \times A} \tag{5.9}$$

$$C = c \times t \times A \tag{5.10}$$

Where t is the thickness of material (measured in m), k is the thermal conductivity of the material per unit volume (measured in

$W/(m.K)$), A is the cross-sectional area (measured in m^2) and finally c is the thermal capacitance per unit volume (measured in $J/(K.m^3)$). A more detailed discussion regarding the thermal model derivation, calibration and its validation against a commercial finite element simulator can be found in the original HotSpot papers [152, 84, 85, 86]. Table B.1, in Appendix B, summarizes all the materials parameters used in this work.

5.4.4 *Tool Chain for Dynamic Thermal Optimization for 3D-NoCs*

NoCs communication, power and thermal models, described earlier, are integrated in an automated flow to work as a tool chain for dynamic thermal optimization for 3D on-chip network. Figure 5.6 illustrates the core modules (represented by circles) and the interaction among them (depicted as arrows), as well as the necessary configuration and initiation parameters required by each core. The technology, architectural, die and package parameters are used to configure the three modules. Moreover there is internal storage between core modules and these are the places where passing parameters and measurements take place. The flow can be summarized as below:

i) Initiation and configuration parameters are fed to the models.

ii) The power simulator computes the dynamic and static energy of routers and links and passes them to the NoCs cycle accurate simulator.

iii) The NoCs simulator is used to generate the power traces for the specified temperature sampling interval and pass the data through internal storage to the thermal simulator.

iv) The thermal simulator receives the power traces and, based on the chip floorplan and the package specification defined by the initiation files, it generates the steady state temperature distribution over the entire chip. The temperature measurements are sent back to the NoCs simulator.

v) The NoCs simulator updates the router temperature received from the thermal simulator and if the simulation time is not finished will go back to step iii.

Finally the cycle accurate NoCs performance results and temperature variations over the entire simulation time are saved to storage in the computer for later analysis and visualization.

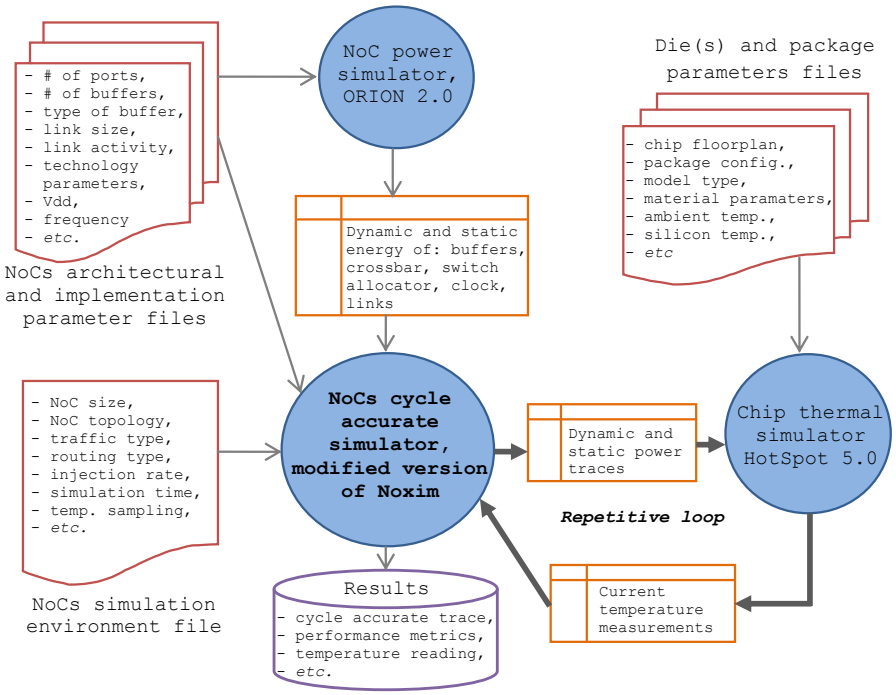


Figure 5.6: Automated flow of the proposed tool for dynamic thermal optimization for 3D-NoCs

5.5 RESULTS AND DISCUSSION

5.5.1 *Evaluation Methodology*

Several experiments were conducted on 2D and 3D NoCs based CMPs to evaluate the proposed method. The CMP configurations such as the floorplan, number of cores, technology features, etc. were adopted from the Intel 80-tiles fabricated chip [157]. The floorplan of the chip was arranged as an 8×10 2D matrix of tiles (core + router + links). In this work the main focus is to study the communication infrastructure and its effect on the overall chip thermal behavior. Thus, the centre of attention here is on the main NoC components, routers and links, while the rest of the tile components such as IPs, Clock, etc. are modelled in the floorplan of the chip and the power consumption is assumed to be 72% of the overall tile power, and as [157] shows, the communication power (NoC) is at 28% of the tile power.

Without any loss of generality, a NoC with mesh topology is chosen with five and seven ports router architecture for 2D and 3D CMP configurations respectively, because NoCs interconnected as mesh are very popular in the NoC's community. Each input channel consists of four flits buffer and one clock cycle is assumed for routing and transmission time across the crossbar. The *Negative-first* routing algorithm [65] is chosen for comparison with the proposed dynamic thermal optimization routing using DP-network. This is mainly for two reasons:

firstly, because *Negative-first* has a better degree of adaptiveness than the rest of turn-model [65] and secondly, because the proposed work adopted the *Negative-first* rules to avoid deadlock (see Section 5.3.5). All deterministic routing algorithms are excluded, like *XY* and *XYZ*, because it is not fair to compare the proposed adaptive routing with non-adaptive ones.

Regarding thermal model simulation, it is crucial that the thermal model be initialized with the right heat sink temperature because the time constant of the heat sink is significantly larger than the time constant of individual architecture blocks in each die(s) [152, 83, 86]. Therefore, as suggested by [83], the simulation is carried out in two phases. First, an estimate of the initial heat sink temperature is acquired using average power consumptions for each architectural block. Then this estimate is used to initialize the second phase run. In this way, more reliable results will be obtained. The overall simulation time is set to $3,000,000$ clock cycles (equivalent to $1ms$). The initial silicon temperature and the ambient temperature is assumed $25^{o}C$, similar to previous works [148].

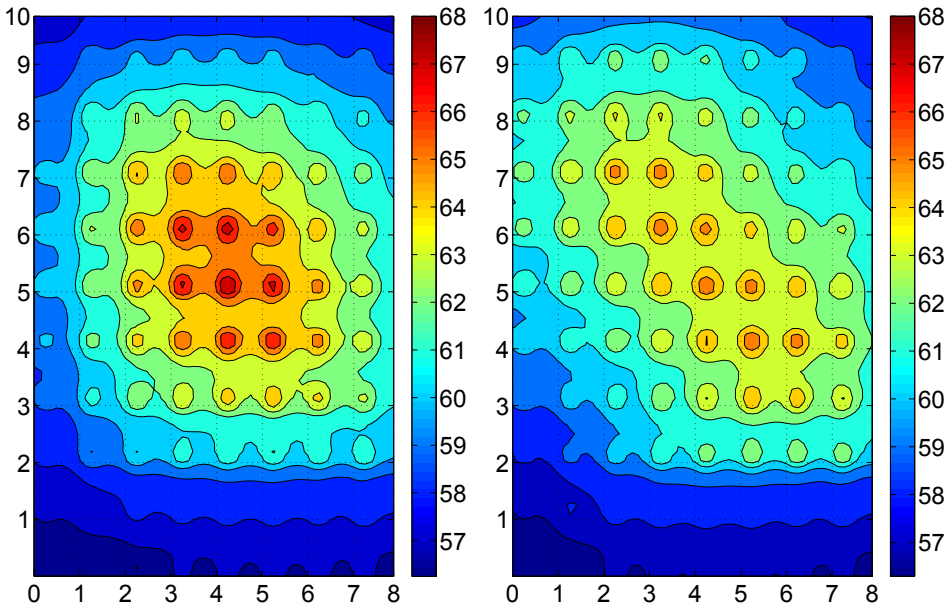
5.5.2 *Evaluation Results*

5.5.2.1 *Two-Dimensional NoCs*

An experiment with a transpose traffic scenario is performed for a 2D-NoC based CMP with *Negative-first* adaptive routing algorithm and with the DP-network routing. Figure 5.7 shows the steady state temperature distribution over the entire chip. Because of the traffic nature, the figure clearly shows a higher temperature along the diagonal. In Figure 5.7a, clearly the chip is subjected to several hotspots areas lying in the centre of the chip. Any of these hotspots can cause fatal damage to the chip or at least they will increase wear-out and aging rate. On the other hand, Figure 5.7b shows the temperature distribution of the chip when using the DP-network. The figure clearly shows that the DP-network successfully diffuses hotspots and distributes the heat more effectively over the entire activity area. In general, it reduces the maximum chip temperature by $2.3^{o}C$.

It can be observed that, in general, the last two rows of tiles are much cooler than the rest of the chip. The main reason for this is the nature of the traffic (transpose) which limits the activity to the upper 64 tiles (8×8). The last two rows will simply send and receive nothing and eventually they converge to ambient temperature, except the the lateral heat transfer effect caused by the upper neighbours' rows. The other three reasons, which also applies for all the boundary nodes, are: 1) the routers in these nodes have either 4 or 3 channels compared to the standard 5 channels router (not located in the boundary) and this makes the router consume less power and is therefore cooler; 2) the boundary nodes have a direct lateral convection heat transfer

to the ambient in addition to the normal conduction heat transfer through the heat-sink and 3) the routers in the boundary have less communication load.



(a) Negative-first (b) DP-network

Figure 5.7: Temperature distribution of the 80-tiles chip subjected to *transpose* traffic scenario

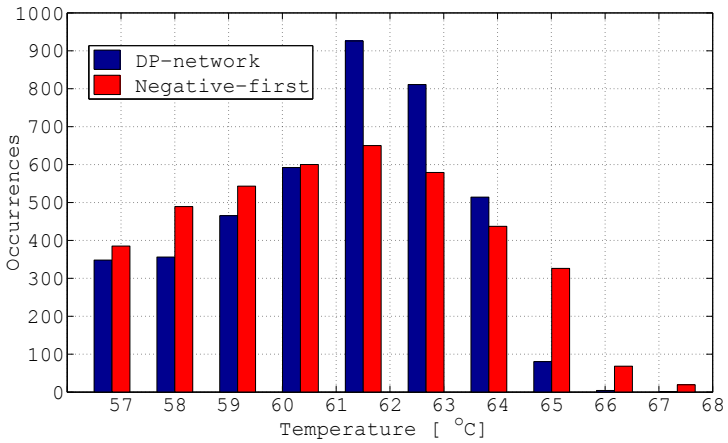


Figure 5.8: Temperature occurrences over the 80-tiles chip with *transpose* traffic scenario

The histogram in Figure 5.8 shows the steady state chip temperature occurrences with and without DP-network. The granularity of the grid model (thermal) used to evaluate the temperature distribution is 64×64 (i.e., 4096 thermal nodes). The figure clearly shows that DP-network effectively eliminates or reduces the higher temperature incidents (68°C, 67°C and 66°C), while it increases the nominal tem-

perature occurrences (63°C, 62°C and 61°C). Figure 5.9 shows the maximum steady state temperature distribution over the 80-tiles with and without utilizing the DP-network. If the last 16-tiles (from 64 to 80) were excluded for the reason explained earlier regarding the traffic nature, the figure clearly shows that DP-network effectively reduces the maximum temperature of the tiles located in the middle of the chip (tile-20 to tile-54) and increases the maximum temperature of the tiles in the boundary (0 to 19 and 55 to 63). This is because of the DP-network's run-time dynamic optimization capability to converge to the cooler paths in the network and thus guide the packets away from any thermal stress and hotspots. This produces better and more evenly distributed temperature over the entire chip and also improves the network performance (throughput and latency), as will be shown in the next section.

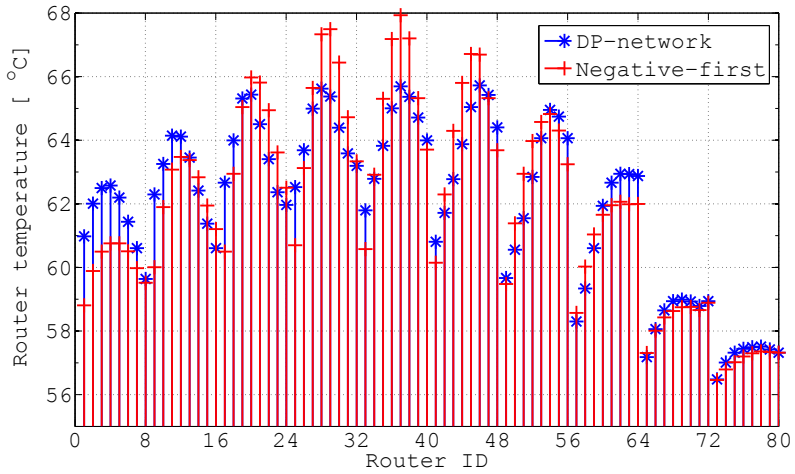
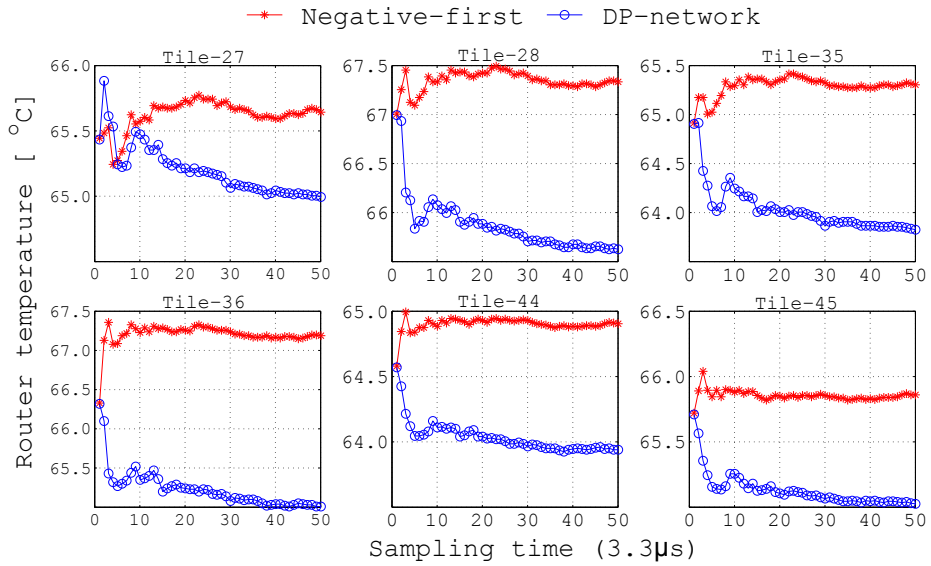
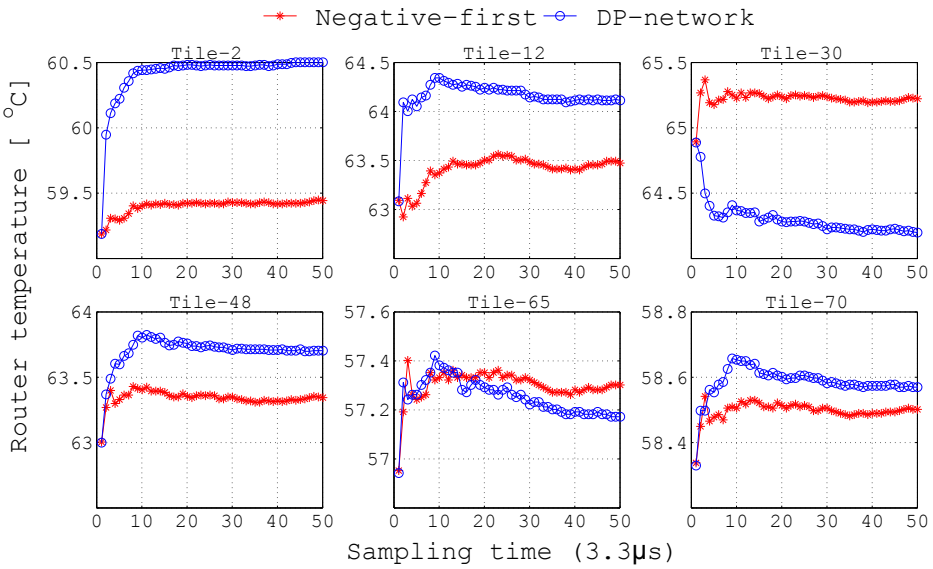


Figure 5.9: Maximum steady state temperature distribution of the 80-tiles chip subjected to *transpose* traffic scenario

Figure 5.10 illustrates the transient thermal behaviour of the tiles over the thermal sampling time (3.3µs). It shows 50 samples of a selection of tiles located in different places. Once again, the DP-network managed to reduce the temperature of the hotspot tiles by directing the traffic away from them. It also selected the cooler tiles to route the traffic through them and thus increased their temperature slightly. Both cases are illustrated in the sample of the selected tiles in the figure. The family of curves presents an interesting and very powerful feature of the DP-network. They show what can be called the "learning curves" of DP-network to converge to optimal paths in terms of temperature distribution. The figure includes only 50 samples because the curves show no dramatic changes after that.



(a) Sample of tiles located in the center of the chip



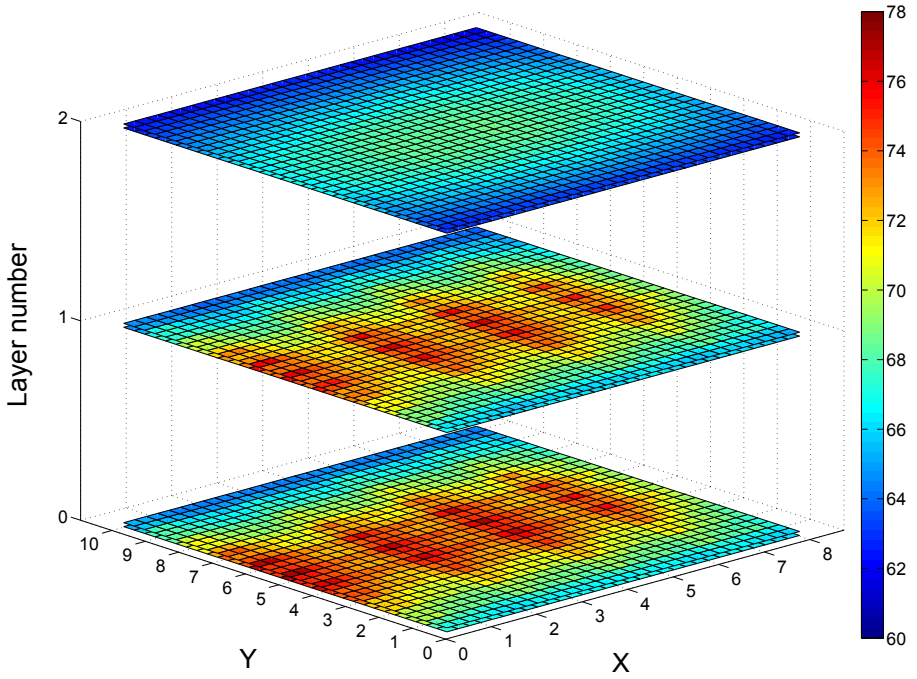
(b) Sample of tiles located near the boundaries of the chip

Figure 5.10: Temperature variation over time for a selection of tiles interconnected as 8×10 2D mesh NoC and subjected to *transpose* traffic

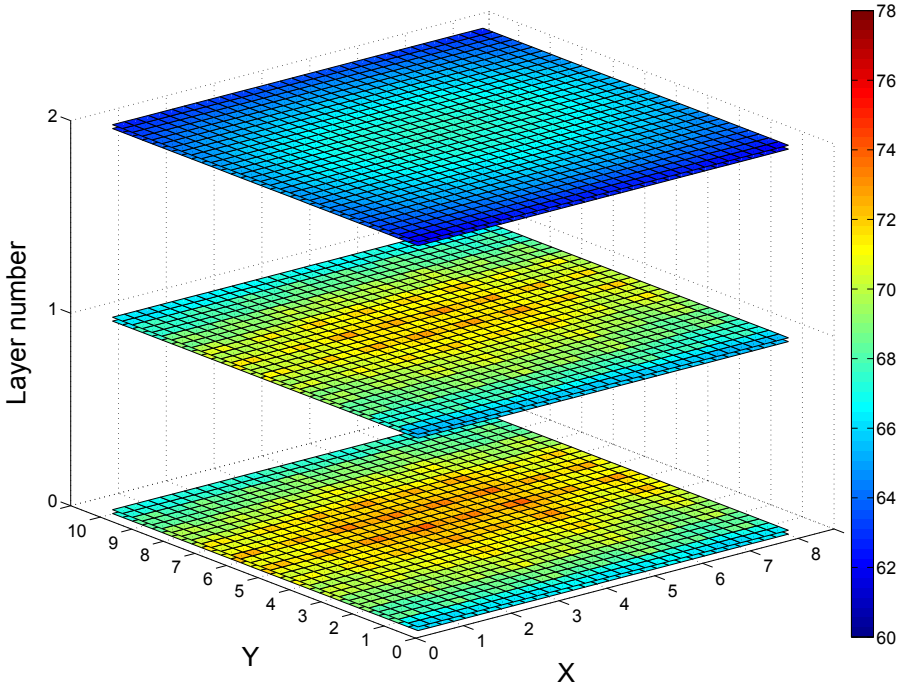
5.5.2.2 *Three-Dimensional NoCs*

For 3D chip the assumption that there are three dies similar to the Intel 80-tiles chip is made and they are integrated using TSV technology process. In such a scenario and because of the physical geometry of the 3D chip, the die-to-heatsink heat conduction efficiency is different for each layer. For example, the top layer in the chip on average has better heat-removal efficiency than the inner layers since it is closest to the heat-sink and to the ambient temperature. This could lead, in the inner layers, to high temperature hotspots and thus could cause permanent device damage. An experiment for the three layers chip is conducted subjected to *butterfly* traffic scenario and fully-adaptive routing algorithm with and without the proposed thermal adaptation strategy. Figure 5.11 shows the steady state temperature distribution of the hottest layer in the chip with and without the DP-network. Figure 5.11b illustrates the ability of DP-network to spread the heat generated from the chip activity over the entire chip area and thus diffuse hotspots that are clear in Figure 5.11a (without DP-network). The dynamic optimization ability of the DP-network leads the routers to use cooler paths to route packets. The DP-network successfully reduced the peak chip temperature by more than 3°C (from 77°C to less than 74°C).

The histogram in Figure 5.12 shows the temperature occurrences for the three layers with and without utilizing DP-network. Again the DP-network effectively eliminates higher temperature incidents (77°C, 76°C and 75°C) and reduces the incidents with (74°C and 73°C). However, it increases the nominal temperature occurrences dramatically (the middle of the figure). Moreover, the DP-network does not have very low temperature occurrences (such as 66°C) and it reduces the occurrences of 67°C considerably. All this data leads to a very important conclusion that the DP-network adaptive capabilities effectively diffuse heat throughout the 3D geometry by converging to cooler paths to route data in NoCs. This will be more evident if Figure 5.13 is investigated. It shows the maximum temperature of each tile in the three layers with and without using the DP-network. The DP-network managed to reduce the temperature of the tiles 16 to 64, while it increased the temperature of tiles 0 to 16 and 64 to 80 in each layer of the chip. The difference between the maximum and minimum temperatures is 11°C in the case of routing without the DP-network. This is reduced to 7°C when the DP-network is employed. Figure 5.13 also shows that DP-network successfully reduces the difference between the individual layers of the chip for better support of more layers. Figure 5.14 shows the transient temperature change of the above described experiment for a selection of tiles distributed over the three layers. It can be inferred that the DP-network over time learns about the cooler and hotter paths in the chip and thus routes packets accordingly.



(a) *Negative-first*



(b) DP-network

Figure 5.11: Temperature distribution of the simulated $8 \times 10 \times 3$ 3D NoC subjected to *butterfly* traffic scenario

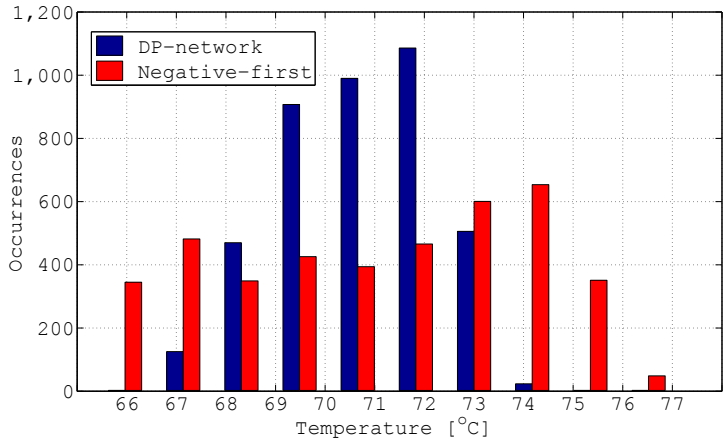
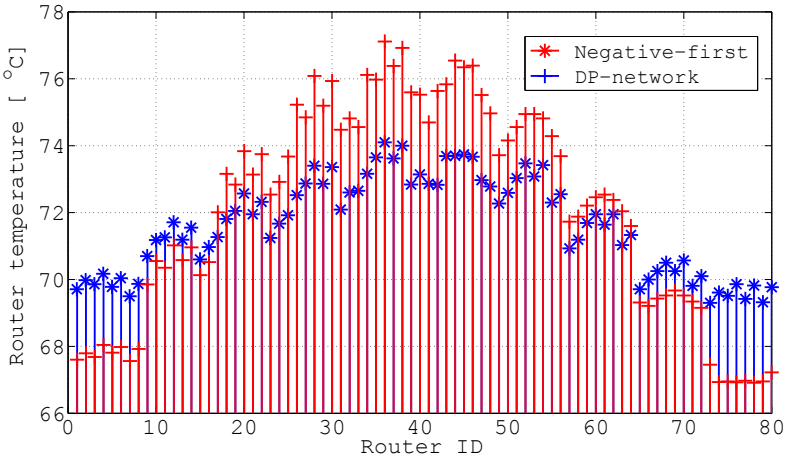


Figure 5.12: Temperature distribution over the 240-tiles chip with *butterfly* traffic scenario

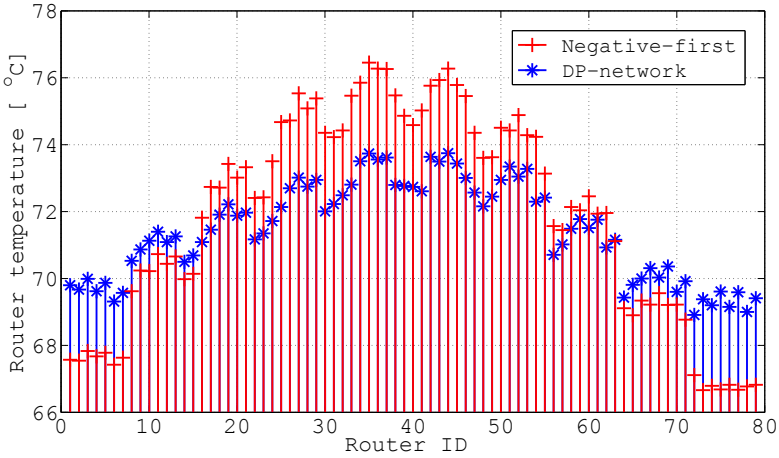
A new experiment is conducted to investigate the performance of the 3D-NoC. A *transpose* traffic is used with a different IR. Figure 5.15 shows the average delay plotted against throughput. It clearly shows the DP-network improving these two important network metrics compared to *Negative-first*. It improves the throughput by 20%. Figure 5.16 shows the minimum and maximum temperature of the hottest layer versus the IR. It clearly shows that the chip peak temperature is increased with the increase of the IR, which is natural as chip activity is increased. The gap between the maximum and the minimum temperatures also increases with the increase of the IR. However, using DP-network reduces the gap between the minimum and maximum temperatures. To quantify this improvement, the following formula can be defined:

$$DP_{temp\_improvement} = (1 - \frac{T'_{max} - T'_{min}}{T_{max} - T_{min}})100\%, \tag{5.11}$$

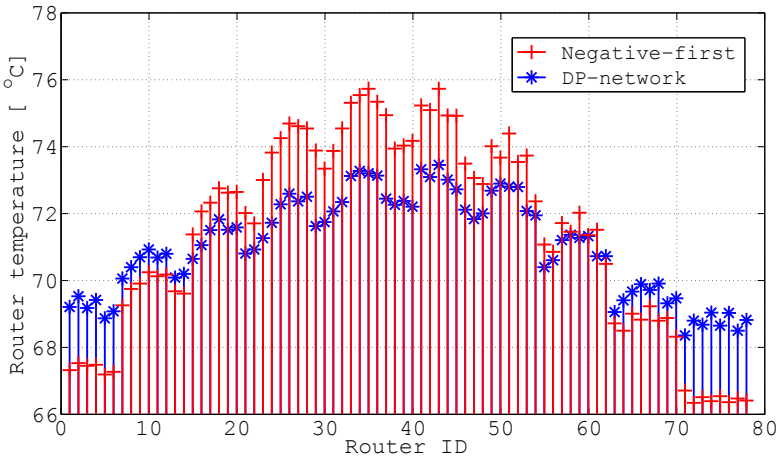
where $(T'_{max} - T'_{min})$ and $(T_{max} - T_{min})$ are the differences between the maximum and minimum temperatures with and without DP-network respectively. Applying Eq. 5.11 on the data presented in Figure 5.16 will yield 32% improvement of the DP-network at IR = 0.4. It means routing using DP-network reduces the maximum variation on the chip temperature by 32%. For the rest of the traffic scenarios the results are summarized in Table 5.1. It shows the performance metrics and the maximum and minimum temperatures of each layer with IR values not causing large throughput differences for fairness in comparison. It also shows the percentage of the DP-network improvement over the *Negative-first* routing for each of the fields computed using Eq. 5.11.



(a) Layer 0

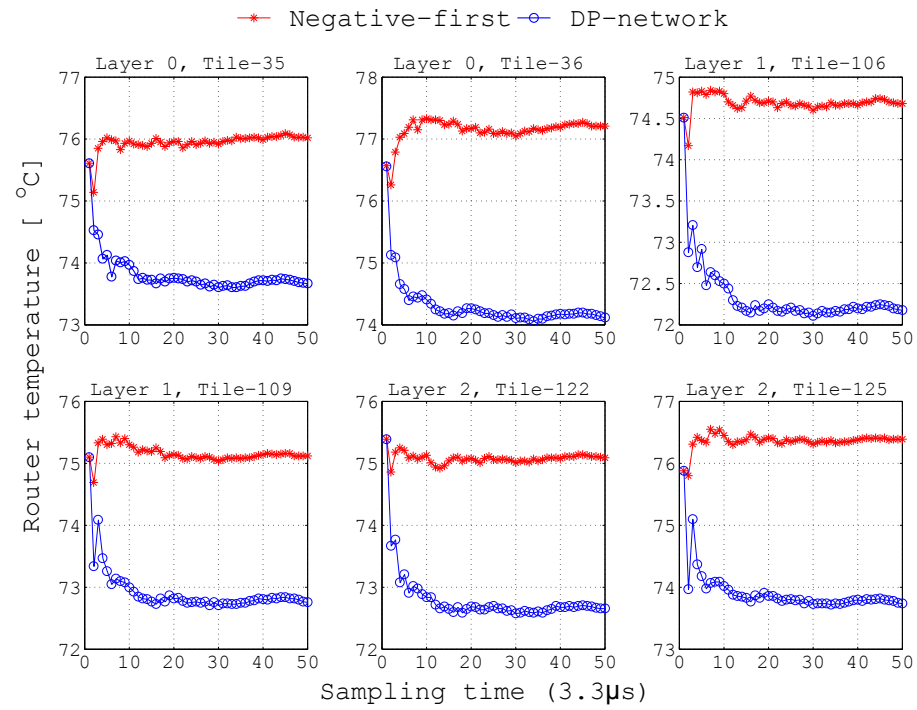


(b) Layer 1

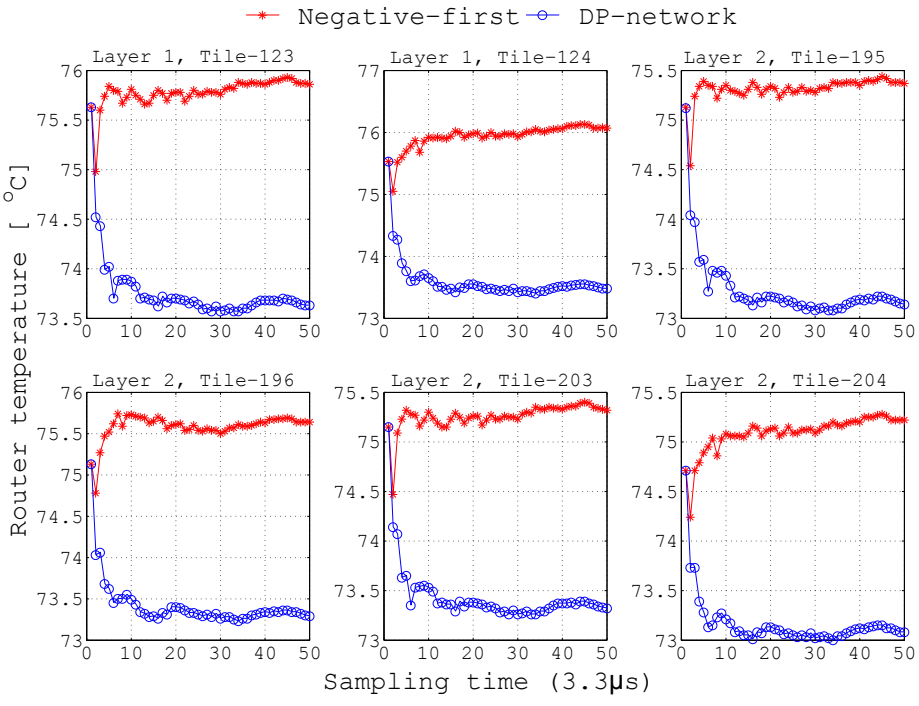


(c) Layer 2

Figure 5.13: Maximum temperature distribution over the 240-tiles chip



(a)



(b)

Figure 5.14: Temperature change over time for a selection of tiles under *butterfly* traffic.

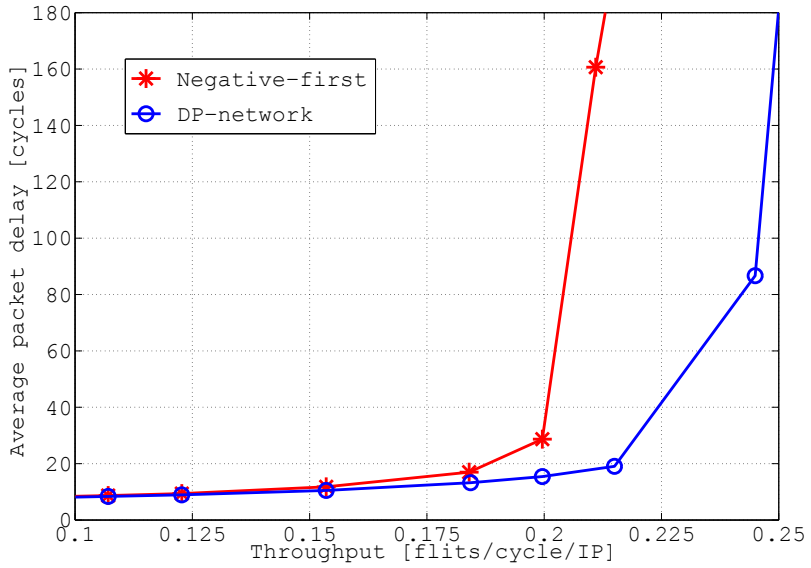


Figure 5.15: NoC performance (Average delay versus Throughput) for *transpose* traffic

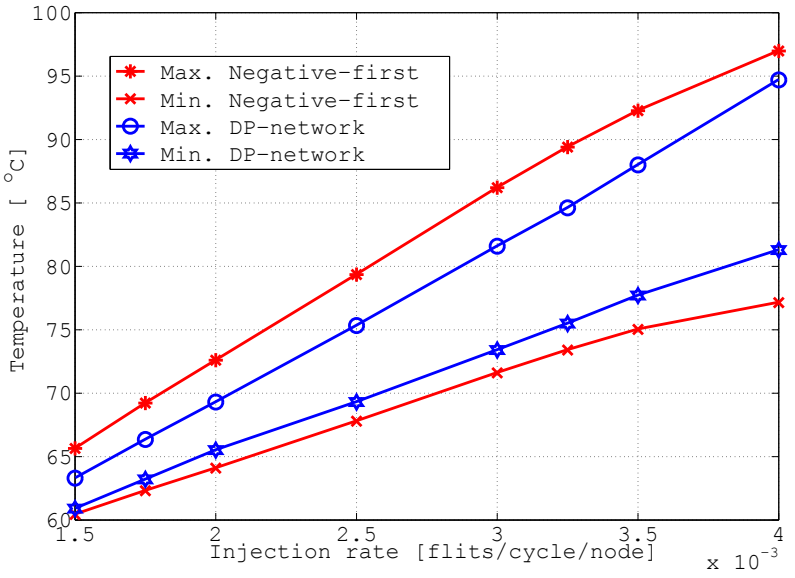


Figure 5.16: Maximum (max.) & minimum (min.) temperature for the hottest layer under *transpose* traffic.

Experiment Settings		Performance Measurements			Layer 0, Temp.		Layer 1, Temp.		Layer 2, Temp.	
Traffic + Routing	IR	Throughput	Avg. Delay	Max. Delay	Max.	Min.	Max.	Min.	Max.	Min.
Random + *Negative-first*	0.048	0.0539	36.17	344.98	81.25	66.69	80.54	66.57	79.54	66.38
Random + DP	0.048	0.0542	30.48	187.15	78.35	67.7	77.8	67.53	77.04	67.2
Improvement with DP		**0.5%**	**15.7%**	**45.8%**	**26.9%**		**26.5%**		**25.2%**	
Shuffle + *Negative-first*	0.048	0.0525	42.70	959.71	78.66	64.37	77.85	64.08	76.86	63.82
Shuffle + DP	0.048	0.0527	31.27	399.27	75.25	65.40	74.68	65.00	73.89	64.70
Improvement with DP		**0.3%**	**26.7%**	**58.4%**	**31.1%**		**29.7%**		**29.5%**	
Bitreversal + *Negative-first*	0.0288	0.0320	126.02	7683	68.8	58.07	67.36	57.80	66.39	57.70
Bitreversal + DP	0.0288	0.0323	34.69	607.0	67.00	58.90	66.20	58.60	65.30	58.30
Improvement with DP		**0.7%**	**72.7%**	**92.2%**	**24.5%**		**20.5%**		**19.4%**	

Table 5.1: Performance metrics, maximum and minimum steady state temperature measurements (in C^o) for different network traffic with and without DP-network

5.6 SUMMARY AND CONCLUSIONS

In networks-on-chip (NoCs) hotspot formation and higher thermal variations over different chip regions/layers can lead to timing closure failures in the communication fabric. With the introduction of the 3D VLSI technology, thermal variations will get worse. These complex thermal behaviours prohibit the advancement of 3D VLSI system. In particular, the high-density through-silicon-via-based 3D integration could lead to ultra-high temperature hot-spots and permanent silicon device damage.

This chapter introduced an adaptive strategy to effectively diffuse heat throughout the 3D geometry. This strategy employs a dynamic programming network to select and optimize the direction of data manoeuvre in a NoC. This led also to developing a tool, which is based on the accurate *HotSpot* thermal model and *SystemC* cycle accurate model, to simulate the thermal system and evaluate the proposed approach. The evaluation results demonstrated that the proposed approach can significantly diffuse the hotspots from a 3D geometry and maximum temperature can be reduced by 4°C. Given the same thermal constraints, the throughput performance of an adaptive NoC can also be improved by 20%.

This work opens up a new avenue to explore on-chip adaptability for future large-scale 3D integration. The methodology of using different cost functions for DP-network is one of the future directions of this work. Moreover, the work presented in this chapter can be integrated with any of the existing heat mitigation techniques to produce a run-time thermal management system (see Chapter 6).

6

RUN-TIME THERMAL MANAGEMENT FOR 3D-NOCS

6.1 INTRODUCTION

Relentless advances in the field of technology scaling over the past decades have resulted in significant improvements in the portability and performance of electronic products. Nevertheless, some challenges have emerged. A major issue is that the scaling favours logic more than interconnects. To mitigate this deficiency, VLSI designers adopted a communication centric methodology which embraced a new on-chip communication paradigm, namely Network-on-Chip (NoC) [33, 55]. On the other hand, recent technological advances in 3D-IC provide a promising platform for the realisation of NoC based multicore systems and rendering augmented performance and heterogeneous integration capabilities. In particular, the TSV [144] of 3D-IC connecting multiple die/wafer layers in a single chip provides opportunities to increase the integration capacity. Considering 3D-NoCs in the future, SoCs and CMPs will bring huge advantages, among which are shorter global interconnects lengths, less delay, better scalability and smaller form factor.

However, the increase in transistors' density and power density as a result of aggressive technology scaling [3], combined with the additional tightly coupled physical layers imposed by 3D integration, has led to chip thermal challenges. On the one hand, the higher power dissipation is converted to higher temperature generation on-chip. On the other hand, the 3D integration exacerbates the spatial temperature gradients over different chip strata as a result of having different heat conduction path lengths towards the heatsink. Thus, the generated high and uneven thermal stress on the silicon layers could affect the performance and the reliability of microelectronic products. In particular, this can cause slower devices, increased leakage current and interconnect delay due to higher resistivity. This will also lead to increasing the probability of devices' timing failures.

In NoC based systems, communication power budget takes up a significant proportion of the overall chip power consumption. The NoC workload contributes significantly to the power and results in significant heat generation in the system. For instance, it has been reported in [148] that the on-chip network dominates processors in heat generation for some communication centric applications in the MIT Raw chip. Hence, this study argues that by better utilizing the on-chip routing and communication resources, the thermal dissipation of

the whole chip can be better controlled. This is particularly important to 3D-NoC, in which a tighter and more aggressive thermal diffusion can be observed. It is, therefore, calling for a reliable and systematic strategy to optimize the communication pathways on 3D chips.

This chapter proposes an adaptive dynamic programming-based run-time thermal management (DPRTM) strategy for networks-on-chip. The strategy consists of two regenerative schemes, namely proactive and reactive control. The proactive control employs a coolest path-first (CPF) technique, similar to what has been proposed in Chapter 5, which aims to reduce the probability of experiencing thermal hotspots by effectively re-directing routing loads towards the coolest paths among the available pathways. This approach can effectively and homogeneously re-distribute the on-chip heat over the entire chip at multiple layers. The reactive control scheme is triggered when there is a significant rise in temperature to a point where it is above the thermal limit. In this case, a throttling technique is integrated into the management strategy to reduce power consumption and, thus, rapidly cool down the chip. Although throttling is effective in moderating heat dissipation, it is at a cost of NoC throughput and average delay. This might result in critical delay and traffic hotspots. Therefore, the proposed DPRTM strategy is equipped with a priority scheme, namely least throttled paths-first (LTPF), which redirects packets away from the throttled paths whenever possible to minimize the impact of throttling on performance, such as throughput and delay. The major contributions of this chapter are:

- Propose a new dynamic programming-based run-time thermal management (DPRTM) system, including reactive and proactive schemes, to effectively diffuse heat throughout NoC-based CMP via routing packets through the coolest paths, when the temperature does not exceed the chip's thermal limit. When the thermal limit is exceeded, throttling is employed to mitigate heat in the chip and DPRTM changes its course to avoid throttled paths and minimize the impact of throttling on chip performance.
- Introduce a distributed architecture based on dynamic programming, to implement the proposed adaptive routing strategy for both proactive and reactive phases.
- Expand the developed traffic and thermal co-simulation tool described in Chapter 5. This tool is used to evaluate the proposed thermal management technique.
- Evaluate the proposed methodology through experimental studies and comparisons with the state-of-the-art techniques for RTM in NoC using various traffic scenarios.
- Evaluate the hardware area and power overhead introduced by the proposed DPRTM.

This chapter is organized as follows. Section 6.2, summarizes the related work for thermal modelling and mitigation of the ICs. Section 5.3 illustrates the methodology behind the proposed run-time thermal adaptation strategy. Section 5.4 presents the on-chip communication, power and thermal models adopted in this work. Section 6.4 illustrates the proposed hardware realization and hardware synthesization result and Section 5.5 includes experimental results and discussion. Section 5.6 summarizes and concludes this chapter and outlines some possible directions for future developments.

6.2 RELATED WORK

Run-time thermal management is scarcely studied in NoCs based systems context. The work proposed in [148] suggests a RTM framework for 2D-NoC which is called *ThermalHerd*. This can be considered as one of the first works evaluating the run-time thermal impact on NoC. The framework consists of the following: temperature monitors; traffic monitoring counters; distributed throttling mechanism; reactive and proactive routing algorithms. The temperature monitors can be updated from either distributed thermal sensors or on-line thermal estimations. This is also applied to the work proposed in this chapter. For the traffic monitoring, this will require four different counters in each network router. The four counters are updated at every timing window which depends on the nature of the traffic. These four counters are combined using weighted average and used to dynamically predict the traffic workload. As a control action, ThermalHerd uses distributed traffic throttling (router-wise) to decrease the local workload by throttling the input traffic. However, the algorithm used for throttling is quite complicated as it uses exponential function to increase the throttling. It also distinguishes between the throttling level of local traffic (injected locally) and the traffic arriving from neighbour routers. Lastly, ThermalHerd uses proactive and reactive routing schemes which both rely on the thermal information communicating using the NoC channels. However, there is a lack of a concrete description on how these schemes work. Moreover, the two schemes are introduced based on intuitive observation regarding the thermal correlation and traffic balancing.

It is crucial in designing NoC that any additional hardware should not consume a large percentage of silicon area compared to the router and IP core blocks. Power consumption is also an important system performance metric. Hence, extra power dissipated (heat generated) by any additional hardware could lead to amortize the benefit of adopting it. In the ThermalHerd case (presented in [148]), the work is lacking in any implementation ideas and thus all the presented evaluation results can be questioned because implementing such a multifarious framework could cost considerable area and power budget.

Exploring 3D-NoC routing capabilities to better control the chip temperature and heat distribution has been partially explored recently in [45], where a non-minimal 3D routing called downward XYZ (DW-XYZ) routing is employed. The authors propose to use DW-XYZ to migrate the workload from upper to lower layers. They argue that by doing so, it will be possible to improve the heat diffusion efficiency of the chip, since lower layers are closer to the heatsink. To comply with bandwidth constraints and to minimize the impact on performance, the level of downward routing is determined by the traffic counters. These counters predict the workload at each router along the vertical pillar and determine the downward level. 'Pillar' in this context means a set of routers located in different layers and aligned on top of each other. However, the implementation of this approach requires an additional hardware circuit to store the values of these counts and additional wiring to communicate them. This extra hardware will increase quadratic with the increase of the number of stacked layers in the chip. When the thermal limit of the chip is exceeded, they propose to use vertical throttling combined with the DW-XYZ to mitigate heat in the chip and at the same time route packets away from the throttled paths.

The DW-XYZ method will migrate the workload only in the vertical direction, assuming that the lower layers are cooler. This assumption is true when the communication workload of the NoC is homogeneous, which is not always the case; for example, assuming that the hotspots (for instance, memory units in the multiprocessor system-on-chip (MPSoC) are located in the lower layers. This is because memory units have higher traffic intensity and would generate more heat; it is better to be placed in the layers closer to the heatsink. In this case, the DW-XYZ based approach on traffic is not likely to be able to migrate the workload to an already congested lower layer. The other problem is that the values of traffic counters need careful tuning, as they depend on the traffic distribution and NoC capacity. Also, this method does not exploit the cool paths within each layer, as it is limited only to vertical migration of the workload.

A method which is more flexible in manoeuvring the packets away from the hotter paths is required. A method which exploits cool paths wherever they are and whenever they become available in the chip would be desirable.

6.3 DYNAMIC PROGRAMMING RUN-TIME THERMAL MANAGEMENT (DPRTM)

6.3.1 *System Overview*

This work introduces an adaptive strategy based on dynamic programming to effectively diffuse heat throughout the 3D-NoC (see Figure

6.1). This method exploits any cool paths for routing, regardless of their positions. This leads to better heat distribution and proactively reduces the chances of thermal hotspot formation. This course of action takes place as long as the thermal limit of the chip has not been exceeded. This phase is called coolest path-first (CPF) (Figure 6.2a). A reactive strategy also proposed to take control when the chip exceeds the thermal limit. In this case, distributed-throttling (DT) is used to reduce the chip temperature and the routing strategy becomes least throttled paths-first (LTPF) (Figure 6.2b). Thus a priority is given to the least throttled paths among the available paths between the source and destination of the packet. This will minimize the impact of throttling on performance and will, also, aid faster cooling of the throttled nodes.

The methodology used to implement the two regenerative routing schemes is based on the dynamic programming network theory explained in detail in Section 5.3 in Chapter 5, with the exception that the cost function used in the DP-network is changed to reflect the new requirements, as can be seen in the next section.

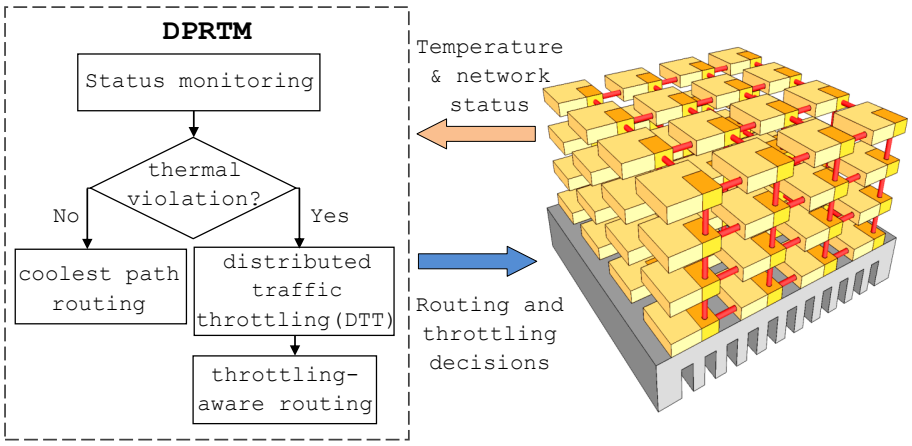
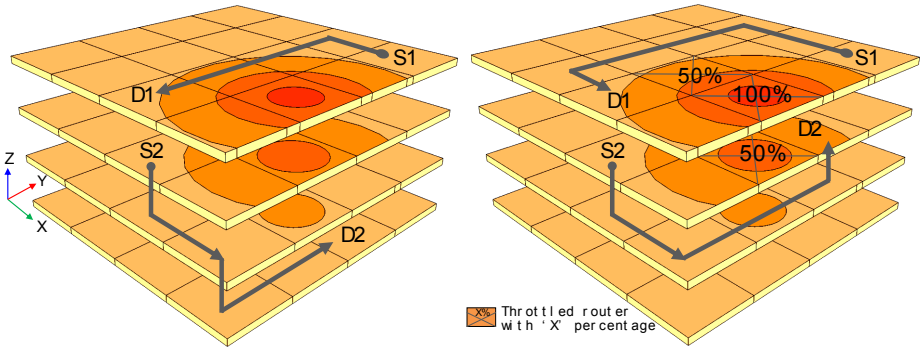


Figure 6.1: The proposed dynamic programming-based run-time thermal management (DPRTM) for NoCs

6.3.2 *Path Cost Computation*

The key idea of the proposed approach is to use router temperature (T_r) to assign a cost for the router during the proactive phase. This will enable better thermal distribution and keep the chip under the thermal limit for as long as possible. When the thermal limit is exceeded, routing would be adapted to the irregularity introduced by throttling during the reactive phase. Therefore, the throttling level (TR_r) of each



(a) coolest path-first (CPF) (b) least throttled paths-first (LTPF)

Figure 6.2: Illustration of the routing paths in DPRTM system.

router is introduced to the cost function. The cost of each routing node in the proposed DP-network C_r is computed as follows:

$$C_r = \alpha * T_r + \beta * TR_r, \tag{6.1}$$

where α and β are the weights assigned to router temperature and throttling level respectively. Throttling has a very high impact on performance. The cost introduced by throttling must dominate other costs in order to make sure that the throttled paths are dominating others. Therefore, weights are set such that during the proactive phase (throttling is zero for all nodes), DP chooses the coolest path among the available minimum paths between any source/destination pair (see Figure 6.2a). When throttling occurs, DP must route packets away from throttled paths even if the resulting routing path is non-minimum (see Figure 6.2b). To ensure the throttling mechanic is carrying a heavy weight, β is chosen significantly larger than α.

6.3.3 *Coupling DPRTM with 3D-NoC*

The DP-network approach is illustrated in Figure 6.3. It consists of a network of distributed computational units. The topology of the network resembles the defined graph topology, which is the communication structure of a NoC. At each node, there is a computation unit, which implements the DP shortest path calculations (Eq. 5.2 in Chapter 5). The numerical solution of the unit will be propagated to the neighbour units. The DP-network is tightly coupled with the NoC and each computation unit locally exchanges control and system parameters with the corresponding NoC node. The DP-network quickly resolves the optimal solution [114] and will pass the control decisions to the router. The cost can be communicated across the DP-network by dedicated links. This will enable fast convergence of the network to respond to rapid change in the cost function (e.g., when the cost

function is the buffer level). Another option is to use the existing NoC structure to propagate the DP-unit cost which will cause some delay in the convergence of the DP-network. In this work, the cost is the temperature and throttling level (which is determined by temperature). The sampling time of temperature is in the order of 10's of microseconds or even a millisecond. This is a very long time period compared to the clock frequency of the NoC communication fabric, which makes it possible to use the existing NoC structure to propagate the cost of the DP-network.

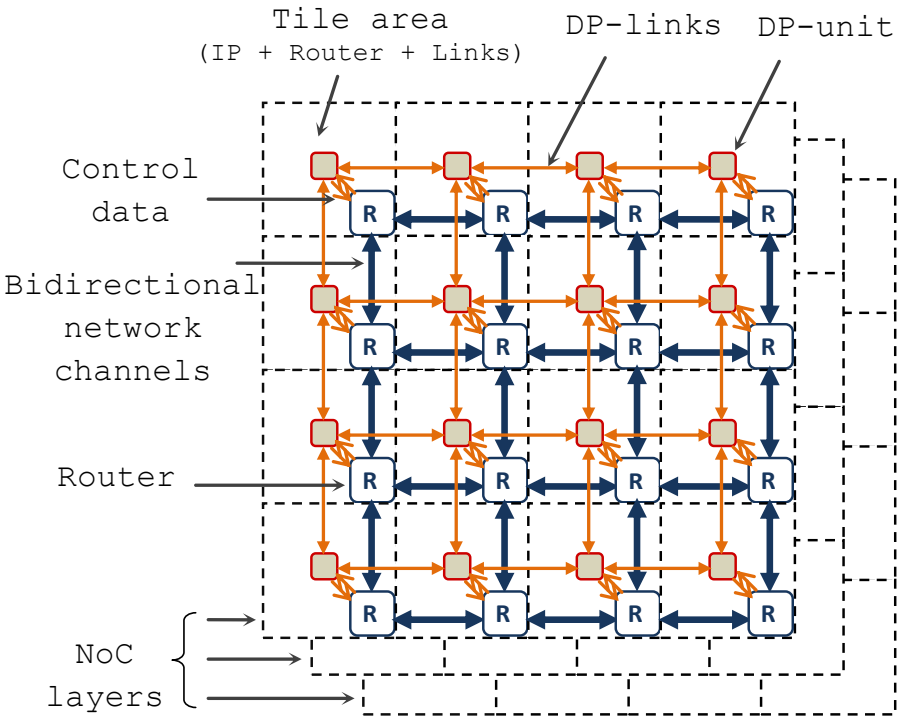


Figure 6.3: DP-network coupled to a 3D mesh NoC

6.3.4 *Throttling Techniques*

Throttling is an effective cooling technique suggested by many researchers [148, 45, 109]. Once the chip temperature exceeds the thermal limit, throttling will trade performance with temperature. Therefore, a careful design of the throttling technique to minimize its impact on performance and maximize the cooling efficiency of the chip is of crucial importance. In [45], because their routing adaptation strategy is limited to downward routing, vertical throttling is employed. Depending on the temperature level, the whole pillar, all tiles in different layers but with the same planar coordinates, may be throttled to mitigate heat. This method is effective in terms of the cooling speed, but has a high impact on performance of the NoC because the throttling is done pillar-wise. Also, there is a need to communicate the temperature

(and the traffic count in the proactive phase) among the routers that belongs to the same pillar, which requires wiring overhead. In this work, however, our proposed routing strategy is more resilient and thus a DT is employed. Each node (tile) decides its throttling level depending on whether its temperature exceeds the thermal limit or not (see Algorithm 6.1). Note that in this throttling technique the decision is taken locally by the node, hence there is no need for extra wiring to communicate temperature in order to decide the throttling level. When the local node temperature T_n is above the thermal limit of the chip T_{MAX}, the throttling level L_n is increased by one level if the temperature is lower than the maximum throttling level. When T_n is below T_{MAX}, the throttling level L_r is decreased by one level if it is not already zero.

Algorithm 6.1 Decide throttling level $TR_n(t)$ of the current node n_c

Definitions: –
n_c is the current node.
Ψ is the set of discrete throttling levels.
Input: –
L_n is the previous throttling level of n_c.
$T_n(t)$ is the temperature of n_c.
T_{MAX} is the chip thermal limit.
Output: –
$TR_n(t)$ is the current throttling level of n_c.
$L = |\Psi|$,

```
1: if T_n(t) >= T_MAX then
2:    if L_n < L then
3:       L_n = L_n + 1
4:    end if
5: else
6:    if L_n > 0 then
7:       L_n = L_n − 1
8:    end if
9: end if
10: TR_n(t) = Ψ(L_n)
```

6.3.5 *NoC Routing in the DPRTM*

Algorithm 6.2 presents the operations required for updating the routing directions with a DP-network. At each node unit, there are k inputs from the k neighbour nodes for the expected costs. The local router temperature comes from distributed embedded sensors in the chip, similar to any of the proposed on-chip sensors in [150, 48]. While the throttling level is computed locally in each node by Algorithm 6.1. The output of the DP-unit at node n_c is the updated expected cost

$V^*(n_c, n_d)$ and is sent to all adjacent nodes. The main algorithm is outlined in lines $1-11$. For each given destination n_d and direction k, the expected cost will be computed and the minimum cost will be selected, as stated in line 9. The optimal direction for routing is computed and used to update the routing directions, as stated in line 10. The cost function used in this algorithm is defined in Eq. 6.1.

6.3.6 *DPRTM Deadlock Freedom*

To guarantee the deadlock freedom of the proposed routing algorithm, the *negative-first* turn model [65] is adopted to avoid deadlocks. *Negative-first* routing has better degree of adaptiveness than other types of turn-model routing, in particular the *west-first* and the *north-last*. The *negative-first* turn model prohibits two turns in 2D mesh networks and six turns for 3D mesh NoC to guarantee freedom from deadlock. Therefore, the following turns are removed: the *south-west, south-down, east-north, east-down, up-west and up-north*. Thus, no cycle can be formed in the channel dependency graph and the routing will never form a cycle in the network. Alternatively, other turn models, such *west-first, north-last* and *odd-even* [47], can also be applied in the proposed DPRTM to avoid deadlock with a similar performance at the designer's disposal.

6.4 HARDWARE IMPLEMENTATION

To implement the proposed DP-network, a number of different strategies can be investigated. For example, the DP-network can be realized using an asynchronous circuit, similar to that proposed in Chapter 3 regarding run-time deadlock detection. Another possible approach would be to use an analog circuit design scheme which could enable high performance and low power realization as in [120, 119]. Alternatively, this section investigates the hardware realization of the DP-network using synchronous circuits. The aim is to realize a DP-network hardware that augments NoC's routers to provide an adaptive strategy to diffuse heat throughout the chip geometry. The resultant hardware overhead is reported in terms of circuit (DP-unit) and interconnects (DP-links) which both constitute the proposed DP-network.

Figure 6.4 shows the architecture of a router, which enables dynamic thermal-aware routing. The architecture supports 3D mesh NoC. The router circuit is the state-of-the-art design [134, 25] for 2D mesh with an additional two channels for upper and lower layers (labelled as *Up* and *Down*). The design is augmented by an additional block (depicted with a dotted line) which implements the proposed adaptive strategy. The temperature sensor circuit provides the DP computational unit with the local cost. The local cost and the costs coming from upstream routers (neighbour routers) are used to compute the *cost-to-go* which will be

Algorithm 6.2 Pseudo code for computing the routing directions performed by each DP-unit of the proposed DPRTM system

Definitions: –
n_c is the current node,
$\mathcal{N}(n_c)$ is all neighbor nodes of node n_c,
par denotes parallel operations.
Inputs: –
n_d is the destination node,
$V^*(i, n_d) \quad \forall i \in \mathcal{N}(n_c)$, the costs for all neighbors of n_c to n_d,
T_c is the current node local temperature.
TR_c is the current node throttling level (determined by Algorithm 6.1).
Outputs: –
$\mu(n_c, n_d)$ is the optimal direction from node n_c to n_d,
$V^*(n_c, n_d)$ is the optimal cost form node n_c to n_d.

```
if n_d = n_c then
    V*(n_c, n_d) = 0
    μ(n_c, n_d) = Local_IP
else
    Local_Cost = αT_c + βTR_c
    par for all directions k ∈ N(n_c) do
        V'(n_c, k, n_d) = V(k, n_d) + Local_Cost
    end par for all
    V*(n_c, n_d) = min_∀k V'(n_c, k, n_d)
    μ(n_c, n_d) = arg min_∀k V'(n_c, k, n_d) {Update optimal routing directions}
end if
```

propagated to all neighbour routers. The design so far merely consists of combinational circuits. However, the control unit block shown in the figure is a mixture of sequential and combinational circuits. It can be realized simply by using a synchronous counter and a few logic gates. The counter provides a sign to indicate which node is regarded as the destination and also supplies an address reference to update the routing directions inside the router. The implementation presumes a router supporting routing table to store the routing directions for each particular destination in the network. The DP-network successively updates these routing directions.

The DP-unit produces zero as a *cost-to-go* if the current router identification (ID) number is equal to the destination ID (i.e., the target destination is the current node), otherwise the DP-unit outputs the result of the DP computation. The coolest path computation presented in Section 5.3.2 is implemented using the DP computational unit, which is shown in more detail in Figure 6.5. The DP-units from different routers in the NoC are interconnected so as to form a DP-network. The coolest path computation necessitates a minimum operation to evalu-

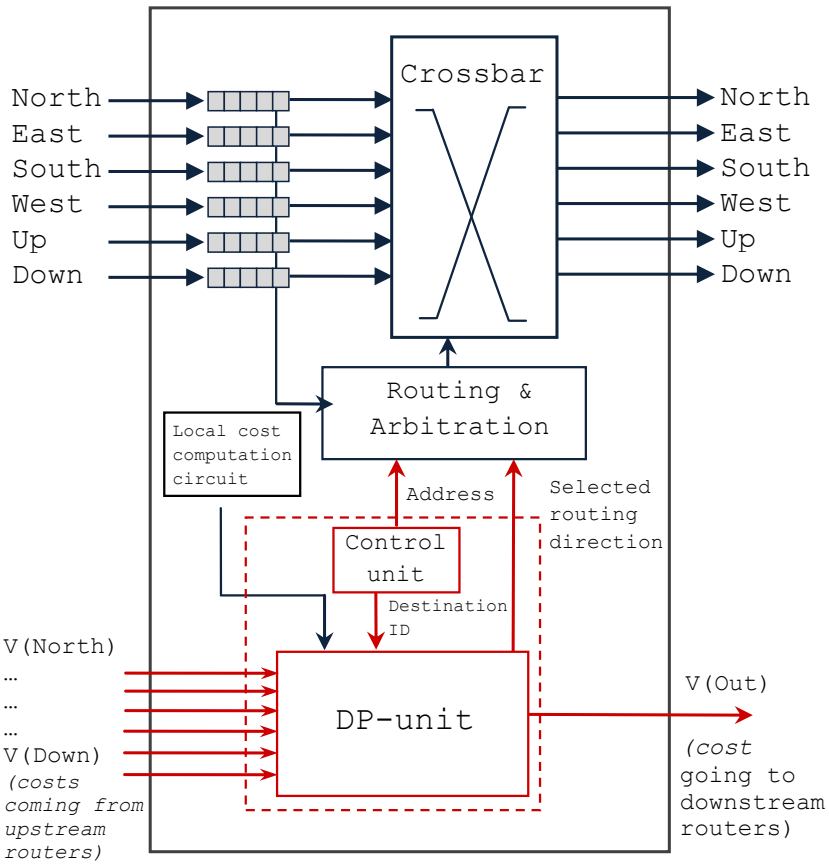


Figure 6.4: Schematic design of 3D mesh NoC router with a DP-network. The computation unit is integrated to the local router to enable dynamic thermal-aware routing. The DP computational unit interconnects with other DP-units locating at adjacent tiles.

ate and compare the coming costs from neighbour routers. This can be realized using comparators and data multiplexers. Also, an adder is required to sum the local router cost (see Eq. 6.1) with the minimum cost coming from the upstream routers. Moreover, another data multiplexer is needed to output the associated action for the minimum expected cost. Therefore, the basic circuit in a DP computational unit for a 3D mesh NoC comprises of two adders, six comparators and six data multiplexers. The DP-unit circuit is also required to compute the best routing direction for the particular received destination (indicated by the *Destination* ID) and then updates the routing table located in the *Routing & Arbitration* block, see Figure 6.4. This operation requires another four data multiplexers (3-bit each for the six directions router presented in the figure).

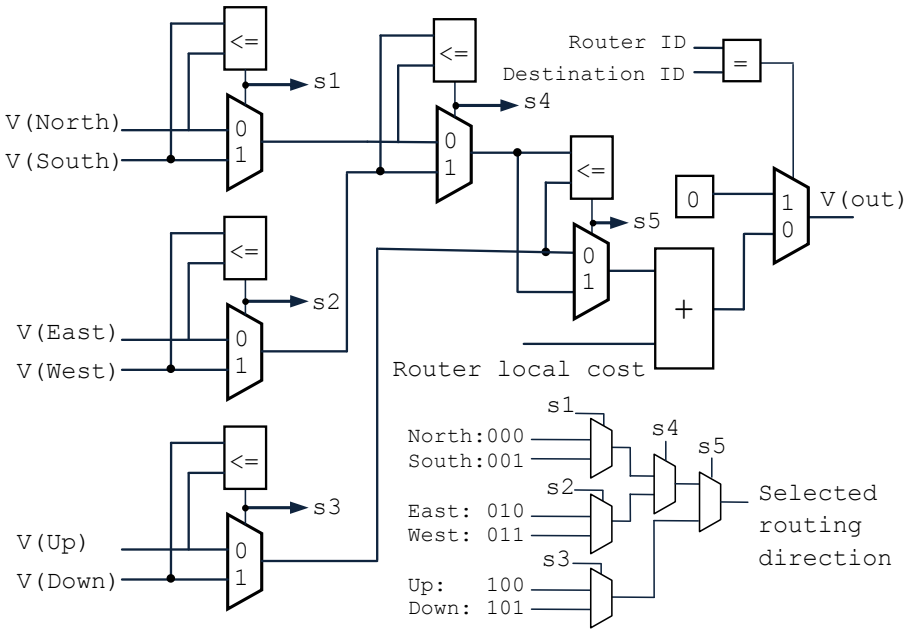


Figure 6.5: Realization of the DP computational unit

6.5 RESULTS AND DISCUSSION

6.5.1 *Evaluation Methodology*

Several experiments are conducted on 3D-NoC based CMP to evaluate the proposed DPRTM system. The CMP configuration is adopted from the Intel 80-Tiles chip [157]. The floorplan of the evaluated chip is arranged as an $8 \times 8 \times 4$ 3D matrix of tiles (core + router + links). Moreover, *XYZ* routing with global-throttling (GT) and the RTM scheme proposed in [45] are modelled and their results are compared with the proposed DPRTM scheme. The RTM scheme proposed in [45] comprises the following: 1) the traffic-aware downward (TADW) routing for the proactive phase; 2) vertical-throttling (VT) for thermal mitigation; and 3) the thermal-aware vertical throttling (TAVT) for the reactive phase.

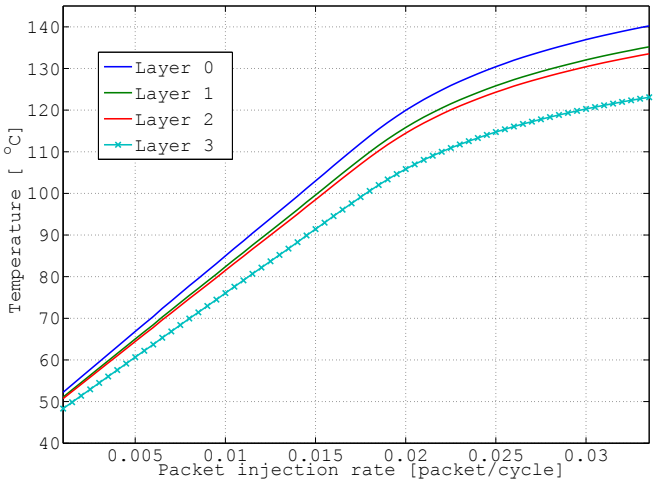
A 3D-NoC interconnected as a mesh topology is chosen with seven port router architectures as a natural extension to the five port routers used in 2D mesh. Each input channel consists of eight flit buffers and the packet size consists of 16 flits. The results are captured after a warm up period of 10,000 clock cycles. The initial silicon temperature and the ambient temperature are assumed 25°C, similar to previous work [148].

6.5.2 *Proactive Routing Evaluation Results*

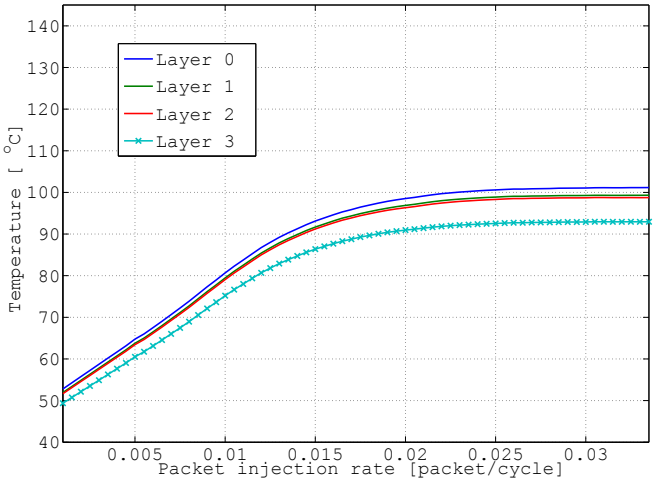
Several experiments are conducted to evaluate the effectiveness of the proposed CPF routing algorithm in the proactive phase such that there is no thermal limit imposed on the NoC. The first experiment

presented in Figure 6.6 shows the maximum temperature of each layer in the simulated NoC with an increase of the PIR for different routing algorithms. The maximum chip temperature can reach up to 140°C in case of *XYZ* routing (Figure 6.6a), 101°C for the CPF routing (Figure 6.6a) and 93°C for the TADW routing (Figure 6.6a) proposed in [45]. The differences in steady state maximum temperature curves among different routing algorithms can be explained by referring to Figure 6.7. This figure shows the performance of each of the routing algorithms in terms of average network delay and the achievable throughout. Because of the nature of the traffic used in this experiment (*uniform* traffic distribution), the deterministic *XYZ* routing has superiority over other adaptive routing algorithms, as has been repeatedly reported in previous work [47, 65]. The higher throughput achieved by *XYZ* routing means the NoC has been able to deliver more packets compared to the other routing algorithms and thus consumed more power. The more power consumed means more heat generated in the chip which is very clear by comparing Figure (6.6a, 6.6b & 6.6c) and Figure 6.7. Although the TADW is a modified version of *XYZ* routing, it behaves the worst among the three routing algorithms. This is because the migrated traffic towards the heatsink (Layer 3 in the experiment) increases the average network delay and causes the NoC to prematurely saturate. This is mainly for two reasons: 1) packets are highly likely to take non-minimal routes (more hops) if the traffic counters are below some threshold; 2) the migrated traffic will cause congestion in the lower layers of the NoC. The TADW result obtained in Figure 6.7 is consistent with the result presented in the TADW original paper [45] (achieved only half the *XYZ* routing throughput).

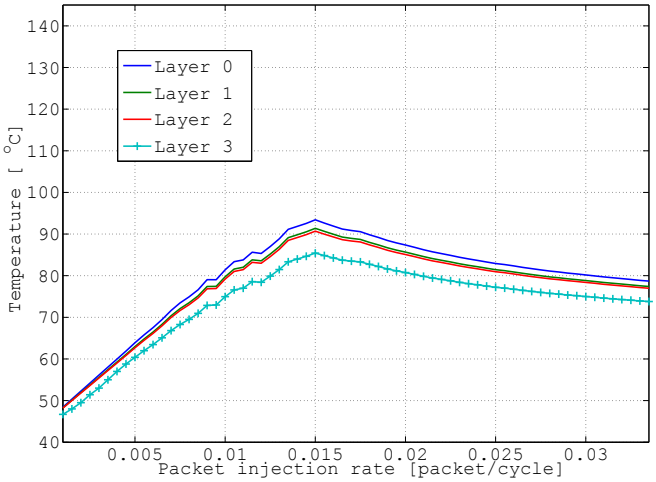
Although the previous experiment gives a clear insight into the overall performance and maximum temperature profile for each of the routing algorithms, it cannot reveal clearly which routing algorithm is performing better in the proactive phase. Therefore another experiment was conducted in order to measure the network throughput, average delay and PIR that causes the first violation for some thermal limits. Table 6.1 presents the results of this experiment with three different thermal limits (70°C, 75°C and 80°C). The chosen thermal limits are in the reasonable working temperature range and in line with the work [148]. The table also shows the proposed routing improvement over TADW. For instance, given 75°C as the thermal limit, the CPF routing can achieve 14.5% higher throughput and 85.8% less average network delay compared to TADW routing. This is because the PIR at which the violation occurs is 0.009 in the case of the CPF compared to 0.008 in the case of the TADW. This clearly demonstrates the run-time dynamic capability of the proposed routing to manoeuvre packets to exploit the coolest paths which delayed the violation to a higher PIR and therefore give the space to the NoC to deliver more packets and hence improve the throughput.



(a) *XYZ* routing



(b) CPF routing (proposed)



(c) TADW routing

Figure 6.6: Maximum temperature of each layer for different routing algorithms under *uniform* traffic distribution.

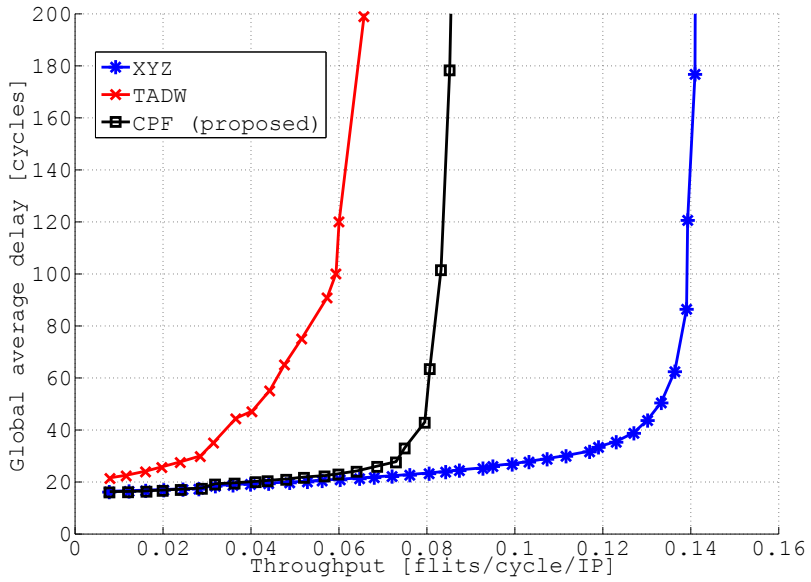


Figure 6.7: NoC performance results for different routing algorithms under *uniform* traffic distribution.

Routing Algorithm	Throughput	Average Delay	PIR
Thermal limit = 70° C			
XYZ	0.0040	19.7	0.006
TADW	0.0061	86.1	0.007
CPF (proposed)	0.0062	22.2	0.007
CPF improvement over TADW	**1.6%**	**74.2%**	**0%**
Thermal limit = 75° C			
XYZ	0.0074	20.2	0.0075
TADW	0.0086	159	0.0080
CPF	0.0101	25.7	0.0090
CPF improvement over TADW	**14.5%**	**85.8%**	**11.1%**
Thermal limit = 80° C			
XYZ	0.0114	22.4	0.009
TADW	0.0129	253	0.0095
CPF	0.0145	45.2	0.010
CPF improvement over TADW	**11.0%**	**82.2%**	**5.0%**

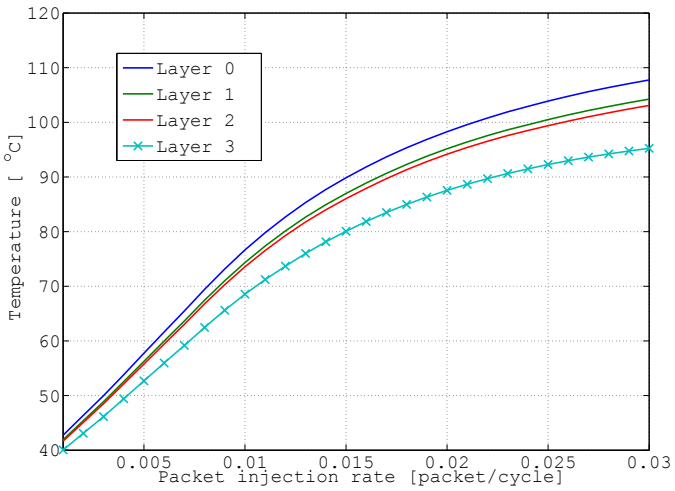
Table 6.1: Achievable performance metrics for different routing algorithms with different thermal limits under *uniform* traffic.

Thermal limit = 70°C			
Routing Algorithm	Throughput	Average Delay	PIR
XYZ	0.0072	37.4	0.009
TADW	0.0071	115	0.009
CPF (proposed)	0.0106	27.2	0.011
CPF improvement over TADW	**33%**	**76.5%**	**18.2%**
Thermal limit = 75°C			
XYZ	0.0087	73.3	0.010
TADW	0.0114	361	0.012
CPF	0.0128	34.4	0.012
CPF improvement over TADW	**10.2%**	**90.5%**	**0%**
Thermal limit = 80°C			
XYZ	0.0122	180	0.012
TADW	0.0147	466	0.014
CPF	0.0171	72.0	0.015
CPF improvement over TADW	**13.5%**	**84.5%**	**6.7%**

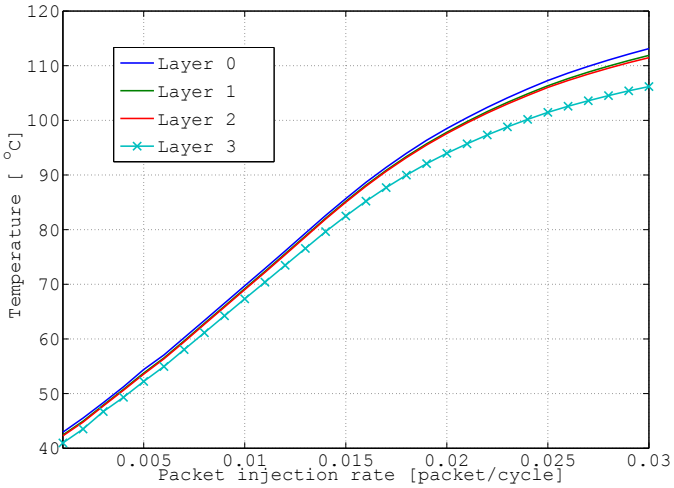
Table 6.2: Achievable performance metrics for different routing algorithms with different thermal limits under *transpose* traffic.

To investigate another traffic scenario, Figure 6.8 shows the maximum temperature of each layer for different routing algorithms under *transpose* traffic. In this figure, the maximum temperature of Layer 0 can reach up to 108°C, 112°C and 88°C for the *XYZ*, CPF and TADW routing algorithms respectively. In the same manner, this figure can be explained using the NoC performance curves shown in Figure 6.9. The CPF routing achieved around 25% improvement compared to *XYZ* routing and 42% compared to TADW. This is certainly reflected in the maximum temperature curves (Figure 6.8b), as the routing with CPF generated more heat due to more power consumed to deliver more packets.

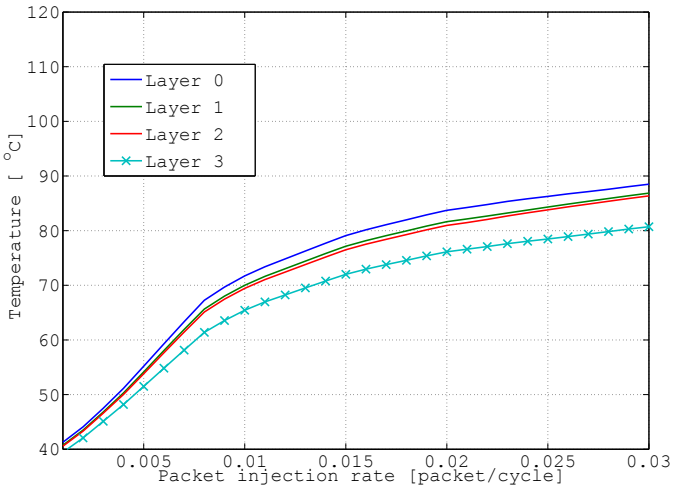
Table 6.2 shows the achievable performance metrics with different thermal limits under *transpose* traffic. Once again, the results suggest that the proposed proactive routing improves all the performance metrics compared to the other routing algorithms. CPF improvement in terms of throughput ranges between 10 to 33 percent and, in terms of network average delay, ranges between 76 to 90 percent compared to TADW under different thermal limits. All the results presented so far, for the proactive phase, lead to a very important conclusion that the DP-network adaptive capabilities effectively diffuse heat throughout the 3D geometry by converging to the coolest paths to route data in NoC.



(a) *XYZ* routing



(b) CPF routing (proposed)



(c) TADW routing

Figure 6.8: Maximum temperature of each layer for different routing algorithms under *transpose* traffic scenario.

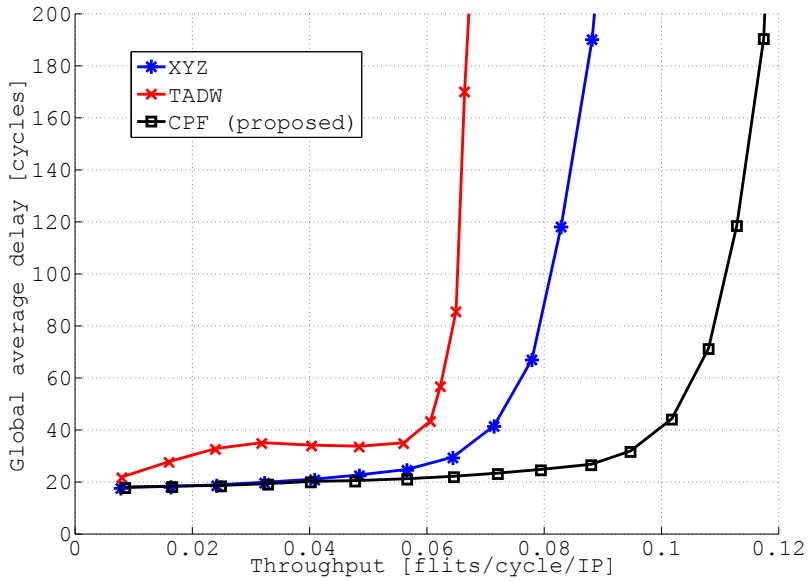


Figure 6.9: NoC performance results for different routing algorithms under *transpose* traffic scenario.

6.5.3 *Reactive Routing Evaluation Results*

In this section, the aim is to evaluate the effectiveness of the proposed reactive LTPF based routing which is primarily designed to avoid any throttled routers as far as possible. Thus, similar to the proactive phase, experiments are carried out with only one change. This change is basically a set of throttled routers imposed in the centre of the NoC which imitates a hotspot taking place in the centre. The hotspot takes place in the centre of the layer farthest from the heatsink and it reduces in the following layers closer to the heatsink and disappears in the last layer (Layer 3). The shape of the hotspot resembles a reverse pyramid, similar to the one illustrated in Figure 6.2. Therefore the throttled routers are arranged 4×4 in Layer 0 with a throttling level of 50% and the next two layers (Layer 1 and Layer 2) consist of 2×2 routers throttled 20% and 10% respectively. Figure 6.10a shows the performance results of the three routing algorithms (*XYZ*, TADW and LTPF) evaluated in the proactive phase under *uniform* distributed traffic. As a general observation, all the routing algorithms lose some percentage from their throughput acquired in the proactive phase because of the imposed throttled routers (around 10% of NoC routers are throttled). However, the reduction in performance is not equal and this study argues that the percentage of the performance loss will reflect how effective the routing algorithm can be in the reactive phase. The figure clearly shows that LTPF dominates the performance among all the rest, as it loses only 39% of its original throughput without throttling (see Figure 6.7), while the *XYZ* and TADW lose 73% and 49%

respectively. Similar patterns can be observed for the fixed throttling with *transpose* traffic (see Figure 6.10b).

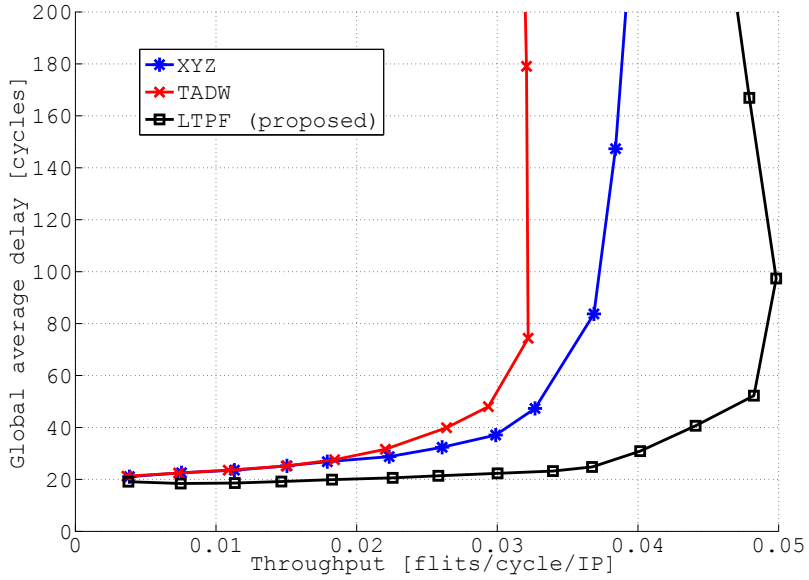
6.5.4 *DPRTM Evaluation Results*

The main task for any RTM system is to regulate the maximum temperature of the chip below some predefined temperature threshold (set-point). This can mainly be achieved by monitoring the router's temperature and preventing it from exceeding the thermal threshold. The control action to regulate the temperature is the throttling. In this section, several experiments are conducted to investigate the proposed DPRTM (see Section 6.3) which comprises the dynamic-programming based routing and the DT. The DPRTM results are compared to the TAVT proposed in [45] which comprises the TADW routing and the VT.

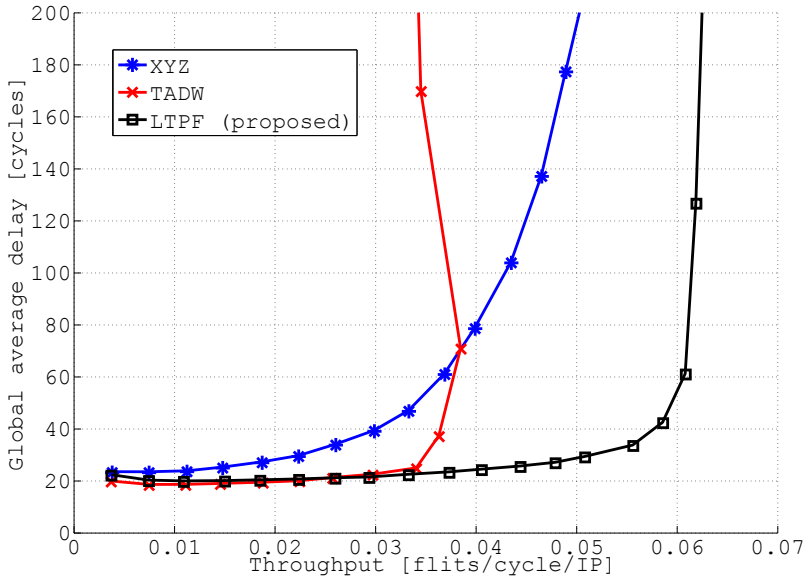
The results shown in Figure 6.11 are the resulting maximum temperature, throughput and network availability for XYZ with global-throttling (XYZ-GT) as well as both TAVT and DPRTM methods and for 800 µs of simulation time with a thermal limit of 70 °C. It can be observed that all three methods succeeded in regulating the temperature of the chip to the desired limit (Figure 6.11a). However, their resulting throughputs (see Figure 6.11b) during the regulation process are different. XYZ-GT has the highest variation of throughput between zero and the maximum throughput. The change is taking place, almost, at every other regulation sample. For TAVT, the throughput fluctuations are lower than XYZ-GT, but it still experiences considerable variations around its mean. DPRTM has the lowest variation of throughput among the three methods, which indicates the ability of DPRTM to deliver consistent quality of service under a fixed thermal constraint. On average, the throughput for DPRTM is higher than TAVT (see Tables 6.3 and Tables 6.4).

The network availability results are shown in Figure 6.11. Network availability here is simply the percentage of available routers in the presence of throttling. It should be noted that both XYZ-GT and TAVT started to throttle the network at t=130 µs, while DPRTM did not need to start throttling until t=280 µs. This indicates the ability of DPRTM to keep the temperature under the thermal limit, in the proactive phase, for longer. This is due to its higher capability to diffuse heat over the chip. This observation is also consistent with the results presented in Section 6.5.2.

When the thermal limit is exceeded, the three methods behave differently. XYZ-GT availability (Figure 6.11c) alternates between 0% and 100%. This is because it has the highest cooling speed caused by GT. However, this leads to rapid changes in the throughput described earlier, since no smart technique is employed either for throttling or for routing. The availability for TAVT (Figure 6.11d) does not reach zero, but fluctuates considerably and continuously to comply with



(a) *uniform* traffic



(b) *transpose* traffic

Figure 6.10: Performance results for different routing algorithms under fixed throttled routers concentrated in the centre of the NoC.

Thermal limit = 70°C			
RTM method	Availability	Throughput	ATtAT
TAVT	98.9%	0.0417	0.0422
DPRTM	87.9%	0.0426	0.0484
DPRTM improvement over TAVT = 14.83%			
Thermal limit = 75°C			
TAVT	99.1%	0.0474	0.0478
DPRTM	91.3%	0.0482	0.0528
DPRTM improvement over TAVT = 10.39%			
Thermal limit = 80°C			
TAVT	99.8%	0.0517	0.0518
DPRTM	95.3%	0.0551	0.0578
DPRTM improvement over TAVT = 11.63%			

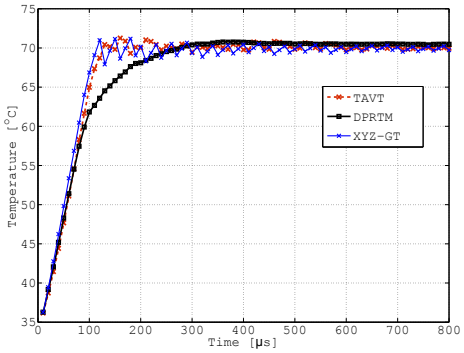
Table 6.3: The resulting network availability and the corresponding throughput different thermal limits for both RTM methods under *uniform* traffic.

the thermal constraint. In contrast to TAVT, the proposed method (DPRTM) is capable of delivering consistent network availability under thermal constraint. This is because the reactive and proactive schemes employed in the proposed DPRTM are interchanging to optimize the routing pathways depending on the critical temperature thresholds, throttled paths and the traffic developments.

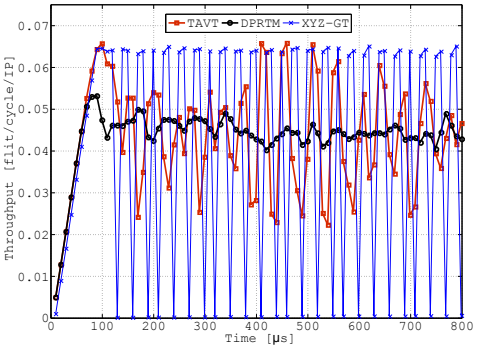
Tables 6.3 and 6.4 summarize the results of network availability and throughput for both TAVT and DPRTM and a range of thermal constraints. It can be observed that TAVT has higher network availability (on average) due to its aggressive throttling mechanism which enables fast chip cooling. For DPRTM, DT, which has a slightly slower cooling rate compared to VT, was employed. In general, higher throughput leads to higher power and thus temperature which requires higher throttling under a thermal limit. Thus, it was noticed that both methods will achieve approximately the same average throughputs. However, DPRTM delivers more throughput given the same network availability (see section 6.5.3). Thus, to measure the efficacy of both methods, the ratio of average-throughput to average-availability (ATtAT) is introduced, as follows:

$$\text{ATtAT} = \frac{\mathit{average\_throughput}}{\mathit{average\_availability}}. \tag{6.2}$$

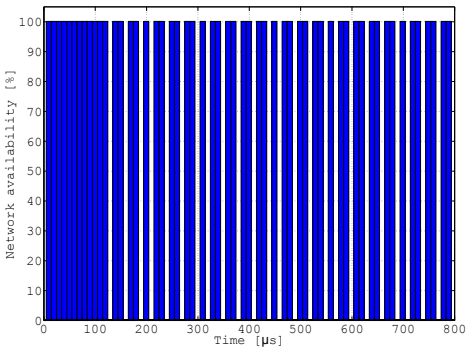
Using this metric, it can be observed that DPRTM is better than TAVT for both *uniform* and *transpose* traffics and for different thermal limits.



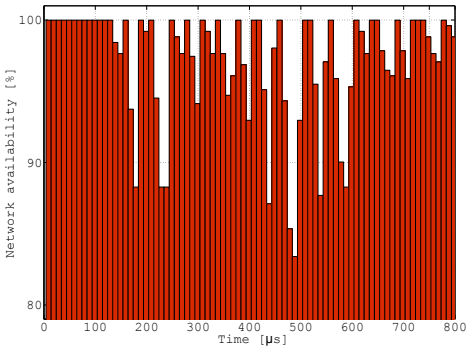
(a) Chip maximum temperature



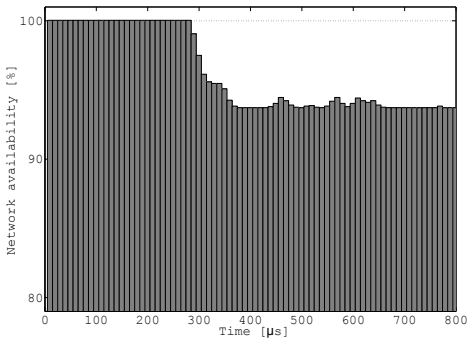
(b) The corresponding throughput



(c) The resulting network availability using XYZ-GT



(d) The resulting network availability using TAVT



(e) The resulting network availability using DPRTM

Figure 6.11: The results of temperature regulation for a thermal limit of 70 °C for different RTM methods

Thermal limit = 70°C			
RTM method	Availability	Throughput	ATtAT
TAVT	98.4%	0.0434	0.0441
DPRTM	82.4%	0.0412	0.0499
DPRTM improvement over TAVT = 13.1%			
Thermal limit = 75°C			
TAVT	99.4%	0.0474	0.0477
DPRTM	87.4%	0.0478	0.0547
DPRTM improvement over TAVT = 14.6%			
Thermal limit = 80°C			
TAVT	99.34%	0.0516	0.0519
DPRTM	91.8%	0.0564	0.0614
DPRTM improvement over TAVT = 18.3%			

Table 6.4: The resulting network availability and the corresponding throughput different thermal limits for both RTM methods under *transpose* traffic.

6.5.5 *Area and Power Estimation*

It is crucial in designing NoC that any additional hardware should not consume a large percentage of silicon area compared to the router and IP core blocks. Power consumption is also an important system performance metric. To evaluate the area and power overhead of the proposed method, the design of the DP computation unit presented earlier is expressed in Hardware Description Language (Verilog) for a 2D and 3D mesh NoC. This is then synthesized using Synopsys Design Compiler and mapped onto the FARADAY UMC 65nm technology library. Table 6.5 summarizes the result for the area and power estimations of the router augmented by the DP-unit for 2D and 3D mesh NoC with different network grid size. It also shows the area and power overhead added by the DP-unit to the total router area and power, which have been estimated using ORION 2.0 [94]. It is clear that the overhead (both area and power) increases with the increases in the NoC grid size and this is because the DP-unit is implemented in this work using table-based routing. However, the overhead is not that big. For instance, the area overhead is 0.59% and the power overhead is 0.4% with respect to the router for a NoC with a size slightly bigger than the one used in Intel 80 tiles Teraflop chip. The router operating frequency is assumed 3GHz, similar to [157], while the DP-unit frequency is calculated using Eq.5.7. Assume a chip similar to Intel teraflop [157], i.e., the number of destinations is 80, NoC, diameter is

NoC size	Area (mm^2)	Power (W)	Area overhead	Power overhead
8×8	0.334	0.488	0.39%	0.2%
10×10	0.334	0.489	0.59%	0.4%
$4 \times 4 \times 3$	0.434	0.672	0.36%	0.2%
$8 \times 8 \times 3$	0.437	0.672	1.13%	0.6%
$10 \times 10 \times 3$	0.439	0.673	1.69%	1.0%

Table 6.5: Router plus DP-unit area and power synthesization result

18 and with 10us being the temperature sampling time (as suggested by [152, 84, 85]), then clocking the DP-unit with 200MHz will suffice.

The DP-network cost can be communicated across the NoC by dedicated links. This will enable fast convergence of the network to respond to rapid change in the cost function. Another approach is to use the existing NoC structure to propagate the DP-unit cost which will cause some delay in the convergence of the DP-network. In this work, the cost is the temperature and throttling level (which is determined by temperature). The sampling time of temperature is in the order of 10's of microseconds or even a millisecond. Thus both approaches are possible at the designer's disposal. If dedicated links are used for the implementation then the total number of extra wires required can be computed using the following formula:

$$DP_{wires} = s(n + m), \tag{6.3}$$

where n and m are the numbers of input and output channels respectively and s is the number of bits to represent the *cost_to_go* value. The overhead of the DP-interconnects for the data used in this study's experiments was evaluated with the following: $n = m = 5$ for 2D mesh NoC, $n = m = 7$ for 3D mesh NoC, flit size of 39 bits and s with 10 bits resolution. Since the DP-unit operating frequency is quite low compared to the NoC operating frequency, this study suggests serializing the DP-interconnect and increasing DP-unit operating frequency to save area. For instance, instead of running the DP-unit at 200MHz, it can be run at 1GHz, thus saving 80% of the DP-interconnect area. The Orion 2.0 simulator [94] was used to estimate the link area and power. Table 6.6 shows the NoC and DP link area, as well as power and the percentage overhead added by the DP-interconnect. The DP-interconnect overheads are very small and will become smaller if bigger flit sizes are used in NoC. A 1GHz as the DP-unit clock frequency and 2 serial wires to communicate the *cost-to-go* between the DP-units are used evaluate the results in Table 6.5 & 6.6.

NoC type	Link area (um^2)	Link power (uW)	Area overhead	Power overhead
2D mesh	2266	439	4.8%	0.2%
3D mesh	3174	614	4.9%	0.1%

Table 6.6: NoC plus DP-interconnects area and power result

6.6 SUMMARY AND CONCLUSIONS

Due to aggressive technology scaling and the migration towards multi-layer 3D VLSI, future 3D NoC-based systems will face difficult challenges. The most alarming challenge is the thermal one. For 3D-NoC-based CMP, both communication and computation workload contribute in heat generation. However, it has been reported that for some communication centric applications, routers can dominate processing elements as a heat source. Thus, proper management of communication can achieve thermal mitigation for the whole chip. This chapter proposes an adaptive, collaborative dynamic programming-based thermal management system for NoC (DPRTM). It manages the routing workload dynamically to achieve thermal mitigation and control. Routing in DPRTM adapts in two modes: coolest path-first (CPF) and least throttled paths-first (LTPF). The first mode is active while the chip works within the thermal limit and is aimed to diffuse heat from the 3D chip geometry and prevent thermal hotspots for as long as possible. When the thermal limit is violated, DPRTM responds by throttling the routers where violation takes place. This action introduces irregularity in terms of communication performance in the network. Thus, DPRTM adapts to this irregularity by changing its routing strategy to give priority to the least throttled paths in order to minimize the performance impact of throttling. Since the proposed routing adaptation strategy is resilient to any throttling taking place, a distribute clock throttling strategy is employed in the DPRTM. Hence, each router decides upon its throttling level depending on whether its temperature exceeds the thermal limit or not and thus the decision is taken locally by the router.

The proposed DPRTM is rigorously evaluated and the results show that it overcomes other RTM systems in terms of adaptation efficiency and thermal regulation. The comparisons are made with the state-of-the-art XYZ routing combined with GT and a recent published work [45] which consists of proactive and reactive routing and VT schemes. Results show that DPRTM achieves up to 33% improvement in throughput compared to other RTM systems for 3D-NoC under the same thermal constraints.

As a future direction, it is important to investigate the methodology of using different cost functions to implement the DP-network and evaluate their performance impact. Exploring the benefits of using different throttling techniques with the proposed RTM system is also

another possible direction. Furthermore, the use of DVFS and other temperature mitigation techniques with the proposed DPRTM can be investigated.

7 CONCLUSIONS AND FUTURE WORK

7.1 SUMMARY AND CONCLUSIONS

Recently there has been a paradigm shift from computation centric towards communication centric in designing high performance computing chips and SoCs. This shift has been fuelled by the need to handle increasingly challenging on-chip communication in a low power and scalable manner. One step forward to achieving this is the replacement of on-chip buses with scalable on-chip network (or NoC). NoCs quickly become essential to support the distributed VLSI design style (*e.g.*, SoCs and CMPs). In many cases in such systems, scheduling and managing communication resources are the major design and implementation challenges rather than the compute resources. This thesis has argued that design time methods may not be able to fully utilize the underlying NoCs architecture to dynamically adapt to the traffic, power and thermal constraints. This thesis has proposed and evaluated a novel run-time management infrastructure for NoCs that can be realized using a distributed architecture. The proposed infrastructure can be defined and built to produce a run-time solution for different on-chip network communication challenges. This section summarizes the main work accomplished in this thesis, as well as presenting the main conclusions.

Data routing becomes a critical issue in NoCs design due to the limited area and power budgets inside the chip. Routing is a key factor to determine the achievable network performance compared to the ideal performance which is determined solely by the network topology. Adaptive routing has very appealing features when it comes to achievable performance and also other aspects like load balancing and faults avoidance. However, NoCs with adaptive routing are susceptible to deadlock, which could lead to performance degradation or system failure. Deadlocks are either avoided at design time or must be detected at run-time. Avoiding deadlocks limits routing adaptivness which is unnecessary [162]. This thesis studies deadlock detection in NoCs and proposes an efficient run-time method which guarantees the discovery of all true-deadlocks without false alarms, in contrast with state-of-the-art approximation and heuristic approaches. Experimental results confirm the merits and the effectiveness of the proposed method (TC-network). The new method eliminates the need for any kind of time out mechanism and delivers true deadlock detections independent of the network load and message lengths, rather than approximating with congestion estimation, as in the existing meth-

ods (as presented in Chapter 3). It has been observed that timing based methods may produce two orders of magnitude more deadlock alarms than the TC-network method. Combining the TC-network with an end-to-end recovery scheme to produce a true adaptive routing algorithm outperforms different routing algorithms based on deadlock avoidance (*XY*, *odd-even* and *west-first*). Moreover, the hardware overhead for the TC-network has been examined and reported. The method has also been applied to 3D-NoC and a prototype is fabricated using three tiers chip interconnected using TSV technology. Prototype testing results suggested that the TC-network is effective and detects deadlock rapidly in a NoC platform.

The advent of 3D VLSI technologies promises to tackle the challenging global interconnects delay which becomes a performance obstruction to on-chip systems design. A variety of vertical cross-die interconnection techniques have been developed and TSV, in particular, offers the prospect to increase the integration capacity, besides tackling the global interconnects issue. This thesis demonstrated the effectiveness of using 3D-NoCs compared to 2D-NoCs using different design measures for different network sizes. The average shortest paths length, network performance and deadlock formation rate all show a considerable improvement when adopting 3D-NoCs compared to 2D-NoCs. This study has shown that the average shortest paths length for 3D-NoCs is 2X to 3X smaller than a 2D version of NoCs with the same number of nodes. It also shows 3D-NoCs can greatly improve the network throughput and the saturation load compared to 2D-NoCs (around 100% throughput and saturation IR improvements, as demonstrated in Chapter 4). Moreover, this study shows that deadlock formation rate is lower in 3D-NoCs compared to 2D-NoCs due to the extra spatial degree of freedom provided by the vertical channels. These findings suggest that, in general, adopting 3D-NoC is advantageous in all the studied aspects.

However, complex thermal behaviours prohibit the advancement of 3D-NoCs based systems. This work introduced an adaptive strategy to effectively diffuse heat throughout the 3D geometry. This strategy employs a dynamic programming network to select and optimize the direction of data manoeuvre in a NoC. This also led to developing a tool chain, which is based on the accurate *HotSpot* thermal model and *SystemC* cycle accurate model, to simulate the thermal system and evaluate the proposed approach. The evaluation results demonstrated that the proposed approach can significantly diffuse the hotspots from a 3D geometry and maximum temperature can be reduced by 4°C (see Chapter 5). Nevertheless, this is not enough and proper control is required to guarantee not violating the thermal limit. Therefore, this strategy has been extended to build an adaptive, collaborative dynamic programming-based thermal management system for 3D-NoC

(DPRTM). It manages the routing workload dynamically to achieve thermal mitigation and control.

Routing in DPRTM adapts in two modes: coolest path-first (CPF) and least throttled paths-first (LTPF). The first mode is active while the chip works within the thermal limit and is aimed to diffuse heat from the 3D chip geometry and prevent thermal hotspots for as long as possible. When the thermal limit is violated, DPRTM responds by throttling the routers where violation takes place. This action introduces irregularity in terms of communication performance in the network. Thus, DPRTM adapts to this irregularity by changing its routing strategy to give priority to the least throttled paths in order to minimize the performance impact of throttling. A distribute clock throttling strategy is employed in the DPRTM. Hence, each router decides upon its throttling level depending on whether its temperature exceeds the thermal limit or not and thus the decision is taken locally by the router. The proposed DPRTM is rigorously evaluated and the results show that it overcomes other RTM systems in terms of adaptation efficiency and thermal regulation. The comparisons are made with the state-of-the-art *XYZ* routing combined with GT and a recent published work [45] which consists of proactive and reactive routing and VT schemes. The results of this investigation show that DPRTM achieves up to 33% improvement in throughput compared to other RTM systems for 3D-NoC under the same thermal constraints (see Chapter 6).

7.2 FUTURE WORK

Future prospects for the work presented in this thesis can be classified as short-term and long-term research directions.

In the short-term, it is recommended that further research be undertaken in the following areas: deadlock recovery, 3D-NoCs architecture exploration and design, thermal mitigation and thermal management. In the deadlock recovery area, the work in this thesis can be extended by investigating a novel method for recovering detected deadlocks in NoCs. The existing mechanisms [22, 96, 123] were originally proposed for off-chip networks and it would be interesting to devise a new scheme to explore NoCs specific features. This could then be combined with the proposed TC-network method and evaluated in a large-scale 3D-NoCs based system.

In terms of 3D-NoCs, it is suggested that further investigation is required to find the best network architecture and topology that fit the existing and future 3D integration technologies. TSVs technique to integrate multiple wafer/die in a chip has the potential over all the other existing techniques in terms of reducing the parasitic and high interconnection density. However, there are concerns when it comes to yield and area overhead. Future research should therefore concentrate

on the investigation of efficient NoC architecture that engages with these concerns.

In terms of thermal mitigation, hotspots diffusion and management, the work presented in thesis could be continued and extended in different ways:

- Investigating the methodology of using different cost functions for the DP-network, such as imposing non-linear weights to different layers in the stacked chip, adding traffic prediction counters to improve network performance.
- Examining the use of different deadlock avoidance rules that maximize the adaptiveness of the DPRTM system (*e.g.*, *odd-even*).
- Exploring the effect of using different clock throttling techniques (*e.g.*, VT).
- Assessing the effects of using the DVFS as a heat mitigation method instead of clock throttling.
- Evaluating the performance of the proposed DPRTM with different network topologies (*e.g.*, 3D torus) with more sophisticated flow control techniques (*e.g.*, virtual channel (VC)).

In the long-term, the proposed dynamic programming based run-time resource management infrastructure can be extended to investigate different run-time NoCs challenges. A number of possible future studies using a similar design methodology are apparent; for example: fault tolerant routing, managing the voltage and frequency islands in DVFS based system, network monitoring and debugging.

Routers and interconnects of SoCs and CMPs fabricated in deep sub-micron technology face a variety of reliability issues such as: increases in vulnerability to physical phenomena (*e.g.*, alpha particle strike, electro-migration, cross-talk, *etc.*), device level interaction, manufacturing defects and PVT variation. This makes the reliability of on-chip communication crucial. In this context, a fault tolerant adaptive routing algorithm is required to avoid faulty paths or regions. This could be a fertile research area to investigate the competence of the dynamic programming routing to guide packets through the NoCs (similar to least throttled path first routing proposed in Chapter 6).

Dynamic voltage and frequency scaling exhibits excellent results for improving the energy efficiency of the multicore processor chip. It has been used in many of Intel prototype CMPs at a tile-granularity [143, 87]. Managing this fine-grain voltage and frequency island at run-time to handle different traffic load in NoCs is a promising research direction.

The design of SoCs and CMPs based NoCs is facing observability and controllability issues of deeply integrated IP cores, routers and interconnects, for the purpose of monitoring and application debugging

[159, 69]. The design methodology presented in this work could be used to investigate these issues.

Part II

Thesis Appendices

A

FABRICATED 3D CHIP SUPPLEMENTARY MATERIALS

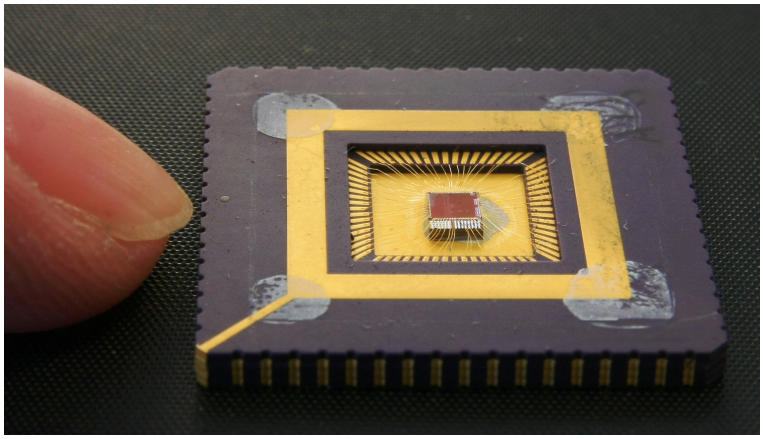


Figure A.1: Snapshot for the fabricated 3D stacked three-layer chip in the package after removing the top cover

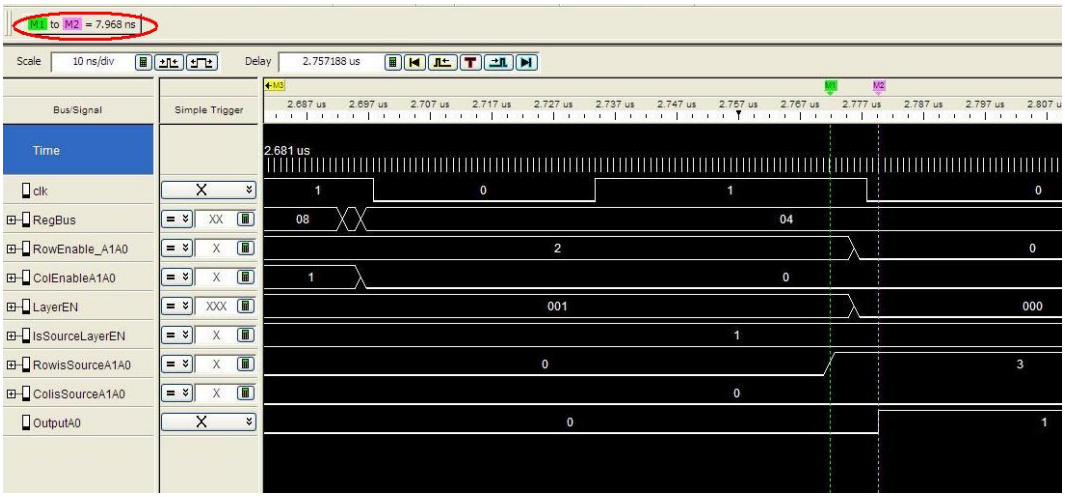


Figure A.2: Snapshot from the digital signal analyzer showing the start of TC computation and the deadlock detection signal (detection time equal to 7.9 ns)

Table A.1: Complete list of I/O pins map of the fabricated chip

Pin #	Description	Pin #	Description
1	vee	35	vee
2	vdd	36	LoCoRoOUT_isZero
3	LoRowEnableA0	37	LoCoRoOUT_A0
4	L1RowEnableA0	38	L2CoRoOUT_isZero
5	LoRowEnableA1	39	L1CoRoOUT_isZero
6	L2ColEnableA1	40	L1ColisSourceEN
7	LoColEnableA1	41	L2ColisSourceA1
8	LoEnableEN	42	L2ColisSourceA0
9	L1EnableEN	43	L1RowisSourceEN
10	L2EnableEN	44	inv_chain_output
11	L2RowEnableA1	45	L2RowisSourceA1
12	L1Regbus<4>	46	L1RowisSourceA1
13	L1Regbus<0>	47	L1RowisSourceA0
14	LoColisSourceA0	48	L2ColisSourceEN
15	clk	49	L2RowEnableA0
16	L1Regbus<3>	50	L1RowEnableA1
17	LoColisSourceA1	51	vee
18	vdd	52	vdd
19	vee	53	L1ColisSourceA1
20	L1Regbus<2>	54	LoRegbus<1>
21	L1Regbus<1>	55	LoRegbus<2>
22	LoColisSourceEN	56	LoRegbus<3>
23	L1Regbus<5>	57	L1ColisSourceA0
24	LoRegbus<4>	58	L2Regbus<3>
25	L2RowisSourceA0	59	L2Regbus<2>
26	L2Regbus<4>	60	L2Regbus<1>
27	LoRegbus<0>	61	L2Regbus<0>
28	L1CoRoOUT_A0	62	L1ColEnableA0
29	L2CoRoOUT_A0	63	L2ColEnableA0
30	L1CoRoOUT_A1	64	LoColEnableA0
31	vdd	65	L1ColEnableA1
32	L2CoRoOUT_A1	66	LoRowisSourceA1
33	LoCoRoOUT_A1	67	LoRowisSourceA0
34	vdd	68	LoRowisSourceEN

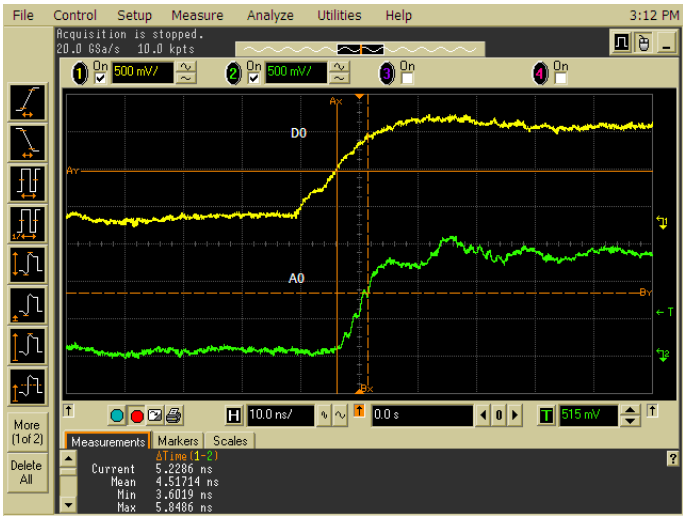


Figure A.3: Oscilloscope capture showing deadlock detection (loop size 4)

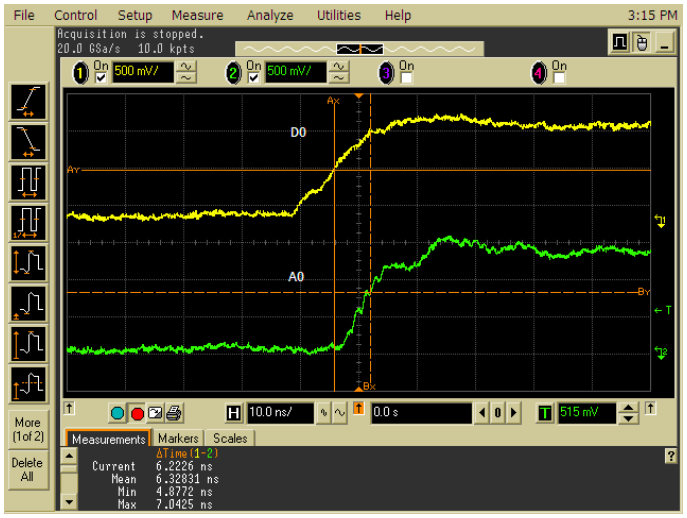


Figure A.4: Oscilloscope capture showing deadlock detection (loop size 6)

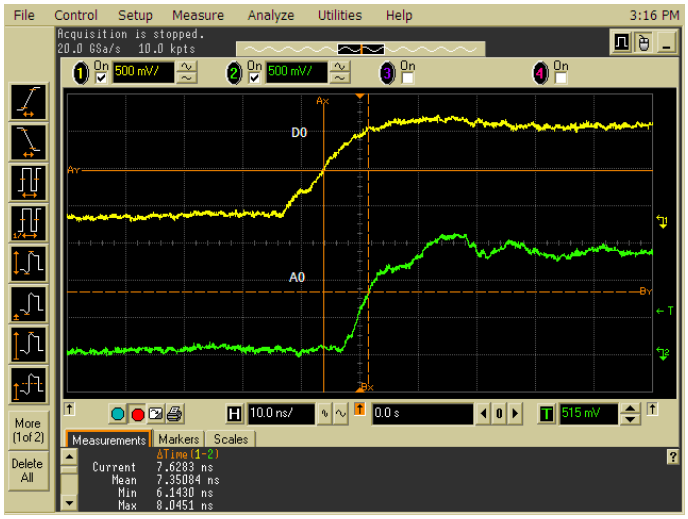


Figure A.5: Oscilloscope capture showing deadlock detection (loop size 8)

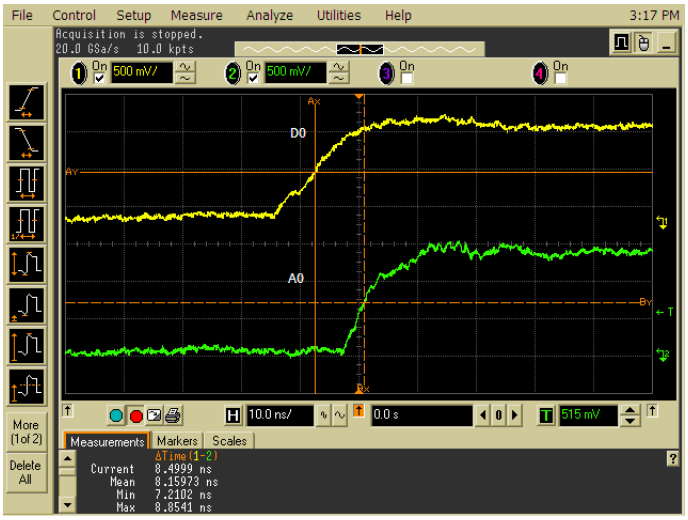


Figure A.6: Oscilloscope capture showing deadlock detection (loop size 10)

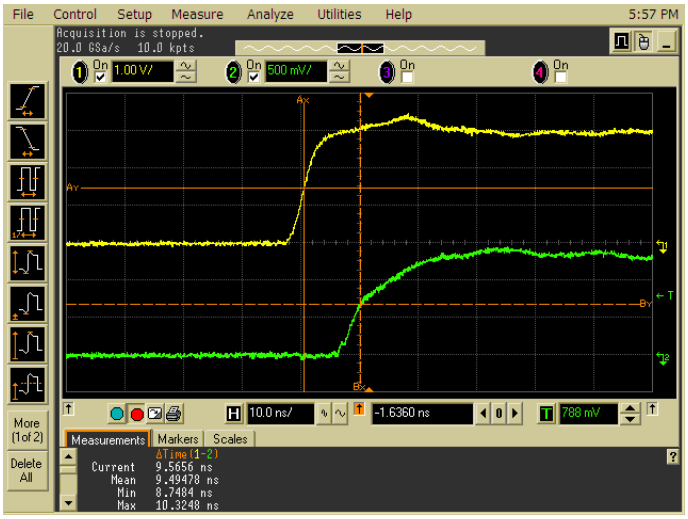


Figure A.7: Oscilloscope capture showing deadlock detection (loop size 12)

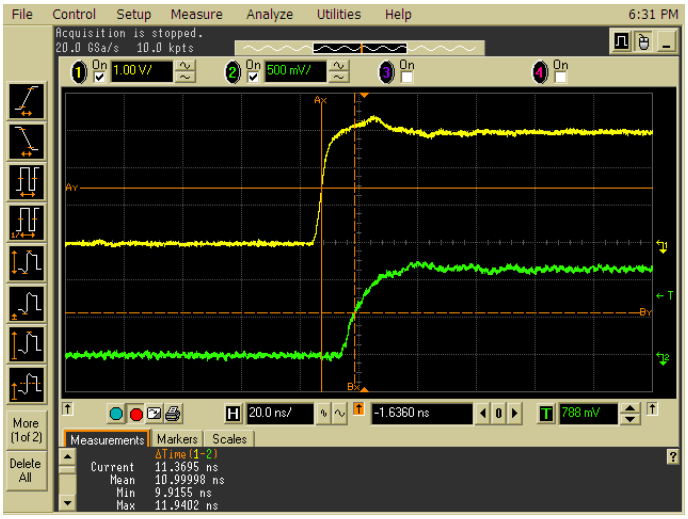


Figure A.8: Oscilloscope capture showing deadlock detection(loop size 14)

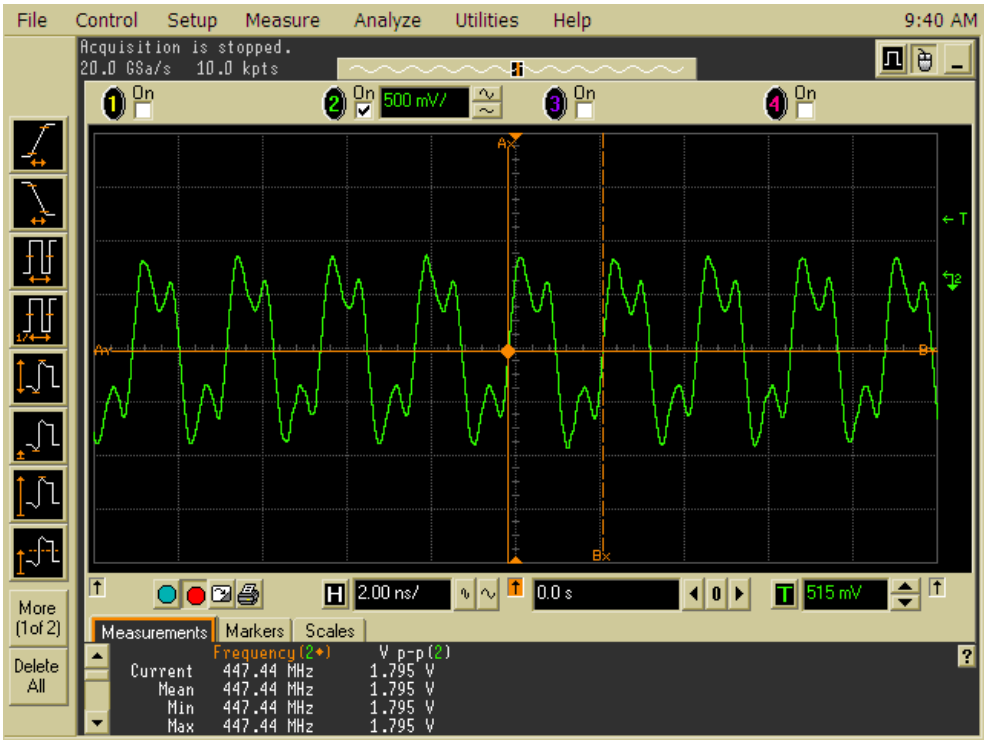


Figure A.9: Oscilloscope capture showing the output of the through layers (3D-IC) ring oscillator

B

THERMAL MATERIAL PARAMETERS

Table B.1 summarizes parameter values of the die layers and IC package layers (*e.g.,* material and dimensions) used in Chapter 5. These materials are fed to *HotSpot* thermal model with the power traces generated from NoC's simulator to estimate the temperature at different part of the chip.

IC layer/material	Thermal Conductivity (W/mK)	Thickness (m)	Width (m)	Width (m)
Heatsink/copper	400	0.0069	0.06	0.06
Heat spreader/copper	400	0.001	0.03	0.03
Thermal interface material	4	0.00002	0.01264	0.02172
Die –layer 0/silicon	100	0.00015	0.01264	0.02172
Die –layer 1/silicon	100	0.0001	0.01264	0.02172
Die –layer 2/silicon	100	0.0001	0.01264	0.02172
Metal layers (8)/copper	400	0.00001	0.01264	0.02172
Underfill	1.25	0.00001	0.01264	0.02172
C4 pads (8390)	2.5	0.00001	0.000001	0.000001
Chip carrier (ceramic)	2	0.001	0.021	0.021
Solder balls	16.6	0.00094	0.021	0.021
Printed circuit board	3	0.002	0.1	0.1

Table B.1: The material parameters used in this work for the thermal simulation

Part III

Thesis Bibliography

BIBLIOGRAPHY

[1] BookSim: Interconnection Network Simulator. URL http://nocs.stanford.edu/cgi-bin/trac.cgi/wiki/Resources/BookSim#. [March 20, 2012].

[2] CompSOC platform home page. URL http://compsoc.eu/. [April. 20, 2012].

[3] ITRS: International Technology Roadmap for Semiconductors, . URL http://www.itrs.net/. [March 26, 2012].

[4] ITRS 2011 Technology Working Group: Interconnect, . URL http://www.itrs.net/Links/2011ITRS/Home2011.htm. [May 10, 2012].

[5] MIT Lincoln Laboratory. URL http://www.ll.mit.edu.

[6] Nostrum NoC home page. URL http://www.ict.kth.se/nostrum/. [April. 20, 2012].

[7] PTM: Predictive Technology Model. URL http://ptm.asu.edu/. [Jan 05 2010].

[8] RAW processor publications home page. URL http://groups.csail.mit.edu/cag/raw/documents/. [Jan. 10, 2012].

[9] SpiNNaker home page. URL http://apt.cs.man.ac.uk/projects/SpiNNaker/. [Feb. 14, 2012].

[10] Tilera company products briefs. URL www.tilera.com/products/processors. [Dec. 10, 2011].

[11] Spartan-3A DSP starter platform user guide. URL http://www.xilinx.com/support/documentation/boards_and_kits/ug454_sp3a_dsp_start_ug.pdf. [May. 04, 2011].

[12] ns-2: A network simulator. URL http://www.isi.edu/nsnam/ns/. [April. 26, 2012].

[13] OPNET Modeler®: Network simulation. URL http://www.opnet.com/solutions/network_rd/modeler.html. [April 01, 2012].

[14] R. Al-Dujaily, T. Mak, Fei Xia, A. Yakovlev, and M. Palesi. Run-time deadlock detection in networks-on-chip using coupled transitive closure networks. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, pages 497–502, march 2011.

[15] R. Al-Dujaily, T. Mak, Kuan Zhou, Kai-Pui Lam, Yicong Meng, A. Yakovlev, and Chi-Sang Poon. On-chip dynamic programming networks using 3D-TSV integration. In *Embedded Computer Systems (SAMOS), 2011 International Conference on*, pages 318 – 325, july 2011. doi: 10.1109/SAMOS.2011.6045478.

[16] R. Al-Dujaily, N. Dahir, T. Mak, A Yakovlev, F. Xia, and C.S. Poon. Dynamic programming-based runtime thermal management (DPRTM): an on-line thermal control strategy for 3D-NoC systems. *(submited) ACM Transactions on Design Automation of Electronic Systems*, pages 1 –28, 2012.

[17] R. Al-Dujaily, T. Mak, K.P. Lam, A Yakovlev, F. Xia, and C.S. Poon. Dynamic thermal optimization in three dimensional networks-on-chip. *(in press) The Computer Journal*, pages 1 –14, 2012.

[18] R. Al-Dujaily, T. Mak, A Yakovlev, F. Xia, K.P. Lam, and C.S. Poon. Dynamic thermal optimization for 3D NoC. In *DATE2012 workshop in: Design, Automation and Test of 3D-ICs*, pages 1 –2, march 2012.

[19] Ra'ed Al-Dujaily, Terrence Mak, Fei Xia, Alex Yakovlev, and Maurizio Palesi. Embedded transitive closure network for runtime deadlock detection in networks-on-chip. *Parallel and Distributed Systems, IEEE Transactions on*, 23(7):1205 –1215, july 2012. ISSN 1045-9219. doi: 10.1109/TPDS.2011.275.

[20] S.M. Alam, R.E. Jones, S. Pozder, R. Chatterjee, S. Rauf, and A. Jain. Interstratum connection design considerations for cost-effective 3-D system integration. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 18(3):450 –460, march 2010. ISSN 1063-8210. doi: 10.1109/TVLSI.2008.2011910.

[21] A. Amory, A. de Mello, F. Moraes, and N. Calazans. ATLAS: an environment for NoC generation and evaluation. URL https://corfu.pucrs.br/redmine/projects/atlas/wiki. [April. 20, 2012].

[22] K.V. Anjan and T.M. Pinkston. DISHA: a deadlock recovery scheme for fully adaptive routing. In *Parallel Processing Symposium, 1995. Proceedings., 9th International*, pages 537 –543, apr 1995. doi: 10.1109/IPPS.1995.395983.

[23] K.V. Anjan, T.M. Pinkston, and J. Duato. Generalized theory for deadlock-free adaptive wormhole routing and its application to disha concurrent. In *Parallel Processing Symposium, 1996., Proceedings of IPPS '96, The 10th International*, pages 815 –821, apr 1996. doi: 10.1109/IPPS.1996.508153.

[24] J.K. Antonio, G.M. Huang, and W.K. Tsai. A fast distributed shortest path algorithm for a class of hierarchically clustered data networks. *Computers, IEEE Transactions on*, 41(6):710 –724, jun 1992. ISSN 0018-9340. doi: 10.1109/12.144623.

[25] G. Ascia, V. Catania, and M. Palesi. Multi-objective mapping for mesh-based NoC architectures. In *Hardware/Software Codesign and System Synthesis, 2004. CODES + ISSS 2004. International Conference on*, pages 182 – 187, sept. 2004. doi: 10.1109/CODESS.2004.241215.

[26] Giuseppe Ascia, Vincenzo Catania, Maurizio Palesi, and Davide Patti. Implementation and analysis of a new selection strategy for adaptive routing in networks-on-chip. *Computers, IEEE Transactions on*, 57(6):809 –820, june 2008. ISSN 0018-9340. doi: 10.1109/TC.2008.38.

[27] S. Assefa, F. Xia, W.M.J. Green, C.L. Schow, A.V. Rylyakov, and Y.A. Vlasov. CMOS-integrated optical receivers for on-chip interconnects. *Selected Topics in Quantum Electronics, IEEE Journal of*, 16(5):1376 –1385, sept.-oct. 2010. ISSN 1077-260X. doi: 10.1109/JSTQE.2010.2048306.

[28] Arnab Banerjee, Pascal T. Wolkotte, Robert D. Mullins, Simon W. Moore, and Gerard J. M. Smit. An energy and performance exploration of network-on-chip architectures. *IEEE Trans. Very Large Scale Integr. Syst.*, 17(3):319–329, March 2009. ISSN 1063-8210. doi: 10.1109/TVLSI.2008.2011232. URL http://dx.doi.org/10.1109/TVLSI.2008.2011232.

[29] R.G. Beausoleil, P.J. Kuekes, G.S. Snider, Shih-Yuan Wang, and R.S. Williams. Nanoelectronic and nanophotonic interconnect. *Proceedings of the IEEE*, 96(2):230 –247, feb. 2008. ISSN 0018-9219. doi: 10.1109/JPROC.2007.911057.

[30] R. Bellman. *Adaptive Control Processes*. Princeton University Press, 1961.

[31] R. Bellman and R. Kalaba. On the role of dynamic programming in statistical communication theory. *Information Theory, IRE Transactions on*, 3(3):197 –203, september 1957. ISSN 0096-1000. doi: 10.1109/TIT.1957.1057416.

[32] R.E. Bellman. *Dynamic Programming*. Dover Books on Mathematics. Dover Publications, 2003. ISBN 9780486428093. URL http://books.google.co.uk/books?id=fyVtp3EMxasC.

[33] L. Benini and G. De Micheli. Networks on chips: a new SoC paradigm. *IEEE Computer*, 35(1):70 –78, 2002. ISSN 0018-9162. doi: 10.1109/2.976921.

[34] D. Bertozzi and L. Benini. Xpipes: a network-on-chip architecture for gigascale systems-on-chip. *Circuits and Systems Magazine, IEEE*, 4(2):18 – 31, 2004. ISSN 1531-636X. doi: 10.1109/MCAS.2004.1330747.

[35] D. Bertozzi, A. Jalabert, Srinivasan Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. De Micheli. Noc synthesis flow for customized domain specific multiprocessor systems-on-chip. *Parallel and Distributed Systems, IEEE Transactions on*, 16(2):113 – 129, feb. 2005. ISSN 1045-9219. doi: 10.1109/TPDS.2005.22.

[36] D. Bertozzi, S. Kumar, and M. Palesi. *Networks-on-Chip*. Hindawi Publishing Corporation, 2007. ISBN 9789775945907. URL http://books.google.co.uk/books?id=19mvNQAACAAJ.

[37] D. Bertsekas. Distributed dynamic programming. *Automatic Control, IEEE Transactions on*, 27(3):610 – 616, jun 1982. ISSN 0018-9286. doi: 10.1109/TAC.1982.1102980.

[38] D.P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 2005. ISBN 1-886529-26-4.

[39] T. Bjerregaard and J. Sparso. A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip. In *Design, Automation and Test in Europe, 2005. Proceedings*, pages 1226 – 1231 Vol. 2, march 2005. doi: 10.1109/DATE.2005.36.

[40] Tobias Bjerregaard and Shankar Mahadevan. A survey of research and practices of network-on-chip. *ACM Comput. Surv.*, 38(1), June 2006. ISSN 0360-0300. doi: http://doi.acm.org/10.1145/1132952.1132953. URL http://doi.acm.org/http://doi.acm.org/10.1145/1132952.1132953.

[41] Bryan Black, Murali Annavaram, Ned Brekelbaum, John DeVale, Lei Jiang, Gabriel H. Loh, Don McCaule, Pat Morrow, Donald W. Nelson, Daniel Pantuso, Paul Reed, Jeff Rupley, Sadasivan Shankar, John Shen, and Clair Webb. Die stacking (3D) microarchitecture. In *Microarchitecture, 2006. MICRO-39. 39th Annual IEEE/ACM International Symposium on*, pages 469 –479, dec. 2006. doi: 10.1109/MICRO.2006.18.

[42] J.F. Cardo, J. Flich, and D. Bertozzi. *Designing Network On-Chip Architectures in the Nanoscale Era*. Chapman & Hall/CRC computational science series. Taylor and Francis, 2010. ISBN 9781439837108. URL http://books.google.co.uk/books?id=epndQgAACAAJ.

[43] M.F. Chang, V.P. Roychowdhury, Liyang Zhang, Hyunchol Shin, and Yongxi Qian. RF/wireless interconnect for inter- and intra-

chip communications. *Proceedings of the IEEE*, 89(4):456 –466, apr 2001. ISSN 0018-9219. doi: 10.1109/5.920578.

[44] M.F. Chang, J. Cong, A. Kaplan, M. Naik, G. Reinman, E. Socher, and S.-W. Tam. CMP network-on-chip overlaid with multi-band RF-interconnect. In *High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on*, pages 191 –202, feb. 2008. doi: 10.1109/HPCA.2008.4658639.

[45] Chih-Hao Chao, Kai-Yuan Jheng, Hao-Yu Wang, Jia-Cheng Wu, and An-Yeu Wu. Traffic- and thermal-aware run-time thermal management scheme for 3D NoC systems. In *Networks-on-Chip (NOCS), 2010 Fourth ACM/IEEE International Symposium on*, pages 223 –230, may 2010. doi: 10.1109/NOCS.2010.32.

[46] A.A. Chein. A cost and speed model for k-ary n-cube wormhole routers. *Parallel and Distributed Systems, IEEE Transactions on*, 9 (2):150 –162, feb 1998. ISSN 1045-9219. doi: 10.1109/71.663877.

[47] Ge-Ming Chiu. The odd-even turn model for adaptive routing. *Parallel and Distributed Systems, IEEE Transactions on*, 11(7):729 –738, jul 2000. ISSN 1045-9219. doi: 10.1109/71.877831.

[48] Ching-Che Chung and Cheng-Ruei Yang. An autocalibrated all-digital temperature sensor for on-chip thermal monitoring. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 58(2): 105 –109, feb. 2011. ISSN 1549-7747. doi: 10.1109/TCSII.2010.2104016.

[49] Leiserson C. E. Cormen, T. H. and R. L. Rivest. *Introduction to Algorithms*. MIT Press and McGraw-Hill Publishers, USA, 2001.

[50] A.K. Coskun, J.L. Ayala, D. Atienza, and T.S. Rosing. Modeling and dynamic management of 3D multicore systems with liquid cooling. In *Very Large Scale Integration (VLSI-SoC), 2009 17th IFIP International Conference on*, pages 35 –40, oct. 2009. doi: 10.1109/VLSISOC.2009.6041327.

[51] M. Dall'Osso, G. Biccari, L. Giovannini, D. Bertozzi, and L. Benini. Xpipes: a latency insensitive parameterized network-on-chip architecture for multiprocessor SoCs. In *Computer Design, 2003. Proceedings. 21st International Conference on*, pages 536 – 539, oct. 2003. doi: 10.1109/ICCD.2003.1240952.

[52] W. J. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers, USA, 2004.

[53] W.J. Dally. Virtual-channel flow control. *Parallel and Distributed Systems, IEEE Transactions on*, 3(2):194 –205, mar 1992. ISSN 1045-9219. doi: 10.1109/71.127260.

[54] W.J. Dally and C.L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *Computers, IEEE Transactions on*, C-36(5):547 –553, may 1987. ISSN 0018-9340. doi: 10.1109/TC.1987.1676939.

[55] W.J. Dally and B. Towles. Route packets, not wires: on-chip interconnection networks. In *Design Automation Conference, 2001. Proceedings*, pages 684 – 689, 2001. doi: 10.1109/DAC.2001.156225.

[56] W.J. Dally, L.R. Dennison, D. Harris, Kinhong Kan, and T. Xanthopoulos. Architecture and implementation of the reliable router. In *Hot Interconnects II, 1994. Symposium Record*, pages 197 –208, 1994. doi: 10.1109/CONECT.1994.765349.

[57] F. Darve, A. Sheibanyrad, P. Vivet, and F. Petrot. Physical implementation of an asynchronous 3D-NoC router using serial vertical links. In *VLSI (ISVLSI), 2011 IEEE Computer Society Annual Symposium on*, pages 25 –30, july 2011. doi: 10.1109/ISVLSI.2011.59.

[58] W.R. Davis, J. Wilson, S. Mick, J. Xu, H. Hua, C. Mineo, A.M. Sule, M. Steer, and P.D. Franzon. Demystifying 3D ICs: the pros and cons of going vertical. *Design Test of Computers, IEEE*, 22(6): 498 – 510, nov.-dec. 2005. ISSN 0740-7475. doi: 10.1109/MDT.2005.136.

[59] S. Dighe, S.R. Vangal, P. Aseron, S. Kumar, T. Jacob, K.A. Bowman, J. Howard, J. Tschanz, V. Erraguntla, N. Borkar, V.K. De, and S. Borkar. Within-die variation-aware dynamic-voltage-frequency-scaling with optimal core allocation and thread hopping for the 80-core teraflops processor. *Solid-State Circuits, IEEE Journal of*, 46(1):184 –193, jan. 2011. ISSN 0018-9200. doi: 10.1109/JSSC.2010.2080550.

[60] J. Donald and M. Martonosi. Techniques for multicore thermal management: Classification and new exploration. In *Computer Architecture, 2006. ISCA '06. 33rd International Symposium on*, pages 78 –88, 0-0 2006. doi: 10.1109/ISCA.2006.39.

[61] Yalamanchili S. Duato, J. and L. M. Ni. *Interconnection Networks: An Engineering Approach*. Morgan Kaufmann Publishers, USA, 2004.

[62] F. Fazzino, M. Palesi, and D. Patti. Noxim: Network-on-chip simulator. http://noxim.sourceforge.net/.

[63] B.S. Feero and P.P. Pande. Networks-on-chip in a three-dimensional environment: a performance evaluation. *Computers, IEEE Transactions on*, 58(1):32 –45, jan. 2009. ISSN 0018-9340. doi: 10.1109/TC.2008.142.

[64] M. Freimer. A dynamic programming approach to adaptive control processes. *Automatic Control, IRE Transactions on*, 4(2):10 – 15, nov 1959. ISSN 0096-199X. doi: 10.1109/TAC.1959.1104848.

[65] C.J. Glass and L.M. Ni. The turn model for adaptive routing. In *Computer Architecture, 1992. Proceedings., The 19th Annual International Symposium on*, pages 278 –287, 1992. doi: 10.1109/ISCA.1992.753324.

[66] M.T. Goodrich and R. Tamassia. *Algorithm design: foundations, analysis, and Internet examples*. Wiley, 2002. ISBN 9780471383659. URL http://books.google.co.uk/books?id=gD8kAQAAIAAJ.

[67] K. Goossens and A. Hansson. The Æthereal network on chip after ten years: Goals, evolution, lessons, and future. In *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, pages 306 –311, june 2010.

[68] K. Goossens, J. Dielissen, and A. Radulescu. Æthereal network on chip: concepts, architectures, and implementations. *Design Test of Computers, IEEE*, 22(5):414 – 421, sept.-oct. 2005. ISSN 0740-7475. doi: 10.1109/MDT.2005.99.

[69] K. Goossens, B. Vermeulen, and A.B. Nejad. A high-level debug environment for communication-centric debug. In *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, pages 202 –207, april 2009.

[70] P. Guerrier and A. Greiner. A generic architecture for on-chip packet-switched interconnections. In *Design, Automation and Test in Europe Conference and Exhibition 2000. Proceedings*, pages 250 –256, 2000. doi: 10.1109/DATE.2000.840047.

[71] N. E. Guindi and P. Elsener. Network on chip: PANACEA - a NOSTRUM integration, 2005. URL http://www.ict.kth.se/nostrum/.

[72] Andreas Hansson, Kees Goossens, Marco Bekooij, and Jos Huisken. CoMPSoC: A template for composable and predictable multi-processor system on chips. *ACM Trans. Des. Autom. Electron. Syst.*, 14(1):2:1–2:24, January 2009. ISSN 1084-4309. doi: 10.1145/1455229.1455231. URL http://doi.acm.org/10.1145/1455229.1455231.

[73] M. Haurylau, Hui Chen, Jidong Zhang, Guoqing Chen, N.A. Nelson, D.H. Albonesi, E.G. Friedman, and P.M. Fauchet. On-chip optical interconnect roadmap: challenges and critical directions. In *Group IV Photonics, 2005. 2nd IEEE International Conference on*, pages 17 – 19, sept. 2005. doi: 10.1109/GROUP4.2005.1516388.

[74] Jun He, Chen Zhao, Sheng-Huang Lee, K. Peterson, R. Geiger, and Degang Chen. Highly linear very compact untrimmed on-chip temperature sensor with second and third order temperature compensation. In *Circuits and Systems (MWSCAS), 2010 53rd IEEE International Midwest Symposium on*, pages 288 –291, aug. 2010. doi: 10.1109/MWSCAS.2010.5548802.

[75] F. Hillier and G. Lieberman. *Introduction to Operations Research*. McGraw-Hill International Editions, 1995.

[76] R. Ho, K.W. Mai, and M.A. Horowitz. The future of wires. *Proceedings of the IEEE*, 89(4):490 –504, apr 2001. ISSN 0018-9219. doi: 10.1109/5.920580.

[77] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar. A 5-GHz mesh interconnect for a teraflops processor. *Micro, IEEE*, 27(5):51 –61, sept.-oct. 2007. ISSN 0272-1732. doi: 10.1109/MM.2007.4378783.

[78] J. Howard, S. Dighe, S.R. Vangal, G. Ruhl, N. Borkar, S. Jain, V. Erraguntla, M. Konow, M. Riepen, M. Gries, G. Droege, T. Lund-Larsen, S. Steibl, S. Borkar, V.K. De, and R. Van Der Wijngaart. A 48-core IA-32 processor in 45 nm CMOS using on-die message-passing and DVFS for performance and power scaling. *Solid-State Circuits, IEEE Journal of*, 46(1):173 –183, jan. 2011. ISSN 0018-9200. doi: 10.1109/JSSC.2010.2079450.

[79] J. Hu and R. Marculescu. Worm-sim simulator: a cycle accurate simulator for networks-on-chip. URL `http://www.ece.cmu.edu/~sld/wiki/doku.php?id=shared:worm_sim`. [May 2, 2012].

[80] Jingcao Hu and R. Marculescu. Energy- and performance-aware mapping for regular noc architectures. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 24(4):551 – 562, april 2005. ISSN 0278-0070. doi: 10.1109/TCAD.2005.844106.

[81] Jingcao Hu and Radu Marculescu. DyAD - smart routing for networks-on-chip. In *In ACM/IEEE Design Automation Conference*, pages 260–263, 2004.

[82] Wei Huang, M.R. Stan, K. Skadron, K. Sankaranarayanan, S. Ghosh, and S. Velusamy. Compact thermal modeling for temperature-aware design. In *Design Automation Conference, 2004. Proceedings. 41st*, pages 878 –883, july 2004.

[83] Wei Huang, M.R. Stan, and K. Skadron. Parameterized physical compact thermal modeling. *Components and Packaging Technologies, IEEE Transactions on*, 28(4):615 – 622, dec. 2005. ISSN 1521-3331. doi: 10.1109/TCAPT.2005.859737.

[84] Wei Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M.R. Stan. HotSpot: a compact thermal modeling methodology for early-stage VLSI design. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 14(5):501 –513, may 2006. ISSN 1063-8210. doi: 10.1109/TVLSI.2006.876103.

[85] Wei Huang, K. Sankaranarayanan, K. Skadron, R.J. Ribando, and M.R. Stan. Accurate, pre-RTL temperature-aware design using a parameterized, geometric thermal model. *Computers, IEEE Transactions on*, 57(9):1277 –1288, sept. 2008. ISSN 0018-9340. doi: 10.1109/TC.2008.64.

[86] Wei Huang, K. Skadron, S. Gurumurthi, R.J. Ribando, and M.R. Stan. Differentiating the roles of IR measurement and simulation for power and temperature-aware design. In *Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on*, pages 1 –10, april 2009. doi: 10.1109/ISPASS.2009.4919633.

[87] Intel. Many Integrated Core (MIC) Architecture. URL http://www.intel.com/content/www/us/en/architecture-and-technology/many-integrated-core/intel-many-integrated-core-architecture.html?wapkw=mic. [March. 12, 2012].

[88] The Intel® Tera-scale Computing Research Program. Intel: The 80-core Tera-scale Research Chip. URL http://techresearch.intel.com/ProjectDetails.aspx?Id=151. [March. 2, 2012].

[89] L. Jain, B. Al-Hashimi, M Zwolinski, M. Gaur, P. Rosinger, and V. Laxmi. MIRGAM: a simulator for NoC interconnect routing and application modeling. URL http://nirgam.ecs.soton.ac.uk/. [April 26, 2012].

[90] A. Jantsch and H. Tenhunen. *Networks on chip*. Kluwer Academic Publishers, 2003. ISBN 9781402073922. URL http://books.google.co.uk/books?id=28hyoFE-OpsC.

[91] A. Jantsch, R. Lauter, and A. Vitkowski. Power analysis of link level and end-to-end data protection in networks on chip. In *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, pages 1770 – 1773 Vol. 2, may 2005. doi: 10.1109/ISCAS.2005.1464951.

[92] N.E. Jerger and L.S. Peh. *On-Chip Networks*. Synthesis lectures in computer architecture. Morgan & Claypool Publishers, 2009. ISBN 9781598295849. URL http://books.google.co.uk/books?id=Nf9q6goSpssC.

[93] James W. Joyner, Raguraman Venkatesan, Payman Zarkesh-Ha, Jeffrey A. Davis, and James D. Meindl. Impact of three-dimensional architectures on interconnects in gigascale integration. *IEEE Trans. Very Large Scale Integr. Syst.*, 9(6):922–928, December 2001. ISSN 1063-8210. doi: 10.1109/92.974905. URL http://dx.doi.org/10.1109/92.974905.

[94] A. B. Kahng, B. Li, L.-S. Peh, and K. Samadi. ORION 2.0: a power-area simulator for interconnection networks. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, PP(99):1 –5, 2011. ISSN 1063-8210. doi: 10.1109/TVLSI.2010.2091686.

[95] N.H. Khan, S.M. Alam, and S. Hassoun. Power delivery design for 3-D ICs using different through-silicon via (TSV) technologies. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 19(4):647 –658, april 2011. ISSN 1063-8210. doi: 10.1109/TVLSI.2009.2038165.

[96] J.H. Kim, Ziqiang Liu, and A.A. Chien. Compressionless routing: a framework for adaptive and fault-tolerant routing. *Parallel and Distributed Systems, IEEE Transactions on*, 8(3):229 –244, mar 1997. ISSN 1045-9219. doi: 10.1109/71.584089.

[97] D. J. Kinniment. *Synchronization and Arbitration in Digital Systems*. Wiley Publishing, 2008.

[98] S. Konstantinidou and L. Snyder. The Chaos router. *Computers, IEEE Transactions on*, 43(12):1386 –1397, dec 1994. ISSN 0018-9340. doi: 10.1109/12.338098.

[99] S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani. A network on chip architecture and design methodology. In *VLSI, 2002. Proceedings. IEEE Computer Society Annual Symposium on*, pages 105 –112, 2002. doi: 10.1109/ISVLSI.2002.1016885.

[100] Sun-Yuan Kung, Sheng-Chun Lo, and P.S. Lewis. Optimal systolic design for the transitive closure and the shortest path problems. *Computers, IEEE Transactions on*, C-36(5):603 –614, may 1987. ISSN 0018-9340. doi: 10.1109/TC.1987.1676945.

[101] K.P. Lam and C.W. Tong. Closed semiring connectionist network for the Bellman-Ford computation. *Computers and Digital Techniques, IEE Proceedings -*, 143(3):189 –195, may 1996. ISSN 1350-2387. doi: 10.1049/ip-cdt:19960379.

[102] A. Lankes, T. Wild, A. Herkersdorf, S. Sonntag, and H. Reinig. Comparison of deadlock recovery and avoidance mechanisms to approach message dependent deadlocks in on-chip networks. In *Networks-on-Chip (NOCS), 2010 Fourth ACM/IEEE International*

Symposium on, pages 17 –24, may 2010. doi: 10.1109/NOCS.2010.11.

[103] Kangmin Lee, Se-Joong Lee, and Hoi-Jun Yoo. Low-power network-on-chip for high-performance soc design. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 14(2):148 –160, feb. 2006. ISSN 1063-8210. doi: 10.1109/TVLSI.2005.863753.

[104] Soojung Lee. Turn-based deadlock detection for wormhole routed networks. In *Computer and Information Technology, 2006. CIT '06. The Sixth IEEE International Conference on*, page 107, sept. 2006. doi: 10.1109/CIT.2006.188.

[105] Soojung Lee. A deadlock detection mechanism for true fully adaptive routing in regular wormhole networks. *Comput. Commun.*, 30:1826–1840, June 2007. ISSN 0140-3664. doi: 10.1016/j.comcom.2007.02.013. URL http://portal.acm.org/citation.cfm?id=1242852.1243197.

[106] D.L. Lewis, S. Yalamanchili, and H.-H.S. Lee. High performance non-blocking switch design in 3D die-stacking technology. In *VLSI, 2009. ISVLSI '09. IEEE Computer Society Annual Symposium on*, pages 25 –30, may 2009. doi: 10.1109/ISVLSI.2009.53.

[107] Ming Li, Qing-An Zeng, and Wen-Ben Jone. DyXY - a proximity congestion-aware deadlock-free dynamic routing method for network on chip. In *Design Automation Conference, 2006 43rd ACM/IEEE*, pages 849 –852, 0-0 2006. doi: 10.1109/DAC.2006.229242.

[108] Zheng Li, D. Fay, A. Mickelson, Li Shang, M. Vachharajani, D. Filipovic, Wounjhang Park, and Yihe Sun. Spectrum: A hybrid nanophotonic-electric on-chip network. In *Design Automation Conference, 2009. DAC '09. 46th ACM/IEEE*, pages 575 –580, july 2009.

[109] Shu-Yen Lin, Tzu-Chu Yin, Hao-Yu Wang, and An-Yeu Wu. Traffic-and thermal-aware routing for throttled three-dimensional network-on-chip systems. In *VLSI Design, Automation and Test (VLSI-DAT), 2011 International Symposium on*, pages 1 –4, april 2011. doi: 10.1109/VDAT.2011.5783639.

[110] Yanbin Lin and Dong Xiang. An effective congestion-aware selection function for adaptive routing in interconnection networks. In *Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2010 International Conference on*, pages 156 –165, dec. 2010. doi: 10.1109/PDCAT.2010.42.

[111] I. Loi, S. Mitra, T.H. Lee, S. Fujita, and L. Benini. A low-overhead fault tolerance scheme for TSV-based 3D network

on chip links. In *Computer-Aided Design, 2008. ICCAD 2008. IEEE/ACM International Conference on*, pages 598 –602, nov. 2008. doi: 10.1109/ICCAD.2008.4681638.

[112] P. Lopez, J.M. Martinez, and J. Duato. A very efficient distributed deadlock detection mechanism for wormhole networks. In *High-Performance Computer Architecture, 1998. Proceedings., 1998 Fourth International Symposium on*, pages 57 –66, feb 1998. doi: 10.1109/HPCA.1998.650546.

[113] Chiao-Ling Lung, Yi-Lun Ho, Ding-Ming Kwai, and Shih-Chieh Chang. Thermal-aware on-line task allocation for 3D multi-core processor throughput optimization. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, pages 1 –6, march 2011.

[114] T. Mak, Kai-Pui Lam, H.S. Ng, G. Rachmuth, and Chi-Sang Poon. A CMOS current-mode dynamic programming circuit. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 57(12):3112 –3123, dec. 2010. ISSN 1549-8328. doi: 10.1109/TCSI.2010.2052661.

[115] T. Mak, R. Al-Dujaily, K. Zhou, K. Lam, and C. Poon. On-chip dynamic programming network using TSV-based 3D stacking technology. In *Conf. GOMACTech, Orlando, USA*, 2011.

[116] T. Mak, R. Al-Dujaily, Kuan Zhou, Kai-Pui Lam, Yicong Meng, A. Yakovlev, and Chi-Sang Poon. Dynamic programming networks for large-scale 3D chip integration. *Circuits and Systems Magazine, IEEE*, 11(3):51 –62, thirdquarter 2011. ISSN 1531-636X. doi: 10.1109/MCAS.2011.942102.

[117] T. Mak, P.Y.K. Cheung, Kai-Pui Lam, and W. Luk. Adaptive routing in network-on-chips using a dynamic-programming network. *Industrial Electronics, IEEE Transactions on*, 58(8):3701 –3716, aug. 2011. ISSN 0278-0046. doi: 10.1109/TIE.2010.2081953.

[118] Terrence Mak, Peter Y.K. Cheung, Wayne Luk, and Kai Pui Lam. A DP-network for optimal dynamic routing in network-on-chip. In *Proceedings of the 7th IEEE/ACM international conference on Hardware/software codesign and system synthesis*, CODES+ISSS '09, pages 119–128, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-628-1. doi: http://doi.acm.org/10.1145/1629435.1629452. URL http://doi.acm.org/10.1145/1629435.1629452.

[119] T.S.T. Mak, K.P. Lam, H.S. Ng, G. Rachmuth, and C.-S. Poon. A current-mode analog circuit for reinforcement learning problems. In *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, pages 1301 –1304, may 2007. doi: 10.1109/ISCAS.2007.378410.

[120] T.S.T. Mak, P. Sedcole, P.Y.K. Cheung, W. Luk, and K.R. Lam. A hybrid analog-digital routing network for NoC dynamic routing. In *Networks-on-Chip, 2007. NOCS 2007. First International Symposium on*, pages 173 –182, may 2007. doi: 10.1109/NOCS.2007.2.

[121] R. Marculescu, Jingcao Hu, and U.Y. Ogras. Key research problems in NoC design: a holistic perspective. In *Hardware/Software Codesign and System Synthesis, 2005. CODES+ISSS '05. Third IEEE/ACM/IFIP International Conference on*, pages 69 –74, sept. 2005. doi: 10.1145/1084834.1084856.

[122] Radu Marculescu. Toward a science for future noc design. In *Proceedings of the 2nd International Workshop on Network on Chip Architectures*, NoCArc '09, pages 1–1, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-774-5. doi: 10.1145/1645213.1645215. URL http://doi.acm.org/10.1145/1645213.1645215.

[123] J.M. Martinez, P. Lopez, J. Duato, and T.M. Pinkston. Software-based deadlock recovery technique for true fully adaptive routing in wormhole networks. In *Parallel Processing, 1997., Proceedings of the 1997 International Conference on*, pages 182 –189, aug 1997. doi: 10.1109/ICPP.1997.622586.

[124] J.M. Martinez-Rubio, P. Lopez, and J. Duato. A cost-effective approach to deadlock handling in wormhole networks. *Parallel and Distributed Systems, IEEE Transactions on*, 12(7):716 –729, jul 2001. ISSN 1045-9219. doi: 10.1109/71.940746.

[125] Gordon E. Moore. Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114 ff. *Solid-State Circuits Newsletter, IEEE*, 20(3):33 –35, sept. 2006. ISSN 1098-4232. doi: 10.1109/N-SSC.2006.4785860.

[126] Simon Moore and Daniel Greenfield. The next resource war: computation vs. communication. In *Proceedings of the 2008 international workshop on System level interconnect prediction*, SLIP '08, pages 81–86, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-918-0. doi: 10.1145/1353610.1353627. URL http://doi.acm.org/10.1145/1353610.1353627.

[127] Fernando Moraes, Ney Calazans, Aline Mello, Leandro Möller, and Luciano Ost. Hermes: an infrastructure for low area overhead packet-switching networks on chip. *Integration, the VLSI Journal*, 38(1):69 – 93, 2004. ISSN 0167-9260. doi: 10.1016/j.vlsi.2004.03.003. URL http://www.sciencedirect.com/science/article/pii/S0167926004000185.

[128] R. Mullins. Netmaker: Interconnection Network Simulator. URL http://www-dyn.cl.cam.ac.uk/~rdm34/wiki/index.php?title=Main_Page. [April 26, 2012].

[129] L.M. Ni and P.K. McKinley. A survey of wormhole routing techniques in direct networks. *Computer*, 26(2):62 –76, feb 1993. ISSN 0018-9162. doi: 10.1109/2.191995.

[130] C. Nicopoulos, V. Narayanan, and C.R. Das. *Network-on-Chip Architectures: A Holistic Design Exploration*. Lecture Notes in Electrical Engineering. Springer, 2009. ISBN 9789048130306. URL http://books.google.co.uk/books?id=5Oi4d5TGdCwC.

[131] K. Ohashi, K. Nishi, T. Shimizu, M. Nakada, J. Fujikata, J. Ushida, S. Torii, K. Nose, M. Mizuno, H. Yukawa, M. Kinoshita, N. Suzuki, A. Gomyo, T. Ishi, D. Okamoto, K. Furue, T. Ueno, T. Tsuchizawa, T. Watanabe, K. Yamada, S.-i. Itabashi, and J. Akedo. On-chip optical interconnect. *Proceedings of the IEEE*, 97(7):1186 –1198, july 2009. ISSN 0018-9219. doi: 10.1109/JPROC.2009.2020331.

[132] M. Palesi, R. Holsmark, and S. Kumar. A methodology for design of application specific deadlock-free routing algorithms for noc systems. In *Hardware/Software Codesign and System Synthesis, 2006. CODES+ISSS '06. Proceedings of the 4th International Conference*, pages 142 –147, oct. 2006. doi: 10.1145/1176254.1176289.

[133] Partha Pratim Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh. Performance evaluation and design trade-offs for network-on-chip interconnect architectures. *Computers, IEEE Transactions on*, 54(8):1025 – 1040, aug. 2005. ISSN 0018-9340. doi: 10.1109/TC.2005.134.

[134] P.P. Pande, C. Grecu, A. Ivanov, and R. Saleh. High-throughput switch-based interconnect for future SoCs. In *System-on-Chip for Real-Time Applications, 2003. Proceedings. The 3rd IEEE International Workshop on*, pages 304 – 310, june-2 july 2003. doi: 10.1109/IWSOC.2003.1213053.

[135] V.F. Pavlidis and E.G. Friedman. 3-D topologies for networks-on-chip. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 15(10):1081 –1090, oct. 2007. ISSN 1063-8210. doi: 10.1109/TVLSI.2007.893649.

[136] Vijayakumar S. Peters J. and Schaal S. Reinforcement learning for humanoid robotics. In *3rd IEEE-RAS Int. Conf. Humanoid Robotics*, pages 1 –20, Sept. 2003.

[137] T.M. Pinkston and S. Warnakulasuriya. Characterization of deadlocks in k-ary n-cube networks. *Parallel and Distributed Systems, IEEE Transactions on*, 10(9):904 –921, sep 1999. ISSN 1045-9219. doi: 10.1109/71.798315.

[138] Luis A. Plana, David Clark, Simon Davidson, Steve Furber, Jim Garside, Eustace Painkras, Jeffrey Pepper, Steve Temple, and John Bainbridge. Spinnaker: Design and implementation of a gals multicore system-on-chip. *J. Emerg. Technol. Comput. Syst.*, 7(4):17:1–17:18, December 2011. ISSN 1550-4832. doi: 10.1145/2043643.2043647. URL http://doi.acm.org/10.1145/2043643.2043647.

[139] J. M. Rabaey, A. Chandrakasan, and B. Nikolic. *Digital Integrated Circuits: A Design Perspective*. Upper Saddle River, N.J.; London: Prentice Hall, 2002.

[140] J.M.M. Rubio, P. Lopez, and J. Duato. FC3D: flow control-based distributed deadlock detection mechanism for true fully adaptive routing in wormhole networks. *Parallel and Distributed Systems, IEEE Transactions on*, 14(8):765 – 779, aug. 2003. ISSN 1045-9219. doi: 10.1109/TPDS.2003.1225056.

[141] P. Salihundam, S. Jain, T. Jacob, S. Kumar, V. Erraguntla, Y. Hoskote, S. Vangal, G. Ruhl, and N. Borkar. A 2 Tb/s 64 mesh network for a single-chip cloud computer with DVFS in 45 nm CMOS. *Solid-State Circuits, IEEE Journal of*, 46(4):757 –766, april 2011. ISSN 0018-9200. doi: 10.1109/JSSC.2011.2108121.

[142] D. Sarkar and A. Mukherjee. Design of optimal systolic algorithms for the transitive closure problem. *Computers, IEEE Transactions on*, 41(4):508 –512, apr 1992. ISSN 0018-9340. doi: 10.1109/12.135564.

[143] The Intel® Tera scale Computing Research Program. Intel: Single-chip cloud computer. URL http://techresearch.intel.com/ProjectDetails.aspx?Id=1. [March. 2, 2012].

[144] L.W. Schaper, S.L. Burkett, S. Spiesshoefer, G.V. Vangara, Z. Rahman, and S. Polamreddy. Architectural implications and process development of 3-D VLSI -axis interconnects using through silicon vias. *Advanced Packaging, IEEE Transactions on*, 28(3):356 – 366, aug. 2005. ISSN 1521-3323. doi: 10.1109/TADVP.2005.853271.

[145] Wolfram Schultz, Peter Dayan, and P. Read Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, 1997. doi: 10.1126/science.275.5306.1593. URL http://www.sciencemag.org/content/275/5306/1593.abstract.

[146] C. Seiculescu, S. Murali, L. Benini, and G. De Micheli. A method to remove deadlocks in networks-on-chips with wormhole flow control. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, pages 1625 –1628, march 2010.

[147] C. L. Seitz, W. C. Athas, C. M. Flaig, A. J. Martin, J. Seizovic, C. S. Steele, and W-K. Su. The architecture and programming of the Ametek series 2010 multicomputer. In *Proceedings of the third conference on Hypercube concurrent computers and applications: Architecture, software, computer systems, and general issues - Volume 1*, C3P, pages 33–37, New York, NY, USA, 1988. ACM. ISBN 0-89791-278-0. doi: 10.1145/62297.62302. URL http://doi.acm.org/10.1145/62297.62302.

[148] Li Shang, L. Peh, A. Kumar, and N.K. Jha. Thermal modeling, characterization and management of on-chip networks. In *Microarchitecture, 2004. MICRO-37 2004. 37th International Symposium on*, pages 67 – 78, dec. 2004. doi: 10.1109/MICRO.2004.35.

[149] A. Sheibanyrad, F. Pétrot, and A. Janstch. *3D Integration for NoC-Based SoC Architectures*. Integrated Circuits and Systems. Springer, 2010. ISBN 9781441976178. URL http://books.google.co.uk/books?id=l7A8F0x9uIUC.

[150] Yen-Hao Shih, Shian-Ru Lin, Tsung-Miau Wang, and Jenn-Gwo Hwu. High sensitive and wide detecting range MOS tunneling temperature sensors for on-chip temperature detection. *Electron Devices, IEEE Transactions on*, 51(9):1514 – 1521, sept. 2004. ISSN 0018-9383. doi: 10.1109/TED.2004.833571.

[151] K. Skadron, M. Stan, R. Ribando, and S. Gurumurthi. HotSpot: an accurate and fast thermal model suitable for use in architectural studies. URL http://lava.cs.virginia.edu/HotSpot/index.htm. [Dec. 20, 2011].

[152] K. Skadron, M.R. Stan, W. Huang, Sivakumar Velusamy, Karthik Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. In *Computer Architecture, 2003. Proceedings. 30th Annual International Symposium on*, pages 2 – 13, june 2003. doi: 10.1109/ISCA.2003.1206984.

[153] F. Steenhof, H. Duque, B. Nilsson, K. Goossens, and R.P. Llopis. Networks on chips for high-end consumer-electronics TV system architectures. In *Design, Automation and Test in Europe, 2006. DATE '06. Proceedings*, volume 2, pages 1 –6, march 2006. doi: 10.1109/DATE.2006.243840.

[154] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[155] M.B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, Jae-Wook Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, and A. Agarwal. The Raw microprocessor: a computational fabric for software circuits and general-purpose

programs. *Micro, IEEE*, 22(2):25 – 35, mar/apr 2002. ISSN 0272-1732. doi: 10.1109/MM.2002.997877.

[156] A.S. Vaidya, A. Sivasubramaniam, and C.R. Das. Impact of virtual channels and adaptive routing on application performance. *Parallel and Distributed Systems, IEEE Transactions on*, 12(2):223 –237, feb 2001. ISSN 1045-9219. doi: 10.1109/71.910875.

[157] S.R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and S. Borkar. An 80-tile sub-100-W TeraFLOPS processor in 65-nm CMOS. *Solid-State Circuits, IEEE Journal of*, 43(1):29 –41, jan. 2008. ISSN 0018-9200. doi: 10.1109/JSSC.2007.910957.

[158] V. I. Varshavsky. *Self-Timed Control of Concurrent Processes: the Design of Aperiodic Logical Circuits in Computers and Discrete Systems*. Kluwer Academic Publishers, USA, 1986.

[159] B. Vermeulen and K. Goossens. A network-on-chip monitoring infrastructure for communication-centric debug of embedded multi-processor SoCs. In *VLSI Design, Automation and Test, 2009. VLSI-DAT '09. International Symposium on*, pages 183 –186, april 2009. doi: 10.1109/VDAT.2009.5158125.

[160] Arseni Vitkovski, Axel Jantsch, Robert Lauter, Raimo Haukilahti, and Erland Nilsson. Low-power and error protection coding for network-on-chip traffic. *IET Computers & Digital Techniques*, 2(6): 483–492, 2008.

[161] E. Waingold, M. Taylor, D. Srikrishna, V. Sarkar, W. Lee, V. Lee, J. Kim, M. Frank, P. Finch, R. Barua, J. Babb, S. Amarasinghe, and A. Agarwal. Baring it all to software: Raw machines. *Computer*, 30(9):86 –93, sep 1997. ISSN 0018-9162. doi: 10.1109/2.612254.

[162] S. Warnakulasuriya and T.M. Pinkston. Characterization of deadlocks in interconnection networks. In *Parallel Processing Symposium, 1997. Proceedings., 11th International*, pages 80 –86, apr 1997. doi: 10.1109/IPPS.1997.580852.

[163] P.M. Watts, N. Barrow-Williams, and S.W. Moore. Requirements of low power photonic networks for distributed shared memory computers. In *Optical Fiber Communication Conference and Exposition (OFC/NFOEC), 2011 and the National Fiber Optic Engineers Conference*, pages 1 –3, march 2011.

[164] Yaoyao Ye, Lian Duan, Jiang Xu, Jin Ouyang, Mo Kwai Hung, and Yuan Xie. 3D optical networks-on-chip (NoC) for multiprocessor systems-on-chip (MPSoC). In *3D System Integration, 2009. 3DIC 2009. IEEE International Conference on*, pages 1 –6, sept. 2009. doi: 10.1109/3DIC.2009.5306588.

[165] Qiaoyan Yu and P. Ampadu. A dual-layer method for transient and permanent error co-management in NoC links. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 58(1):36 –40, jan. 2011. ISSN 1549-7747. doi: 10.1109/TCSII.2010.2092817.

[166] C.A. Zeferino and A.A. Susin. SoCIN: a parametric and scalable network-on-chip. In *Integrated Circuits and Systems Design, 2003. SBCCI 2003. Proceedings. 16th Symposium on*, pages 169 – 174, sept. 2003. doi: 10.1109/SBCCI.2003.1232824.

[167] Changyun Zhu, Zhenyu Gu, Li Shang, R.P. Dick, and R. Joseph. Three-dimensional chip-multiprocessor run-time thermal management. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(8):1479 –1492, aug. 2008. ISSN 0278-0070. doi: 10.1109/TCAD.2008.925793.