Universitat Politècnica de València
Departamento de Informática de Sistemas y Computadores

High Performance and Power Efficient On-Chip Network Designs through Multiple Injection Ports

A dissertation submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
(Computer Science)

Author

Jesús Camacho Villanueva

Advisor

José Flich Cardo

Valencia, September 2012

Acknowledgements

I want to thank this work to the people who has been supporting me during all these years. In particular:

To Jose Flich for his leadership, and specially, for the great flexibility he has offered me throughout this thesis. I have no words!

To Jose Duato to be able to lead a large and competitive research group in which I have had the pleasure to work. I hope you get well very soon!

To Hans Eberle to give me the opportunity to work in two large companies such as Sun Microsystems and Oracle Inc. Thank you so much!

To the department's administration for their great efficiency. Continue like this!

To all the colleagues in the research group who have helped me during the course of this thesis, and specially, to Jesus Friginal for the good times at lunch. Good luck to all of you!

To my family for their unconditional support. I love you!

To my friends because they have always been there. I need you!

To Bubu for your contagious happiness. You are the best!

To Raquel to show me a new future where the dreams are able to come true. You are all for me!

Contents

Acknowledgements **iii**

Abstract **xvii**

Resumen **xix**

Resum **xxi**

1 Introduction **1**

1.1 Motivation 1

1.2 Objectives 6

1.3 Contributions 7

1.4 Dissertation Outline 9

2 Technical Background and Related Work **11**

2.1 On-chip Interconnection Networks 11

2.1.1 Design Factors 12

2.2 Interconnection Network Basics 13

2.2.1 Network Topology 14

2.2.2 Switch Device 21

2.2.3 Data Units 24

2.2.4 Switching 24

2.2.5 Flow Control 30

2.2.6 Arbitration 32

2.2.7 Routing 33

2.3 Related Work 36

2.4 Conclusions 40

3 NR-Mesh Topology 43

3.1 NR-Mesh 44
3.1.1 Tile-Based Design 45
3.1.2 Injection Algorithm 47
3.1.3 Network Routing Algorithms 50
3.1.4 Topology Properties 55
3.2 Power Management Algorithm 57
3.2.1 Algorithm for Switching Off an Input Port 58
3.2.2 Algorithm for Routing Messages 59
3.2.3 Algorithm for Switching On a Port 60
3.3 Performance Evaluation 62
3.3.1 Evaluated parameters 63
3.3.2 Simulation Model 64
3.3.3 Implementation Results 67
3.3.4 Analysis with Synthetic Traffic 68
3.3.5 Analysis with Applications 72
3.3.6 Additional Performance Comparisons and Analysis . . . 74
3.4 Conclusions 82

4 PC-Mesh Topology 85

4.1 PC-Mesh 87
4.1.1 Tile-Based Design 87
4.1.2 Injection Algorithm 88
4.2 Power Management Algorithm 92
4.3 Fault tolerance 94
4.4 Performance Evaluation 95
4.4.1 Threshold Analysis 95
4.4.2 Synthetic Traffic Results 98
4.4.3 Application Execution Time and Power Results 99
4.4.4 Results with Overloaded Systems 100
4.5 NR-Mesh versus PC-Mesh topology 103
4.5.1 Uniform Synthetic Traffic 103
4.5.2 Real Applications 104

4.6 Conclusions 105

5 HPC-Mesh Topology 107

5.1 HPC-Mesh 108

5.1.1 Tile-Based Design 109

5.1.2 Injection Algorithm 110

5.2 Fault tolerance 112

5.2.1 Faulty subnetworks 113

5.2.2 Faulty switches 113

5.3 Performance Evaluation 114

5.3.1 Synthetic Traffic Results 115

5.3.2 Real Application Results 116

5.3.3 Performance Under Faulty Networks 118

5.4 Towards a 3D mesh structure 120

5.5 Conclusions 122

6 HNPC-Mesh Topology 125

6.1 HNPC-Mesh 125

6.2 Tile Based Design 126

6.3 Injection algorithm 127

6.4 Performance Analysis 128

6.4.1 Real Applications 129

6.4.2 Real Applications with Background Traffic 130

6.5 Conclusions 131

7 Conclusions 133

7.1 Conclusions 133

7.2 Contributions 134

7.3 Future Work 136

7.4 Industry Internships and Related Publication 136

Bibliography 139

List of Figures

1.1 2D-Mesh and C-Mesh topologies. Circles are nodes and squares are switches. 3
1.2 Comparison between topologies, routing algorithms, and power management. 5
1.3 Interrelation among contributions. 9

2.1 A general overview of an interconnection network. 14
2.2 Routing, switching and flow control in a network. 15
2.3 Network architectures. 16
2.4 A crossbar network. 18
2.5 A 4×4 2-dimensional mesh and torus. 18
2.6 A multistage network topology. 19
2.7 The processing element in a tile-based CMP. 20
2.8 Different link crossings in a 2-dimensional mesh. 21
2.9 Switch architecture. 22
2.10 Switch stages. 23
2.11 Message, packets, and flits. 24
2.12 Circuit switching. 26
2.13 Store and forward Switching. 27
2.14 Virtual cut-through switching. 28
2.15 Wormhole switching. 29
2.16 Virtual channels. 29
2.17 Ack/nack flow control. 30
2.18 Stop & go flow control. 31
2.19 Credit-based flow control. 31

2.20 Deadlock situation when using XY and YX routing at the same time. 35
2.21 Concentrated mesh. 36
2.22 Flattened butterfly. 37
2.23 Diagonal Mesh. 38
2.24 End nodes connecting two switches. Circles are nodes and squares are switches. 39

3.1 NR-Mesh on-chip network topology. 44
3.2 Tile design for the NR-Mesh topology. 46
3.3 Using alternative paths in the NR-Mesh topology depending on traffic conditions. 47
3.4 Injection algorithm for the NR-Mesh topology. 48
3.5 Example of minimal and non-minimal injection ports for the NR-Mesh topology. 49
3.6 Average number of paths assuming DOR in NR-Mesh and 2D-Mesh networks. The figure also includes the average number of injection ports per node for the NR-Mesh topology. In the 2D-Mesh the number is one. 51
3.7 A non-minimal path in the NR-Mesh topology. 51
3.8 Deterministic routing algorithm for the NR-Mesh topology. . . 52
3.9 Node and switch IDs assumed by routing algorithms and messages in the network. 53
3.10 All possible paths in the NR-Mesh. 55
3.11 Adaptive routing algorithm for the NR-Mesh topology. 56
3.12 Examples of additional properties of the NR-Mesh topology. . . 57
3.13 Powering on/off the next *IP* (input port) from the previous switch. 58
3.14 Algorithm for *switching off* an input port. 59
3.15 Routing algorithm assuming the status of the network. 60
3.16 Switch stages. 61
3.17 Pipeline of the switch stages. 61
3.18 First stages of the pipeline when an input port is being powered on. 62

3.19 The NR-Mesh and other topologies. 64

3.20 Accepted traffic and power consumption for 16-node systems. Synthetic traffic patterns. 69

3.21 Accepted traffic and power consumption for 32-node systems. Synthetic traffic patterns. 70

3.22 Performance and power consumption for 16- and 32-node systems. Hot-spot scenario. 71

3.23 Normalized execution time for different applications under different topologies and routing algorithms. 73

3.24 Normalized energy consumption for different applications under different topologies and routing algorithms. 75

3.25 Link injection variation in the NR-Mesh. 76

3.26 Accepted traffic and power consumption for 16-node systems. Injection variation in a hot-spot scenario. 77

3.27 Accepted traffic and power consumption for 32-node systems. Injection variation in a hot-spot scenario. 78

3.28 16-node link injection variation in the NR-Mesh. 79

3.29 32-node link injection variation in the NR-Mesh. 80

3.30 Normalized execution time for Splash-2 applications and commercial workloads using different topologies. 81

4.1 Switch IDs and Node IDs in PC-Mesh network (left side), and Fault-Tolerant PC-Mesh network (right side). 86

4.2 16-tile design assumed for the PC-Mesh topology (left side) compared to the C-Mesh one (right side). 88

4.3 X-Y in every graph is followed from S_x to D_x represented with the same color as the subnetwork, through switches of the same color. 90

4.4 Logic for the power management algorithm. The main difference with the PML in the previous chapter is that here we power off the entire switch. 93

4.5 Faul tolerance comparison for the PC/FTPC-Mesh topologies. X-Axis shows the number of faulty switches. Y-Axis shows the fault tolerance percentage for each number of faulty switches in each topology. 94
4.6 Threshold analysis. Synthetic traffic patterns (uniform distribution of message destinations). Accepted traffic measured as flits/cycle/tile and power consumption as W. 96
4.7 Different subnetwork thresholds cases. Results are normalized for 16- and 32-node system to the C-Mesh topology where $T_{swoff} = 1$. Then, T_h is shown for C-Mesh case and (T_{swoff} T_h) are shown for PC-Mesh. 97
4.8 16-node synthetic traffic comparison. Accepted traffic measured as $flits/cycle/tile$, power consumption as W and latency as cycles. 98
4.9 32-node synthetic traffic comparison. Accepted traffic measured as flits/cycle/tile, power consumption as W and latency as cycles. 100
4.10 Application execution time and network energy consumption. . 101
4.11 Application execution time and network energy consumption with a background of 12% (16-node) and 6% (32-node) of synthetic traffic. 102
4.12 Accepted traffic and power consumption in hot-spot scenarios. 16-node system. Accepted traffic measured as flits/cycle/tile and power consumption as W. 103
4.13 Accepted traffic, power consumption and latency comparison between NR- and PC-Mesh topologies using uniform synthetic traffic. 104
4.14 Normalized application execution time and normalized network energy consumption comparison between NR- and PC-Mesh topologies using real applications. 105

5.1 HPC-Mesh topology. 108
5.2 PC-Mesh topology. 109
5.3 HPC-Mesh overview. Octagons are the end nodes and squares the switches. 110

5.4 HPC-Mesh topology tile design. 111
5.5 Injection algorithm for the HPC-Mesh topology. 112
5.6 Fault tolerance comparison for PC-Mesh and HPC-Mesh using 16 and 32 nodes. 114
5.7 16-node synthetic traffic comparison (accepted traffic measured as flits/cycle/tile). 115
5.8 32-node synthetic traffic comparison (accepted traffic measured as flits/cycle/tile). 117
5.9 16-node execution time and network energy consumption comparison. 117
5.10 32-node execution time and network energy consumption comparison. 118
5.11 16-node fault tolerance support. HPC-Mesh versus PC-Mesh (accepted traffic measured as flits/cycle/tile). 119
5.12 32-node fault tolerance support. HPC-Mesh versus PC-Mesh (accepted traffic measured as flits/cycle/tile). 120
5.13 HPC-Mesh 3D structure. 122
5.14 HPC-Mesh versus HPC-3DMesh. 123

6.1 PC-Mesh, HPC-Mesh and HNPC-Mesh tile based design. . . . 127
6.2 Injection algorithm for the HNPC-Mesh topology. 128
6.3 2D-Mesh topology with express channels. 129
6.4 Application execution time and network energy consumption comparison. Results normalized to the 2D mesh case. 130
6.5 Application execution time and network energy consumption comparison with background traffic. 131

List of Tables

3.1 Diameter (hops) and bisection BW (flits/cycle) comparison between NR-Mesh and 2D-Mesh topologies. Links with 1 flit/cycle bandwidth are assumed. 45
3.2 End node and cache coherency protocol parameters. 66
3.3 Network parameters. 66
3.4 Power consumption of the different components for the NR-Mesh topology. Target frequency set to one GHz. 67
3.5 Area and delay overheads for the different components. Target frequency set to one GHz. 68
3.6 16- and 32-node execution degradation when removing injection links from the NR-Mesh in real applications. 81

4.1 Example for different switches which can reach the node 10. . . 89

5.1 Faulty Network Support. The used acronyms in the table are FNS (Faulty Network Support) which shows the failed networks separated by a dash. For instance, 1-2 means they are failed components in subnetworks 1 and 2. N/A means Not Applicable (never supported) and Y (supported failures) or N (supported failures). 113

Abstract

Networks on-chip are becoming a key element of multiprocessor systems. As technology scales, more computing elements (processors) are included into the same chip. These components are interconnected by a network within the chip which should offer ultra low transmission latencies (tens of nanoseconds) and high bandwidth. Therefore, the design of an efficient on-chip network plays a central role.

In this thesis we analyze alternative on-chip network designs. In particular, we use different injection and ejection ports from processors to the network (several switches are reached from the same processor) to obtain several improvements.

First, network performance increases because the processors have different alternatives to inject traffic. Second, the on-chip network fault-tolerance degree increases in front of manufacturing defects (becoming more important as technology advances). Third, this technique allows aggressive policies to switch off components which allows to reduce power consumption significantly.

Different topologies, derived from the injection mechanism have been proposed and evaluated in terms of performance, implementation cost and energy (or power consumption) savings. Specific network on-chip simulators for different techniques have been developed, to analyze and to support the claimed results.

In this thesis we follow an incremental approach, where each topology designed is an improvement over the previous proposal, and, taking into account the existing topologies in the state of the art. To summarize our work, our effort is focused in obtaining an excellent trade-off between performance, power consumption and fault tolerance support in a network on-chip.

For the first proposal (Nearest neighboR Mesh Topology or NR-Mesh topology), we achieve improvements in performance up to 7% and up to 75% in power consumption, on average, when compared to the 2D-Mesh topology. For the second proposal (Parallel Concentrated Mesh Topology or PC-Mesh topology), the benefits compared to the NR-Mesh are 20% in performance and 60% in power consumption in a 32-Node sytem. In addition, when high traffic arises the PC-Mesh topology outperforms the Concentrated Mesh Topology (C-Mesh topology), otherwise, its behavior is similar to the concentrated mesh topology. With the next proposal (Homogeneous Parallel Concentrated Mesh Topology or HPC-Mesh topology) we fix a drawback in the PC-Mesh network, that is, we provide full tolerance support without adding extra resources and without decreasing performance in low traffic conditions. An hybrid design between PC-Mesh and HPC-Mesh (HNPC-Mesh) allows the last one to achieve the PC-Mesh performance level when high traffic arises. Finally, we explore the use of express links over 2D-Mesh network on-chip topology and compare it against HNPC-Mesh. Although the execution time in real applications is only slightly higher on average in the 2D-Mesh with express links, the power consumption (due to the high degree of the switches) increases dramatically.

Resumen

Las redes dentro de un chip se están convirtiendo en el elemento principal de los sistemas multiprocesador. A medida que aumenta la escala de integración, más elementos de cómputo (procesadores) se incluyen en el mismo chip. Estos componentes se interconectan con una red dentro del chip que debe ofrecer latencias de transmisión ultra bajas (orden de nanosegundos) y anchos de banda elevados. El diseño, pues, de una red eficiente dentro del chip juega un papel fundamental.

En la presente tesis se analizan diferentes alternativas de diseño de las redes en el chip. En particular, se hace uso de la posibilidad de utilizar diferentes puertos de inyección desde los procesadores con el fin de obtener diferentes mejoras.

En primer lugar, las prestaciones aumentan al tener procesadores con distintas alternativas de inyección de tráfico. En segundo lugar, además aumenta la tolerancia a fallos frente a defectos de fabricación (mas importantes conforme avanza la tecnología). Y en tercer lugar, permite una política de apagado de componentes más agresiva que nos permita un ahorro significativo de energía.

Hemos evaluado diferentes topologías derivadas del mecanismo de inyección en términos de prestaciones, coste de implementación, y ahorro de consumo. Además, hemos desarrollado simuladores específicos para las distintas técnicas utilizadas.

Cada topología diseñada supone una mejora respecto a la anterior, y por supuesto, teniendo en cuenta las topologías existentes. En resumen, nuestro esfuerzo se centra en conseguir un excelente compromiso entre prestaciones, consumo y tolerancia a fallos dentro de una red en chip.

Para la primera propuesta (topología NR-Mesh), se alcanzan mejoras en prestaciones de un 7% y hasta de un 75% en reducción de consumo de media, comparado con la malla 2D o malla de 2 dimensiones. Para la siguiente propuesta, la malla concentrada paralela (PC-Mesh), el beneficio en prestaciones que se obtiene es de hasta un 20%, así cómo de un 60% en reducción de consumo, para un sistema de 32 nodos. Además, cuando el tráfico en la red aumenta, la malla concentrada paralela es capaz de superar a la malla concentrada (C-Mesh). Sin embargo, cuando el tráfico es más bien reducido, la PC-Mesh se comporta exactamente igual a la C-Mesh. Para la siguiente red, llamada malla paralela concentrada homogéna (HPC-Mesh), se consigue una tolerancia a fallos total sin necesidad de aumentar recursos en la red a diferencia de la anterior propuesta. Para tráficos moderados, la HPC-Mesh se comporta de forma adecuada, sin embargo, cuando el tráfico en la red aumenta significativamente, se requiere la implementación de un diseño híbrido entre la PC-Mesh y la HPC-Mesh, la cuál es la última propuesta presentada en esta tesis (llamada topología HNPC-Mesh). Finalmente, se explora la malla 2D con canales exprés comparándola con las propuestas anteriormente mencionadas. Aunque el tiempo de la malla 2D con enlaces exprés pueda ser ligeramente inferior a nuestras propuestas, el aumento de consumo es enorme debido al alto grado de enlaces que contienen los conmutadores.

Resum

Les xarxes dins d'un xip s'estan convertint en l'element principal dels sistemes multiprocessador. A mesura que augmenta l'escala d'integració més elements de còmput (processadors) s'inclouen en el mateix xip. Estos components s'interconnecten amb una xarxa dins del xip que ha d'oferir latències de transmissió ultra baixes (orde de nanosegons) i amples de banda elevats. El disseny, doncs, d'una xarxa eficient dins del xip juga un paper fonamental.

En la present tesi s'analitzen diferents alternatives de disseny de les xarxes en el xip. En particular, es fa ús de la possibilitat d'utilitzar diferents ports d'injecció des dels processadors a fi d'obtindre diferents millores.

En primer lloc, les prestacions augmenten al tindre els processadors distintes alternatives d'injecció de tràfic. En segon lloc, augmenta la tolerància a fallades enfront de defectes de fabricació (mes importants conforme avança la tecnologia). I en tercer lloc, permet una política d'apagat de components més agressiva que ens permeta un estalvi significatiu d'energia.

Diferents topologies, derivades del mecanisme d'injecció són avaluades en termes de prestacions, cost d'implementació, i estalvi de consum. Simuladors específics per a les diferents tècniques han sigut desenrotllats.

Cada topologia dissenyada suposa una millora respecte a l'anterior proposta, i per descomptat, tenint en compte les topologies existents. En resum, el nostre esforç es centra a aconseguir un excellent compromís entre prestacions, consum i tolerància a fallades dins d'una xarxa dins del xip.

Per a la primera proposta (la topologia NR-Mesh), s'aconseguixen unes millores en prestacions d'un 7% i fins a un 75% en reducció de consum com a mitjana, comparat amb la malla 2D (o malla de 2 dimensions). Per a la següent proposta, la malla concentrada paral.lela (PC-Mesh), els beneficis en prestacions són de fins a un 20%, així com d'un 60% en reducció de consum, per a un sistema de 32 nodes. A més, quan el tràfic en la xarxa augmenta, la malla concentrada paral.lela supera a la malla concentrada (C-Mesh), no obstant això, quan el tráfic és reduït, la PC-Mesh es comporta exactament igual a la malla concentrada. En la segent proposta, que és la malla parallela concentrada homogènia (HPC-Mesh), aconseguim una tolerància a fallades completa sense necessitat d'augmentar recursos en la xarxa. Per a tràfics no excessius, la HPC-Mesh hi és prou. No obstant això, quan el tràfic en la xarxa augmenta significativament, es requerix d'un disseny híbrid entre la PC-Mesh i la HPC-Mesh, el quin és la nostra última proposta (anomenada HNPC-Mesh). Finalment explorem la malla 2D amb canals exprés. Encara que el temps d'execució és lleugerament inferior al de les nostres propostes, el consum augmenta enormement a causa de l'alt grau d'enllaços dels encaminadors.

Chapter 1

Introduction

In this chapter, we first introduce the reasons that have motivated this dissertation (Section 1.1) together with a brief context description and the challenges this thesis addresses. Then, we briefly define the specific objectives aimed by the dissertation (Section 1.2). After that, we summarize the main contributions (Section 1.3). Finally, we outline the structure of the remaining chapters in this document (Section 1.4).

1.1 Motivation

High-performance computing (HPC) is defined as the use of parallel processing for running advanced application programs efficiently, reliably and quickly. The term applies especially to systems that function above a teraflop and occasionally used as a synonym for supercomputing, although technically a supercomputer is a system that performs at or near the currently highest operational rate for computers. Some supercomputers work at more than a petaflop. See [53] for more details.

On the other hand, research in microarchitecture has always been shaped by underlying technology trends, making it a rapidly changing and vigorous field. As technology advances, previously discarded approaches are revisited with dramatic commercial success (e.g., superscalar processing became possible with ten-million transistor integration). By the same token, technology limitations cause a rethinking of the status quo (e.g., deeper pipelinining seems unsustainable due to increasing power consumption). As we build increasingly complex parallel systems, one of the greatest challenges is in providing the interconnection networks that permit the system components to efficiently communicate. These must be high-performance (low-latency and high-bandwidth), flexible, scalable, simple to design and power efficient.

As technology scales, the number of transistors that can be integrated on a chip increases. This allows designers to add more functionality on current microprocessors. The current trend, however, is to replicate basic simple processor components (together with cache memories) and thus, increase the number of processing elements on the same chip. These chips are known as Chip MultiProcessors (CMPs). This design style is preferred over a design of a big and complex processor core. The reason is that power consumption is becoming the limiting factor and simpler processors have a better performance-power consumption trade-off.

Currently, there are chip prototypes and real products with tens of processors. Examples include the Intel Polaris chip [47] with 80 simple cores, and the Single-chip Cloud Computer [50] with 48 x86 compatible processors, each able to run an operating system. Also, Tilera provides its new 100-core chip [52].

With advances in technology, we can expect chips with hundreds of cores in the near future. Thus, the way these cores are connected becomes an important and challenging issue. Although buses, rings, and crossbar topologies were used in initial systems (e.g. Cell Broadband Engine processor [45]), these structures do not scale well and therefore achieve low performance when the number of cores is high. Beyond simple buses, the idea of an on-chip network illuminates a vast design space for building scalable interconnects.

In fact, current CMP systems rely on a 2D-Mesh topology. In such a topology, every switch is connected to its neighbors in the north, east, west, and south directions. The 2D-Mesh is shown in Figure 1.1. The mesh network is appealing since it matches the planar surface of the chip. Indeed, the tile-based design (a tile is designed and the chip is built by replicating the same tile design) promotes the use of a 2D structure. One negative aspect of the 2D-Mesh topology, however, is its increase in the number of hops when communicating with distant nodes. This becomes a problem as the system size increases.

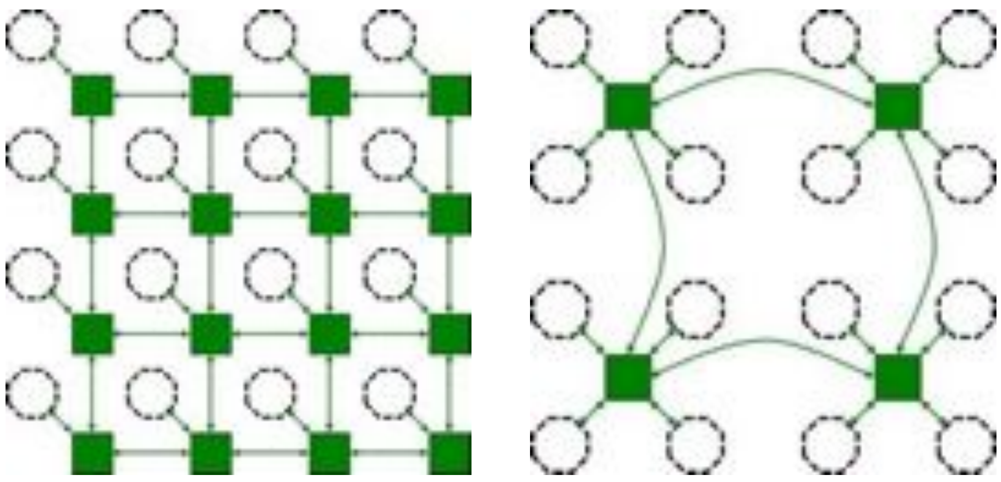


Figure 1.1: 2D-Mesh and C-Mesh topologies. Circles are nodes and squares are switches.

Anyway, once the topology is decided, the routing algorithm determines which output port a message must take in order to reach its destination. In a 2D-Mesh topology, the most efficient routing algorithm (in terms of implementation complexity and power consumption) is dimension-order routing (DOR). DOR is implemented on every switch and requires small logic blocks. The message first moves in the X dimension, and once it reaches the destination column, then it moves through the Y dimension, always following minimal paths. The low complexity of DOR makes it very appealing for network on-chip (NoC) designers.

However, DOR routing is not flexible as it allows only one single path for every source-destination pair. Therefore, DOR does not tolerate a single failure since the failure will disconnect several pairs of end nodes. In addition, DOR may lead to congestion in the network. As it does not support alternative paths, messages are forced to follow a single path, and thus, there is no way to avoid or escape from a congested spot in the network.

An alternative to DOR is the use of adaptive routing. In such a protocol, switches are able to select different output ports for the same destination, depending on the current status of those ports. Thus, local congestion can be alleviated by the use of adaptive routing. Typically, minimal paths are supported by adaptive routing, thus, the message gets closer to its destination at every performed hop. To avoid deadlocks (messages holding resources and ciclically requesting those resources, thus never advancing), an acyclic escape path is implemented as a different virtual channel. If adaptive output ports are not available, then the escape path is taken [10]. An evaluation of routing algorithms is done in [26]. More details about deterministic and adaptive routing can be found in [11].

On the other hand, a low network utilization may lead to a large waste of power. Most of the time, network components (switches and links) will be idle but powered on. As power consumption is becoming the limiting design factor in current chips, it is a requirement to adjust the power consumption of the network components to the real needs of applications. For example, if the network could be switched off during the time the components are not used (90% of the time, on average), then large savings would be achieved. In [35], it is reported that 30% of total chip power consumption is due to the network indicating that power savings in the network can have a significant overall impact.

However, switching off network components must be done carefully. The policy to turn a component on and off must be done in collaboration with the routing algorithm and topology. For instance, DOR over a 2D-Mesh offers only a single path for every source-destination pair, thus, switching off the links and switches can disconnect frequent communication flows. More flexible topologies (offering alternative paths) would be beneficial. Also, the routing algorithm may affect the effectiveness of a power consumption management technique. A proper routing algorithm must provide enough alternative paths to messages to maximize the time network components are switched off. Obviously, the longer the time a component is off, the greater the savings in power are. In this sense, adaptive routing can be of great help. Figure 1.2 shows the dependencies between the topology, the routing algorithm and the flexibility provided in terms of path and in terms of power savings.

In addition to this, power consumption can be high due to the excessive number of network components. An alternative network on-chip topology (to the 2D-Mesh case) is the concentrated mesh (C-Mesh) [2]. In this topology, the hop count decreases by reducing the number of switches (75% reduction when compared with the 2D-Mesh with the same number of end nodes) as sets of four neighboring nodes are connected to the same switch. See Figure

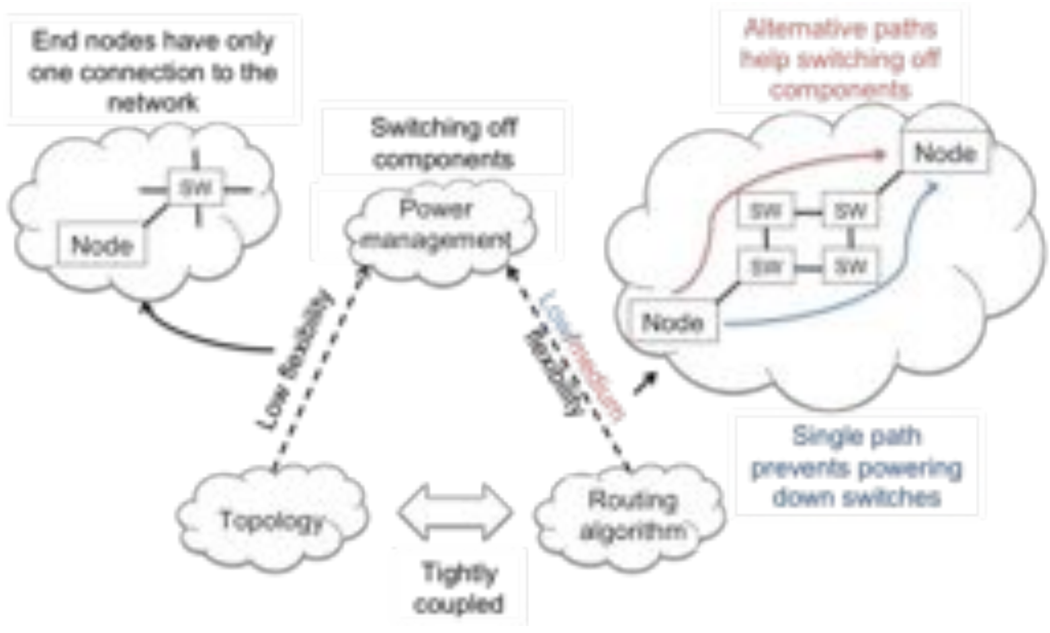

End nodes have only one connection to the network
SW
Node
Switching off components
Power management
Alternative paths help switching off components
Node
SW
SW
SW
SW
Node
Low flexibility
Low/medium flexibility
Single path prevents powering down switches
Topology
Routing algorithm
Tightly coupled

Figure 1.2: Comparison between topologies, routing algorithms, and power management.

1.1. The C-Mesh network scales better than the 2D-Mesh topology because of its lower hop count. In addition, despite the larger number of switch ports, having a quarter of switches than in 2D-Mesh leads to large savings in power consumption. Unfortunately, the reduced bisection bandwidth of the C-Mesh topology leads latency to exponentially increase and, potentially, high energy consumption due to congestion.

Therefore, the C-Mesh topology is a good candidate topology when traffic requirements are low. With low injection loads, packets will experience low latency values and the network power consumption will be low. However, with high traffic requirements (bursty traffic, barrier synchronization, hot-spots) the low bisection bandwidth of the network leads to congestion, thus high packet latencies and large power consumption values.

There are different solutions that address the capacity problem of the C-Mesh network for high traffic requirements. One possible solution is the use of express links to directly connect non-neighboring switches. Other solutions rely on higher dimensional topologies. However, most of these solutions require higher radix switches, which have proven to severely impact the operating frequency of switches and the power consumption [33].

From all these comments, we can deduce that the main drivers of power consumption in on-chip networks are the number of resources, the way they are connected (topology), the switch complexity (mainly its radix), and the flexibility provided by the routing algorithm. A trade-off between performance and power consumption exists, and these four design decisions impact on this trade-off with different degrees. In this thesis we address this issue.

Besides the performance and power consumption dimensions, one third additional dimension is fault tolerance. As systems become more complex, the manufacturing challenge increases significantly (devices are each time smaller). This inevitably leads to components that become hard to be manufactured without failures and/or components that are largely insensible to the external environment (heat, wear out, electrical noise). Thus, deriving fault tolerant mechamisms becomes critical. As previously stated, XY routing does not offer any back up solution for a broken link, thus, its use becomes compromised. In this thesis we also take into account the fault tolerant dimension in perspective, although to a lower extent.

1.2 Objectives

This section presents the objectives of this dissertation. The main goal we pursue in this thesis is to improve the throughput, reduce the power consumption and provide a good degree of fault tolerance for future network on-chip designs. However we tackle this generic and well-known goal in a different manner. We exploit the end node network interfaces and design them in accordance to the switches and topology. Notice that network interfaces (NIs) are usually left out when researching on networks. End nodes are usually assumed as ideal injector and sink elements. NIs will provide the required flexibility to achieve a good performance-power-fault tolerance trade-off. In order to achieve this, we pursue the following specific goals:

- Achieve new topology designs where the end nodes have much larger flexibility for routing packets throughout the network, thus giving more chances to network components to switch off and thus save power.

- Implement new and power-aware routing algorithms for the proposed topologies in order to maximize power saving but without compromising performance. The new power-aware routing algorithms will exploit the new properties of the topologies.

- Design injection algorithms (depending on the topology) in combination with the routing algorithms in order to achieve the power consumption goals.

- Analyze the fault tolerance properties of the proposed topologies and asses its suitability.

- Design power management mechanisms to power on and off unnused components in the network to save power consumption. This will be done in collaboration with the injection algorithm and with the power-aware routing algorithm.

- Care about network latency, by providing topologies with lower average hop distances, together with topologies that allow high throughput values.

1.3 Contributions

The previous objectives led to the following contributions briefly summarized in this section.

Usually, it is assumed that every end node is connected to one switch. When combined with DOR routing a single path exists for each pair of end nodes. The contributions in this thesis provide several injection ports between an end node and more than one switch, improving the fault tolerance support and avoiding congested situations, all of this increasing the bisection bandwidth. Besides, this technique allows us the definition of a logic capable of efficiently powering off unnused network components for longer periods of time.

We take as a reference design the 2D-Mesh and C-Mesh topologies, and address its capacity limitations with an alternative approach. Indeed, our challenge is to address these inconveniences by developing new network on-chip topologies able to obtain savings in power consumption in low traffic conditions, as in the C-Mesh topology, and to avoid congestion when high traffic loads appear, but still exhibiting low end-to-end latency. The topologies are introduced in a sequential manner. Each topology tries to solve the inefficiencies of the previous one, but always based on the concept of multiple injection ports. This is the key common property of all the proposals.

In order to achieve large power saving values, every topology is enriched by a simple (but small when implemented) injection algorithm at the network interface of each node. The algorithm is in charge to manage the injection ports in order to maximize power savings without compromising performance. We provide a detailed implementation and evaluation (in terms of area, power, and delay) of the injection algorithm.

As a first proposal we introduce the *NR-Mesh* topology. In this topology, every end node is able to inject packets to up to four different switches providing higher flexibility, reducing the hop count and improving the fault tolerance. NR-Mesh is combined with a deterministic routing algorithm, and a fully adaptive one. Also, an injection algorithm is conceived for NR-Mesh.

The second proposal, called *PC-Mesh*, improves the NR-Mesh by decoupling network components into four parallel concentrated networks. This leads to obtaining larger power savings in low traffic conditions, and achieving a very good trade-off between performance and power consumption when the network load increases.

The *HPC-Mesh* is then introduced, which improves the previous proposal in terms of fault tolerance, and provides a simple and flexible implementation for a 3D-Mesh structure.

Finally, the *HNPC-Mesh* is a hybrid design between PC-Mesh and HPC-Mesh topologies, taking the best of every topology with a minimal effort design. Besides, we compare both topologies with a 2D-Mesh with express links, obtaining better results for our proposals.

All of these topologies have a Power Management Logic (PLM) able to power off unused network components to save power consumption when possible. In the NR-Mesh case we turn off unused ports, while in the rest of proposals we are able to switch off entire parallel subnetworks.

Figure 1.3 shows the contributions and how they interrelate among them.

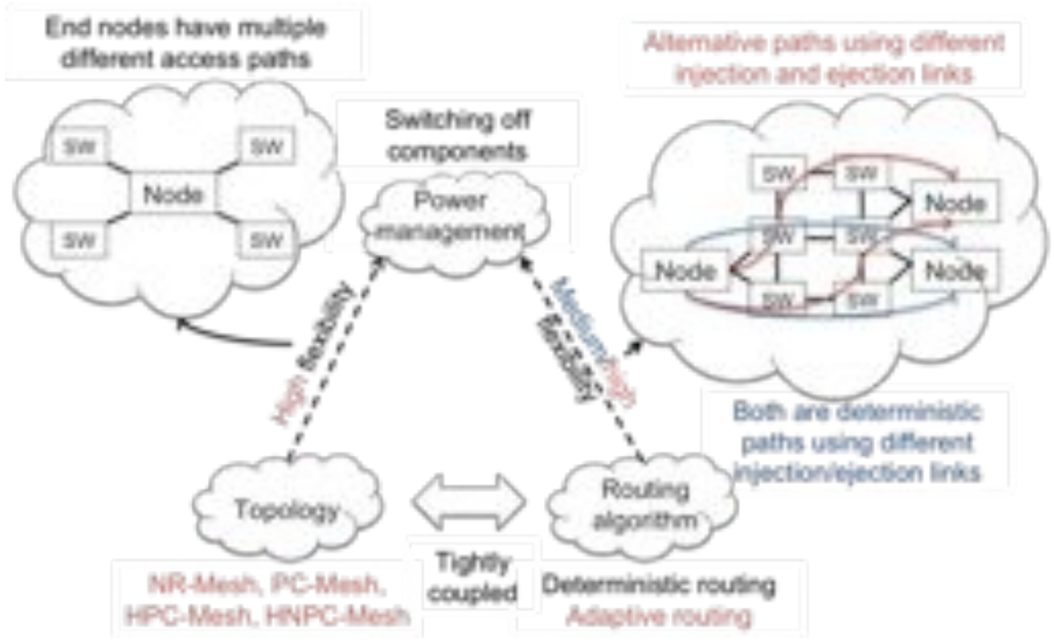

End nodes have multiple different access paths
SW
SW
Node
SW
SW
Switching off components
Power management
Alternative paths using different injection and ejection links
SW
SW
Node
SW
SW
Node
Node
SW
SW
High flexibility
Medium/High flexibility
Both are deterministic paths using different injection/ejection links
Topology
Routing algorithm
Tightly coupled
NR-Mesh, PC-Mesh, HPC-Mesh, HNPC-Mesh
Deterministic routing
Adaptive routing

Figure 1.3: Interrelation among contributions.

1.4 Dissertation Outline

This dissertation starts with the introductory chapter (Chapter 1). After that, it continues with Chapter 2 describing the basics of on-chip interconnection networks and an analysis of the current state of the art that contributes to the matter of this dissertation. Chapter 3 presents the NR-Mesh (Nearest neighboR Mesh) on chip network topology. Chapters 4 and 5 present the PC-Mesh (Parallel Concentrated Mesh) and HPC-Mesh (Homogeneous Parallel Concentrated Mesh) topologies, respectively. In Chapter 6 we mix the PC-Mesh and HPC-Mesh topologies by building a hybrid design obtaining the best from every topology, called HNPC-Mesh (Homogeneous/Non homogeneous Parallel Concentrated Mesh). All the proposed topologies are compared with the most common topologies currently used, and the most challenging proposals for networks on chip topology nowadays. Finally, we end with Chapter 7, summarizing the conclusions and displaying the contributions related to the research field.

Chapter 2

Technical Background and Related Work

In this chapter, the goal is to describe the basics and terminology of on-chip interconnection networks (the technical background). For the sake of brevity we cover the main concepts, but it is not the intention of this chapter to provide an in-depth overview of the subject, since the on-chip network field is as complex as the general interconnection network field, and there exist several aspects that are beyond the scope of this dissertation. We refer the reader to established books on this topic and related ones for further background and introductory material [8, 11, 12, 27].

First, in Section 2.1, we present a brief description of the design parameters that involve networks on-chip. Then, in Section 2.2, we dive into a more extensive description of the main aspects that surround this kind of networks, paying special attention to the topologies and routing algorithms. Finally, this chapter, in Section 2.3, shows the related work and existent contributions that serve as a reference for this dissertation.

2.1 On-chip Interconnection Networks

In the field of interconnection networks, there is a growing interest and amount of research in the on-chip domain. Since the appearance of the NoC concept, many research groups and institutions have turned their attention into it and they have contributed to a plethora of proposals in related conferences and journals. The integrated circuit technology has evolved to accommodate a multiprocessing device capable of high-performance computation. As a result of the high integration scale in the deep sub-micron domain and the increasing number of connecting elements, on-chip interconnection has become a need and influences the performance of the final system. So, any gain in the efficiency of the on-chip interconnection layer will be highly beneficial for the entire system.

Next, we describe the main design factors that should drive any research devoted to NoCs.

2.1.1 Design Factors

As aforementioned, NoCs play a major role in the design of the modern high-performance computers, nevertheless, they are not simple; there are many factors that affect the choice of an appropriate interconnection layer at design time. The main factors are:

- *Performance.* As commented, performance is a design factor point in interconnection networks, not only from the point of view of raw throughput, but also from the point of view of latency. Latency is a critical design issue in several systems such as real-time systems. Moreover, in on-chip networks, messages must reach destinations in terms of few nanoseconds. The topology (how the elements are connected between them) and the routing algorithm (the path that the messages should take) influence both throughput and latency.

- *Scalability.* Scalability is the first design rule that an interconnect designer should keep in mind. Scalability in interconnection networks implies that the bandwidth of the network increases proportionally to the number of elements of the system. Latency should also be kept to reasonable limits when increasing the system size. Otherwise, the interconnection network would become a bottleneck, limiting the efficiency of the whole system. Scalability also implies that network cost and resources are proportional to the network size.

- *Reliability.* An interconnection network should be able to deliver information in a reliable manner. Interconnection networks should be designed for continuous operations in the presence of a limited number of faults. More important, as technology scales, manufacturing defects will increase, thus demanding an efficient treatment.

- *Simplicity.* Not only for the sake of cost, but making simpler designs leads to architectures that work with higher operating frequencies, thus, increasing the system performance, and occupying less area. In fact, the silicon area usage is a critical aspect in on-chip networks. Reducing the area translates into the opportunity for making room for more devices inside the chip, that is, providing more functionality.

- *Power consumption.* One of the most important aspects in networks on-chip, not so critical in other network environments, is the reduction or minimization of power consumption. Indeed, effective power-aware techniques are needed to bring better management of the total power consumed by the processing cores.

All these previous factors must be specifically considered when designing an on-chip network. In this thesis all the contributions take these factors, directly or indirectly, as a reference. In the next section we present the basics for interconnection networks.

2.2 Interconnection Network Basics

The network architecture design is the result of several design choices like network *topology*, *switching, flow control* and *routing strategies.* The network topology defines the physical interconnection between nodes and other elements. The switching and flow control techniques define how and when the information is transmitted (advances) through the network resources. Finally, the routing strategies manage the different path choices of communication between the nodes.

There are some common elements that conform a network architecture. The first elements are the nodes. Nodes are the elements that communicate

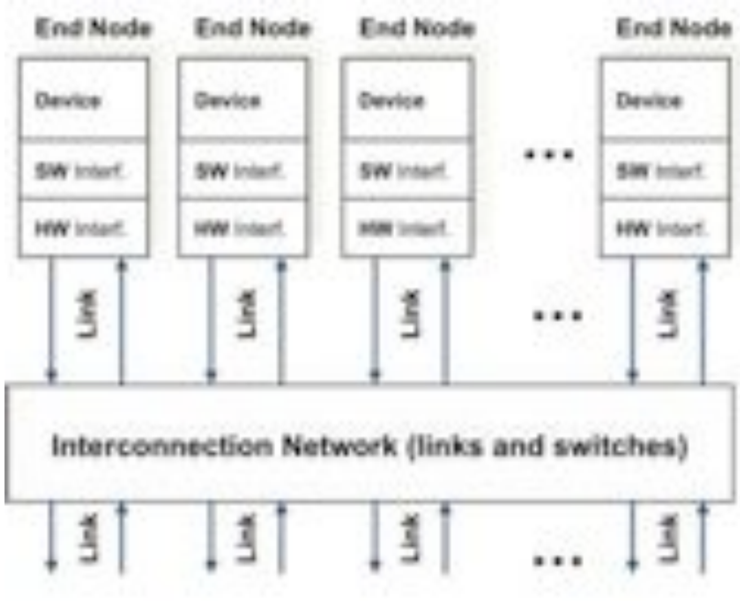


Figure 2.1: A general overview of an interconnection network.

through the network and perform basically two tasks: computation and/or storage. Nodes connect to other nodes through a network interface associated to a switch, depending on the topology of the network. A switch is the basic component that connects different devices. Links are used to connect the devices (network interfaces and switches) among them. Figure 2.1 shows an overall overview of the interconnection network and its devices. Figure 2.2 shows a network with a 2D-Mesh topology highlighting where switching, flow control and routing is performed. Next, we describe each network component.

2.2.1 Network Topology

Different network categories can be devised based on how all the elements of a system are connected to the network (see examples in Figure 2.3):

- *Shared-medium networks:* In this type of network there is a transmission medium that is shared by all the nodes, and only one node is able to communicate at a time while the rest of nodes read (and monitor) from the shared medium. Every device has the circuitry to handle addressing of other nodes and data management. In these networks, the routing device is the shared medium, called also bus. Buses have limited

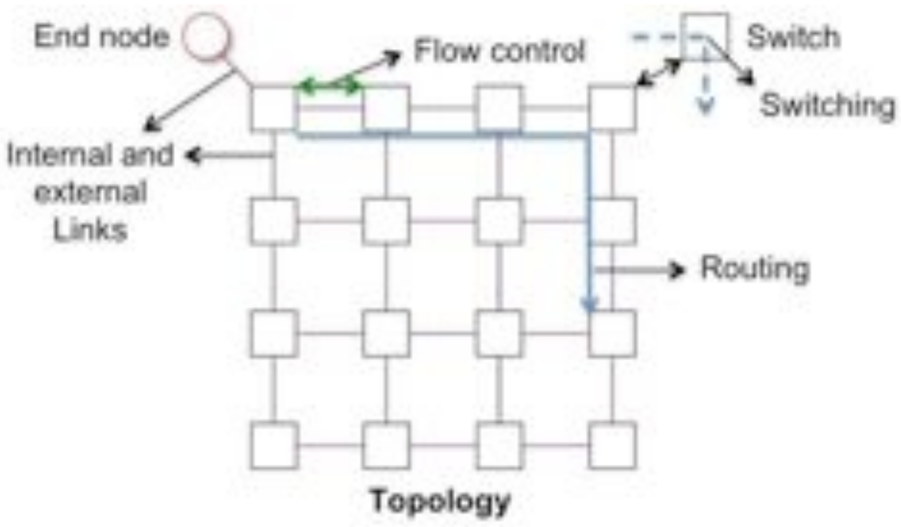
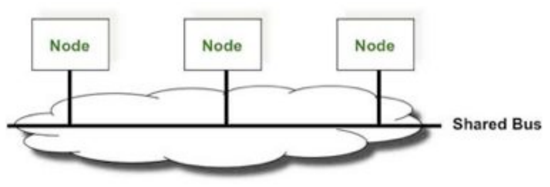


Figure 2.2: Routing, switching and flow control in a network.

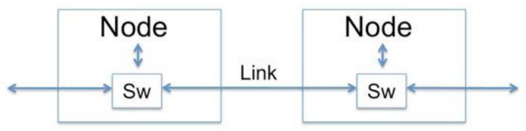
bandwidth, so they suffer from scalability problems, as the number of connected nodes increases.

- *Direct networks:* Each node has a routing device attached, called switch, which is the component that establishes the connection to other nodes through point-to-point links. The concept of network interface is weak in this type of networks as the end node and the switch (also called router) are tightly connected. Nodes are connected according to a certain interconnection pattern (topology).
- *Indirect networks:* Instead of directly connecting the nodes through point-to-point links, the communication between a pair of nodes can be performed by intermediate stand-alone switches. Every node has a network interface that connects to a switch (through a point-to-point link) and switches are connected between them (also through point-to-point links).
- *Hybrid networks:* This type of network is a mixture of the previous approaches. In general, it combines mechanisms from shared-medium-networks and direct or indirect networks.

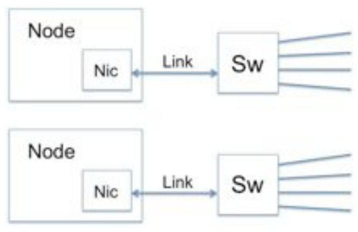
Although there are very subtle differences between direct and indirect networks, the functionality is similar in many aspects. An indirect network in



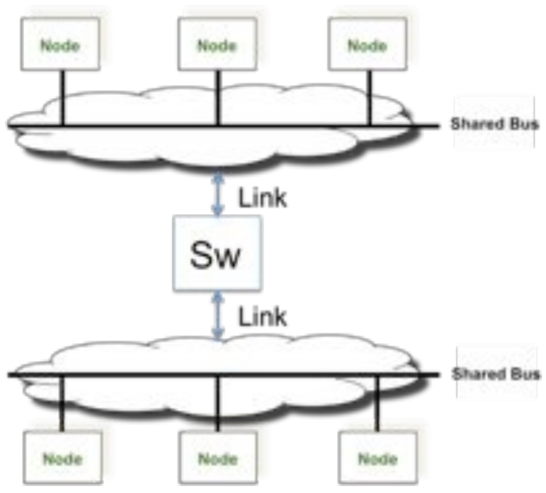
(a) Shared-medium network



(b) Direct network



(c) Indirect network



(d) Hybrid network

Figure 2.3: Network architectures.

which every switch is connected to a single node is equivalent to a direct network. Also, terms router and switch, although having different meanings, are typically used with no distinction by the community, so both terms for the routing devices are interchangeable. In the rest of the dissertation, unless noted, the term switch or router (the last one mainly in figures and tables) can be assumed.

There are also some common aspects to all these types of networks. Although links are usually formed by two communication channels, one in each direction, one of the basic aspects of a network is how communication channels are arranged. Network performance significantly differs if links are bidirectional or unidirectional. This choice impacts directly on the routing techniques and algorithms and associated issues, like deadlock avoidance. We assume the use of bidirectional channels on every link, though.

Each type of network can also be categorized with different properties:

- *Switch degree:* This property refers to the number of channels that connect a switch to its neighbours.

- *Diameter:* Is defined as the maximum distance between a pair of end nodes in the network.

- *Regularity:* A network is defined as regular when all the switches have the same degree.

- *Bisection Bandwidth:* Bisection of the network is the minimum set of links that split the network in two equal halves. Bisection bandwidth is the resulting bandwidth at the bisection.

- *Homogeneity:* A network is homogeneous if every node and its connectivity is equal in all aspects to the rest of nodes providing a homogenous floorplan.

There are three common basic topologies used in interconnection networks. The first one is the *crossbar*. A crossbar (see Figure 2.4) allows the connection from any node to any other node simultaneously at the same time other connections are established (as long as the requested input and output are free). Crossbar networks, typically, are used for high-performance computing multiprocessor solutions and in the design for switches in direct networks. The drawback with crossbar topologies is that they do not scale as system grows due to the quadratic requirement of connections.

Strictly orthogonal topologies are the second common type. In this kind of networks we can find the *n-dimensional meshes and tori* (see Figure 2.5). A n-dimensional mesh or torus has k nodes placed along each dimension. A mesh differs from a torus because it does not have the wraparound channels that connect the nodes in the borders of the topology. Note that the torus topology duplicates the bisection bandwidth of the mesh topology and reduces its diameter. These topologies are the typical examples used for direct networks.

Multistage interconnection networks (MINs) are topologies driven by the concept of indirect networks as seen in Figure 2.6. Between input and output devices there are several switch stages. The arrangement of stages and the connection patterns determine the routing in these networks. MINs have been widely used to interconnect parallel computers with large number of processors

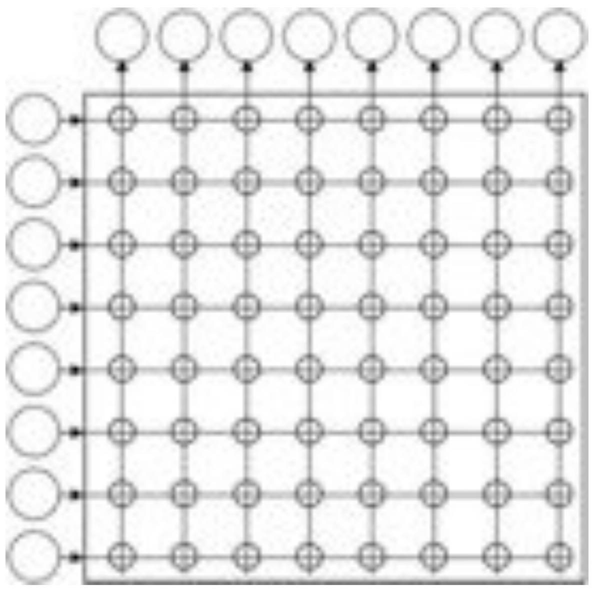
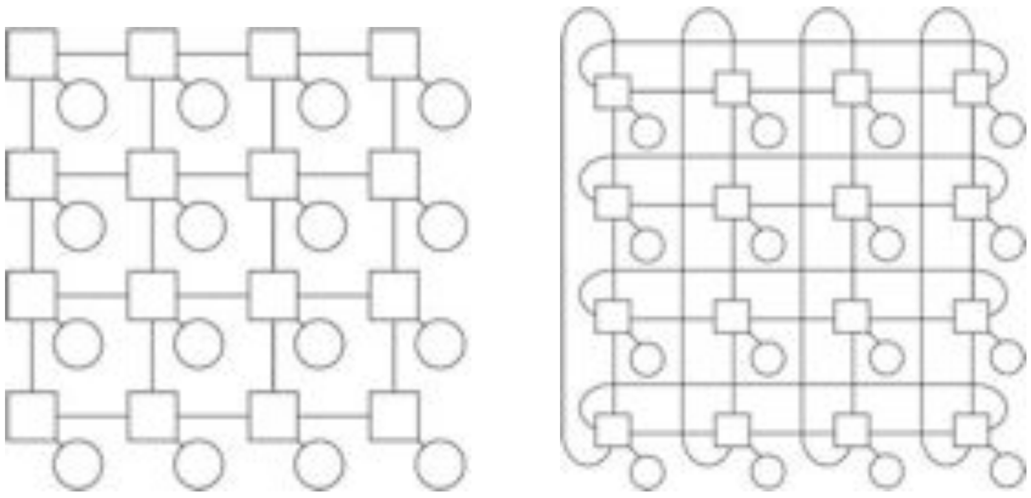


Figure 2.4: A crossbar network.



(a) Mesh (b) Torus

Figure 2.5: A 4 × 4 2-dimensional mesh and torus.

in commercial and high-performance solutions. However, for on-chip networks mapping of such topology patterns in the 2-dimensional surface of the chip is a big challenge.

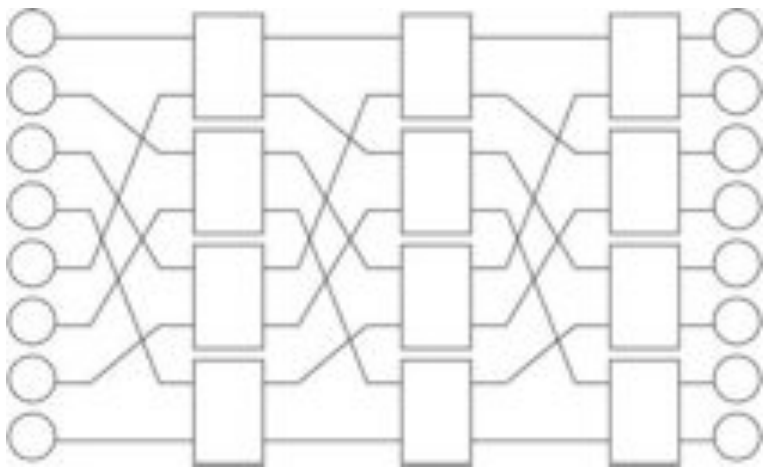


Figure 2.6: A multistage network topology.

Networks-on-chip Topologies

Earlier on-chip communication architectures relied on the share-medium network paradigm, that included buses as the communication subsystem. But the trend nowadays is to include a reasonably large number of processing cores inside the chip, and shared-medium network designs have poor scalability and bandwidth impacting heavily on the network performance.

NoCs emerged, thus, as a response to effective on-chip communication. NoCs are based on a paradigm that is a mixture of the concept of direct and indirect networks. Current multicore architecture designs made of elemental brick nodes work together to achieve the high-performance computing goal (the chip is formed by several processing devices). These devices are called usually *tiles*. A tile, fundamentally, apart from the processing elements, has also a switch attached that handles the communication between tiles. See a simplified schematic of a tile in Figure 2.7.

As the chip can be seen as a collection of tiles, there is a major taxonomy where chips can be differentiated between homogeneous (inducing regular topologies) and heterogeneous designs (more suited with irregular topologies). Every tile is connected to a subset of other tiles through an on-chip network. An example of homogeneous configurations are the tiled chip multiprocessors (CMPs) where all the tiles are equal, i.e, tiles are replicated along the chip (see Figure 2.7). Instead, high-end multiprocessor systems-on-chip (MPSoCs) are examples of heterogeneous designs where nodes are different in many aspects:

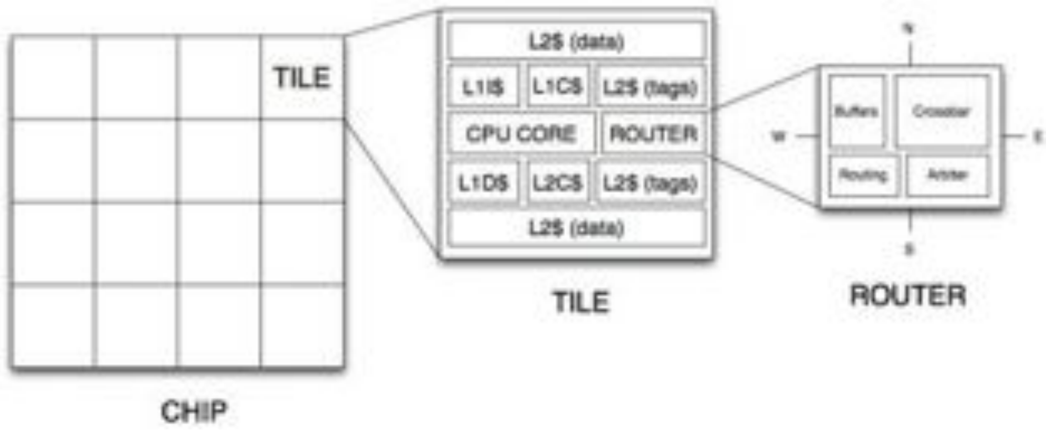


Figure 2.7: The processing element in a tile-based CMP.

size, functionality, performance, throughput, etc. In this thesis, we focus on CMP systems with regular structures.

A popular choice in NoC designs is the use of orthogonal topologies as most of the direct network architectures are implemented with this property in mind. Orthogonal topologies, which are associated with regular patterns, allocate the nodes in a n-dimensional space, with k nodes along each dimension. Every switch has at least one link crossing one dimension. Every switch is labelled with an identifier depending on its coordinates, and all the links that communicate to other switches are bidirectional (formed by two channels, one in each direction). As the distance between a pair of switches is the sum of the offsets in all dimensions, the routing strategy is usually implemented as a function of selecting the links that decrement the absolute value of the coordinate offsets between a source node and a destination node, a very simple mechanism. The most popular design in NoCs is the n-dimensional mesh, used in most of the commercial and non-commercial (prototype) NoC designs. The most suitable topology is the 2-dimensional mesh (Figure 2.5(a)). This topology is vastly used (or at least assumed) as it fits the chip layout.

As every switch is located within the network by its coordinates on a n-dimensional space, a switch in a 2-dimensional graph will be numbered by a group of two coordinates, (x, y), one for each dimension. Crossing a link means decrementing or adding an unitary value to the offset of the dimension between the two nodes that share the associated link. See an example in Figure 2.8. Moving from node 1, with coordinates $(1, 0)$, in $Y+$ direction results in node

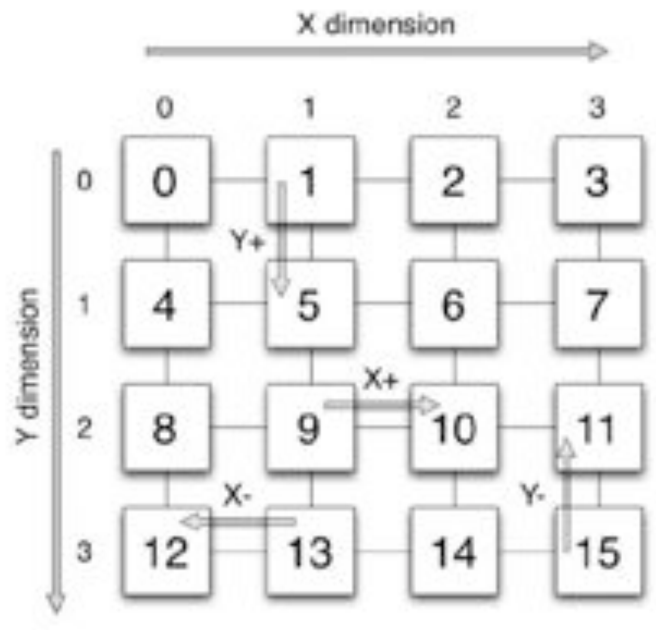


Figure 2.8: Different link crossings in a 2-dimensional mesh.

5, coordinates $(1,1)$. Typically, nodes are numbered by a single id, computed as a function of the coordinates and the number of nodes per dimension. In the case of the example for the 2-dimensional mesh, the value follows this equation: $ID_{Node} = X_{coordinate} + k \times Y_{coordinate}$, being k the number of nodes per dimension. So, in the example in Figure 2.8, node $(3,1)$ has an ID of 7 (k=4).

There are other topologies proposed in the literature to overcome the limitation of 2D meshes. They are later reviewed in this chapter.

2.2.2 Switch Device

As aforementioned, each tile is composed of several elements. The switch is in charge of the communication between the associated node and the rest of the nodes through the network layer. Typically, a switch is made by the following general parts (Figure 2.9):

- *Buffers:* Buffers are a key component of the switch and its design and their position inside the switch affect other aspects of the switch design. The task of a buffer is to store temporarily units of information (typically called *flits*, messages and/or packets). Buffers are tipically associated to

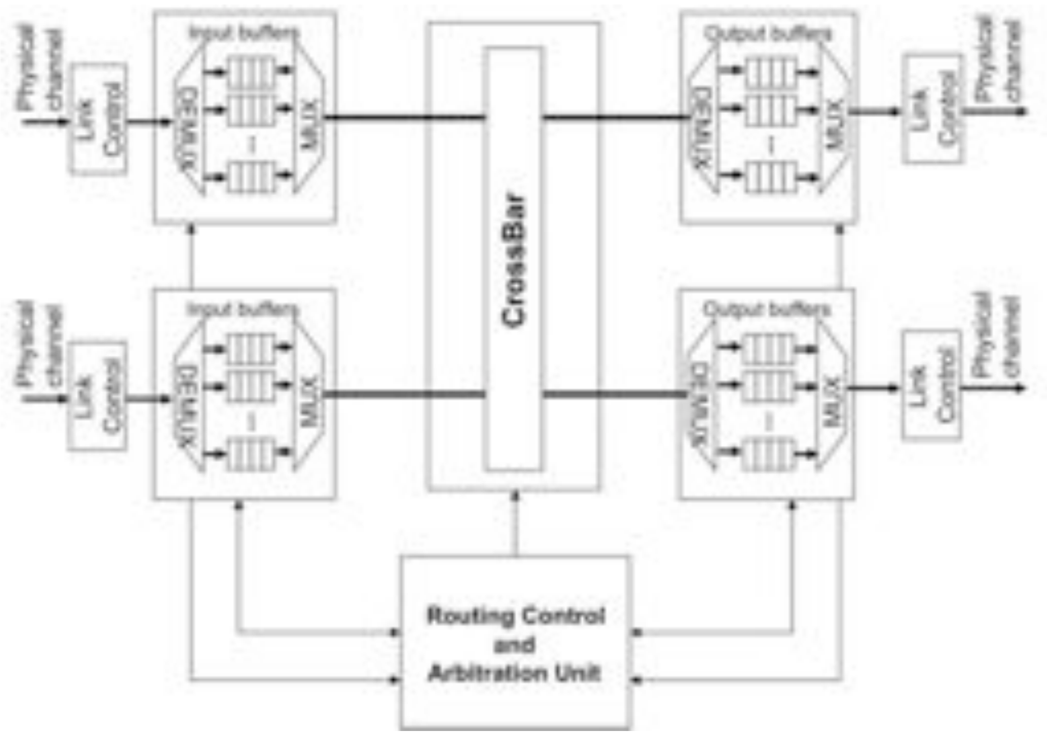


Figure 2.9: Switch architecture.

the channels that are connected to the switch. Channels are accessed through ports, and they are divided in input ports, streams that receive data and are subject to the routing decisions, and output ports, streams the send data to other switches or nodes. Note that, to save area and power, buffers at the output ports are usually not implemented in NoCs.

- *Crossbar:* The crossbar is the switching element and is tipically non-blocking. Crossbars allow the connection between all inputs of the switch to all outputs. Crossbars are classified by their radix, i.e. the maximum numbers of connections they can make. As has been already identified, crossbars do not scale, thus switches with many ports do not scale neither, especially in NoCs.

- *Routing unit:* This unit is the responsible for decoding the unit of information provided by the incoming message, and based on the routing function and destination of the message, computes the most suitable output ports for transmitting the message.

- *Arbiter unit:* This unit reads from the routing unit and configures the

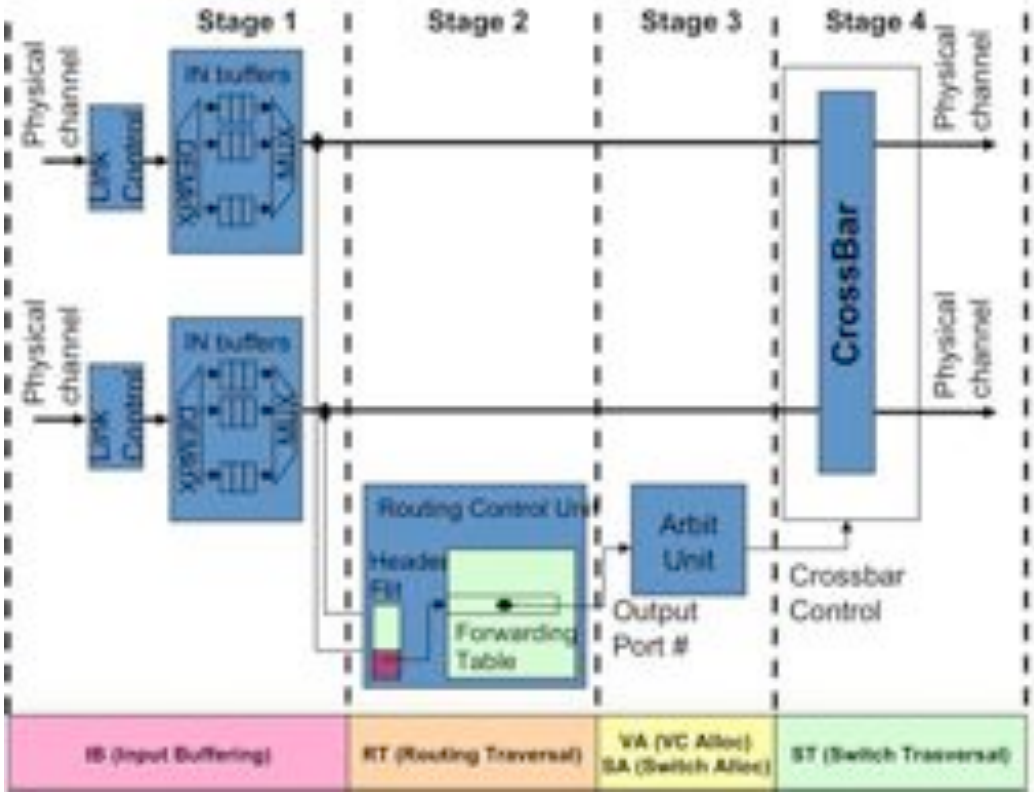


Figure 2.10: Switch stages.

crossbar accordingly to the requests from the input ports to the output ports, taking into account *switching* and *flow control* issues (both will be explained later).

- *Link control:* This component adapts the incoming traffic from the link to the switch. In NoCs, this component is typically omitted (data does not need to be translated).

Pipelining is a typical design method for high-performance switches. The different stages work in parallel with different data streams, thus providing parallelism and, thus, high throughput. A typical pipeline design of a switch can be seen in Figure 2.10 where four stages are shown (IB, RT, VA/SA, and ST). In this thesis we assume this pipeline design.

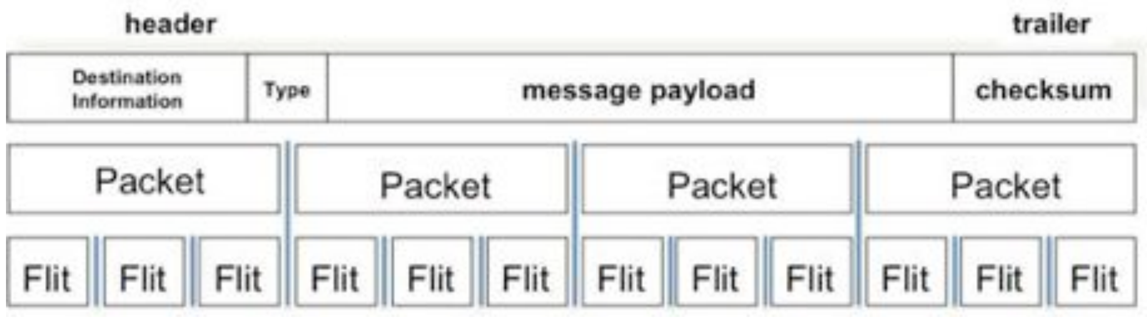


Figure 2.11: Message, packets, and flits.

2.2.3 Data Units

In an interconnection network, the general unit of information between nodes is the *message* (see Figure 2.11). A message is a collection of bits that the sender wishes to transmit to a destination (or a set of destination nodes), i.e. it contains the data that must be transmitted. This information unit, however, due to resource restrictions affected by design choices, may need to be divided into smaller units, called *packets*, through a packetization process (usually performed at the network interface). A packetization process of a message implies some reassembly and order handling at the destination. A packet (or the message) is comprised of a header, which contains the information for routing and control, to be used by the switches, a body which contains the data, and optionally a tail, for flow control. Often, packet and message terms are interchangeable by the community, when both are equal in size. The term packet is usually employed even when the message has not been packetized.

A message is divided further into *flits* (flow control digits), which are the smallest unit of information that is flow-controlled. As the width of the link can be lower than the size of a flit, the flit is further divided at the physical level, into *phits* (physical digits). It is left to the designer and the parameters involved, the size of every unit. However, in NoCs, due to the large amount of bandwidth available, the phit size usually equals the flit size.

2.2.4 Switching

Switching techniques are the responsible for the allocation of network resources to messages/packets inside the switches. Their basic function is to perform the setting of the connections between the buffers of the input and the output ports. The choice imposes several design constraints in the switch that impact the performance, manufacturing cost and power consumption of the elements in the network. Next, we describe the main switching techniques suitable for NoCs.

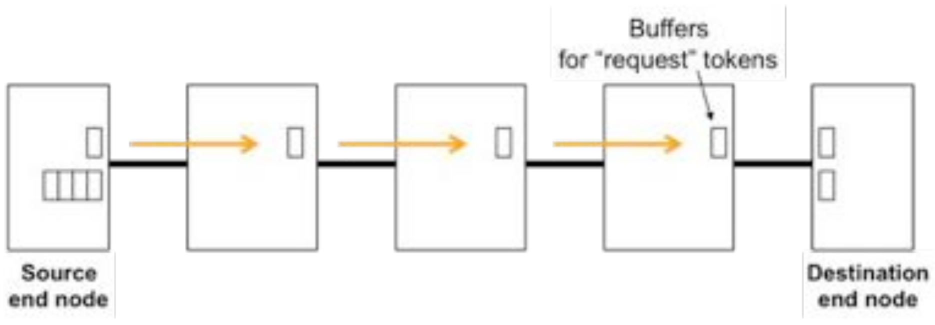
Circuit Switching

In circuit switching (Figure 2.12), the network establishes a reserved path between source and destination nodes prior to the transmission of the message. This is performed by injecting in the network a flit header, which contains the destination end node ID. This header acts as some kind of routing probe that progresses towards the destination node reserving the channels that it gets. When the probe reaches its destination, a complete path between source node and destination node has been set up due to the acknowledgement sent back to the source node. As the path has been reserved for this flow, messages cross the network avoiding buffer needs and collisions with other flows. The circuit is torn down when transmission finishes. An example of a circuit switching-based on-chip network is described in [39].

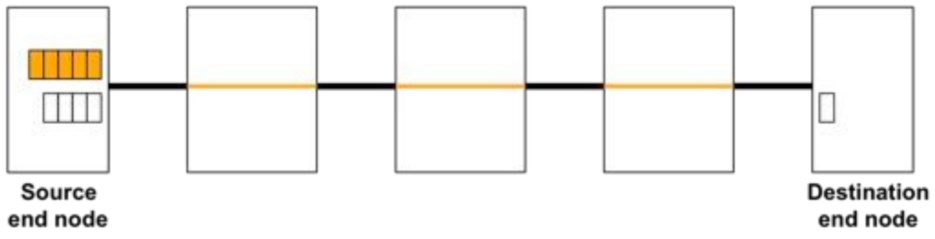
Circuit switching can be very advantageous when messages are very frequent and long. Nevertheless, this switching technique has several important drawbacks. If circuit set up time is long compared to transmission time of the data, it will strongly penalize the performance of the network since links will be poorly used. Additionally, as channels are reserved for a given flow, no other flows can use them even if the connection is idle, thus channels may become even more under utilized.

Store and Forward

Instead of reserving all the path for a certain flow, there are some techniques that operate at packet granularity. These techniques are referred to as packet switching. The most basic technique related to packet switching is *store and forward* (SAF). When a packet arrives to a switch, the switch waits to store the whole packet in its input port buffer before the packet is forwarded. So, input port buffers must be large enough to store a packet (see Figure 2.13).



(a) Request for circuit establishment



(b) Acknowledgment and circuit establishment

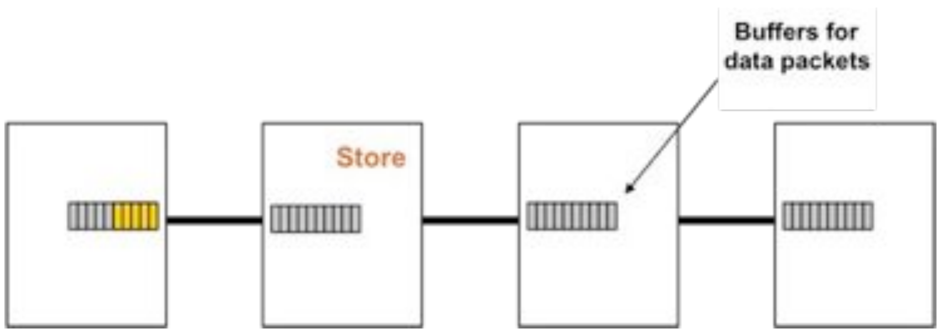
Figure 2.12: Circuit switching.

As can be deduced SAF has larger buffer needs than circuit switching. In addition, latency of packets is multiplicative with hop count along the path (as the forward operation waits for the completion of the store operation).

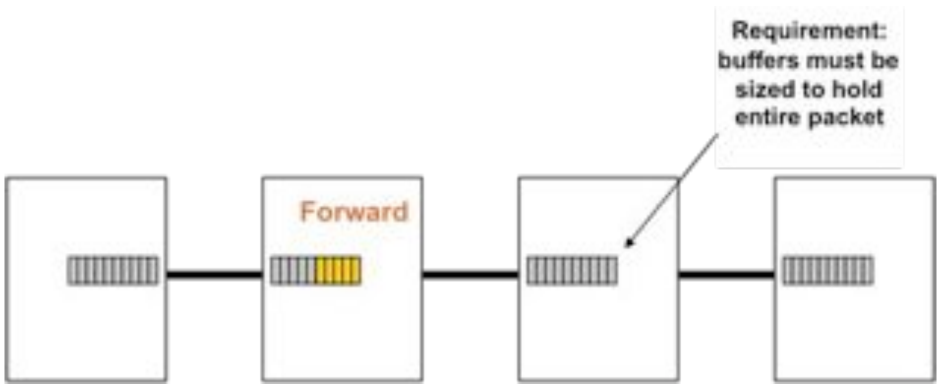
Virtual Cut-Through switching

SAF switching is based on completely receiving a packet before any routing decision is made. But, this is not a very practical decision, since the packet header contains all the required information to perform the routing, and it is physically located at the beginning of the packet (typically in the first flit). So, the routing process can be started as soon as the packet header arrives to the input buffer, without waiting for the rest of the packet. Thus, the packet can be forwarded provided the selected output port chosen by the routing strategy is free. This is what is done in *virtual cut-through* (VCT) switching (Figure 2.14).

In this case, as packets can advance through the switches of the network once the packet header has arrived to each buffer (and has been decoded), the base latency for this switching technique is mostly additive to the distance



(a) Store phase



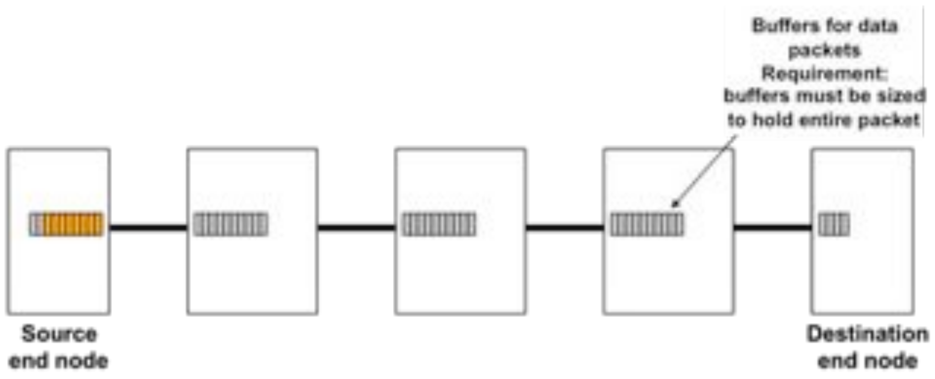
(b) Forward phase

Figure 2.13: Store and forward Switching.

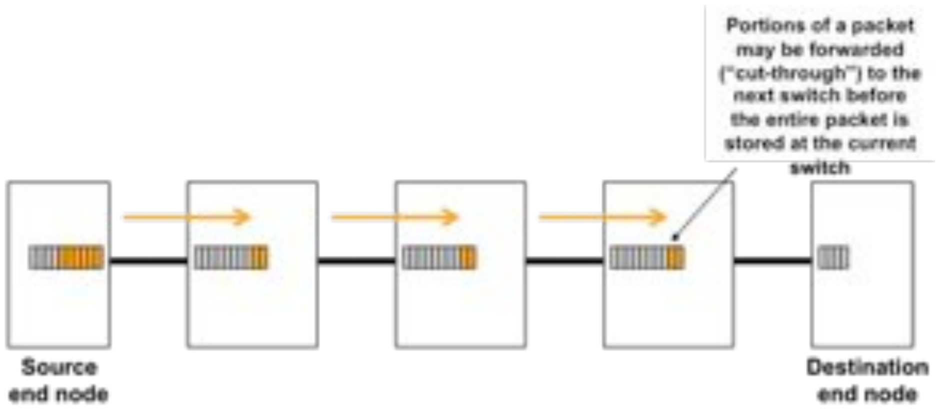
between the nodes (hop count). Despite this, buffer requirements are the same for VCT and SAF. VCT requires there is enough free buffer space to store the entire packet. In fact, VCT behaves like SAF when the output port is busy. The switch needs to completely allocate the entire packet. This is the switching technique commonly used in off-chip high-performance interconnects [8, 11] as buffer size is not as critical as in NoCs.

Wormhole Switching

VCT switching is an improvement over SAF, but in some network architectures, the choice of a buffer size to hold an entire packet could be critical. The requirement to completely store a packet in the buffer of a switch may prevent to design a small, compact, and fast switch [11]. In *wormhole switching* (WH) buffers at the ports of a switch only have to provide enough space to



(a) Packet stored in the source node


Portions of a packet may be forwarded ("cut-through") to the next switch before the entire packet is stored at the current switch
Source end node
Destination end node

(b) Portions of packet being forwarded

Figure 2.14: Virtual cut-through switching.

store few flits, depending on the round-trip time delay (RTT) [1], instead of the whole message. In WH switching (Figure 2.15), the message is forwarded immediately before the rest of the message is entirely received, but as opposed to VCT, there is no need to have enough space for the rest of the message in case the message blocks. In that case, the entire message remains stored through the buffers at several switches. The major advantage of WH switching is the low storage requirements at switches. However, the most important drawback is that WH switching could lead to high contention levels in the network, because a message may block several resources when traversing the network, causing low utilization of links and buffers.

[1]Round-trip time can be defined as the elapsed time between the time a unit of information is sent and the time the acknowledgement of that transmission is received.

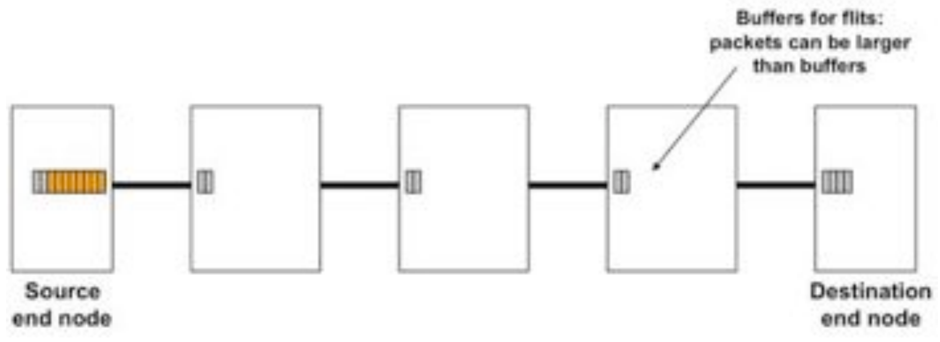


Figure 2.15: Wormhole switching.

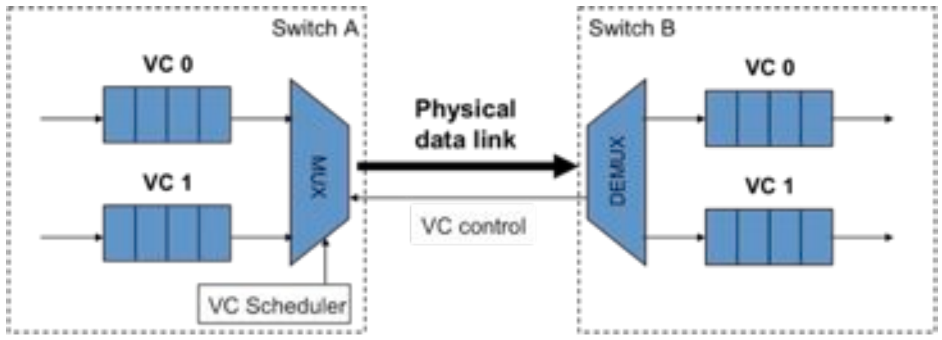


Figure 2.16: Virtual channels.

Virtual Channels

To overcome the contention problem induced by wormhole switching, *virtual channels* [9] were proposed. Buffers basically are operated as FIFO (First-in, First-out) queues. Therefore, if a message reserves the channel but due to the saturation of the network it remains blocked at the current switch, no other message behind this message can use the physical channel even if its requested output port is available. This problem is known as *head-of-line blocking.*

When using virtual channels the buffer at the input port is divided into different virtual buffers and the channel is shared by all the virtual buffers (see Figure 2.16). Of course this virtual multiplexing method requires some local arbitration and must be taken into account by flow control and switching techniques. Virtual channels can be used to improve message latency and network throughput as well. Their major drawback is that the available link bandwidth is distributed over all the virtual channels sharing a physical link, resulting in lower speeds. Again, in the on-chip network domain, the designer must eval-

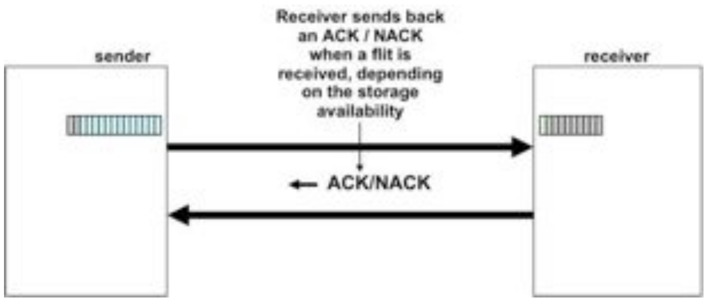

Receiver sends back an ACK / NACK when a flit is received, depending on the storage availability
sender
receiver
← ACK/NACK

Figure 2.17: Ack/nack flow control.

uate the trade-off and the impact overhead on the network. Virtual channels are not restricted to wormhole switching, the concept can be extrapolated to other design choices, depending on the need of their functionality (examples are deadlock-free routing algorithms and quality-of-service protocols).

2.2.5 Flow Control

Transmission of a flit between the input and output ports in a switch is a task performed by the switching technique. *Flow control*, however, is in charge of administering the advance of information through links. Buffers are a resource where to temporarily store flits, but they are finite. Flow control protocols are in charge of determining when the flits can be forwarded evaluating the capacity of the buffers and the link bandwidth. The main goal of flow control mechanisms is to avoid flits being dropped due to the lack of buffer resources to store them.

There are mainly three flow control mechanisms that are commonly used: *ack/nack*, *stop & go* and *credit-based*. The ack/nack flow control mechanism (see Figure 2.17) is based on data acknowledgements. When a flit arrives to a buffer, if the buffer has space available, then the flit is accepted and an acknowledgement signal (ack) is sent back. Instead, if there is no space available, the flit is dropped and a negative acknowledgement (nack) is sent. The flit must be retained at its origin until it receives a positive acknowledgement.

Stop & go emerged as an alternative to reduce the signalling (control traffic) between the sender and the receiver. Stop & go flow control (see Figure 2.18) is based on every buffer having two thresholds corresponding to certain sizes computed from the round-trip time. When the space occupied in the

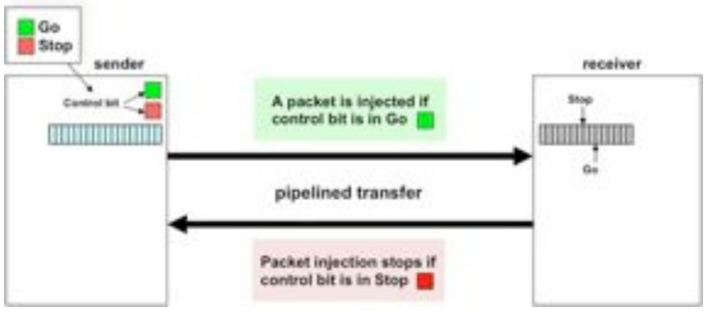

Go
Stop
sender
Control bit
receiver
A packet is injected if control bit is in Go
Stop
Go
pipelined transfer
Packet injection stops if control bit is in Stop

Figure 2.18: Stop & go flow control.

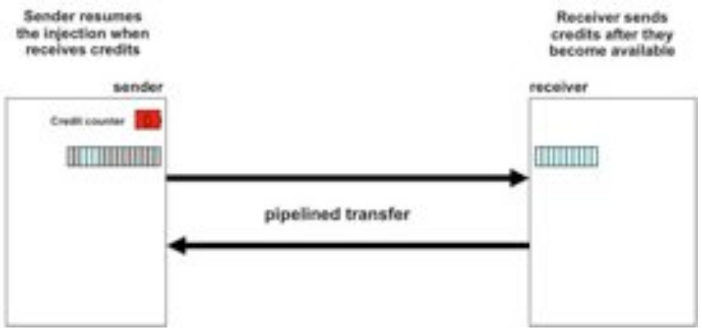

Sender resumes the injection when receives credits
Receiver sends credits after they become available
sender
Credit counter
receiver
pipelined transfer

Figure 2.19: Credit-based flow control.

buffer reaches the *stop* threshold, a signal is sent back to the sender precisely to stop the transmission, taking into the account that enough buffer space still remains for the flits that are still being transmitted on the fly by the sender. When the buffer occupancy diminishes under or equal to the second threshold, *go*, then another signal is sent to reactivate the transmission of flits.

With credit-based flow control (see Figure 2.19), each sender, at its end of the link, maintains a count of credits, which is equal to the number of flits that can still be stored at the buffer on the receiver side. Whenever a flit is forwarded to the receiver buffer, as it occupies a slot, then the counter is decremented. If the counter reaches zero, it means that there is no available buffer space at the other end, and no flit can be forwarded. On the other hand, whenever a flit is forwarded and frees the associated buffer space, a credit is sent back to increment the counter. The drawback of this flow control mechanism is the significant amount of credit signalling sent backwards, which could impact on network performance.

2.2.6 Arbitration

A switch is composed of multiple input and output ports with their associated buffers and channels. Multiple inputs, according to routing decisions, may request the same output port. In this scenario, an arbitration operation is required to decide which one of the requests is allowed to connect to the output port. The arbitration mechanism must ensure to assign the output to only one of the inputs that have requested it, and the others must wait until they are allowed. As the arbitration operation introduces a latency to determine the assignment of the different output ports, it is critical for a network on-chip environment that these operations are performed fast enough to keep low latencies.

The main goal of an arbitration mechanism is to provide fairness between all the ports while achieving maximal matchings between requests and resources. Although there are many proposals for arbitration algorithms and implementations, we can distinguish two general arbitration techniques that differentiate on how they assign priorities between the requestors and the resources.

The first one is *fixed priority*. An arbiter with fixed priorities grants the requests in an established order to the different input ports. This order is determined by the priority assigned to each input port. In this mechanism, the arbitration is simple, but introduces unfairness and potentially, starvation. If one of the input buffers with higher priority keeps requesting the associated output, the inputs with lower priority get blocked, even, inducing the chance that the inputs with low priority never get their requests satisfied.

The second one is called *round-robin*. An arbiter that implements round-robin arbitration cycles priorities between all the input ports by assigning the lowest priority to the input port which request was last served. This arbitration technique introduces fairness between the requestors, but is more complex to implement. In this thesis we assume round-robin arbitration policies. For further arbitration mechanisms and policies, please refer to chapters 2 and 3 of [12].

2.2.7 Routing

As we have described before, topology defines the physical organization of the network composed by the nodes. In fact, a given topology defines the available paths between all the nodes. The routing algorithm is the responsible of deciding which path has the message to follow to be effectively routed from its source to its destination. The choice of the routing algorithm becomes of outmost importance in the network performance. Indeed, in the on-chip network domain not all solutions from the off-chip network domain are suitable due to environmental restrictions. The designer must find a trade-off between efficiency, flexibility and implementation cost of routing.

Implementation Types

Although any implementation is specific to the nuts and bolts of the technology, there are three main trends to implement the routing strategy.

The first one is *logic-based* routing. This kind of routing is the result to translate a logical or arithmetical function of a routing algorithm into the equivalent in circuitry inside the switch. So, when the message header is decoded at the input buffers, the output port is computed based on the hardware that represents the routing function. Logic-based routing is a good design choice in terms of delay, area, and power consumption. The main drawback is its lack of flexibility as these implementations could become non-functional if the topology changes due to manufacturing defects, just to name a reason.

Alternatively, *routing tables* are basically composed of row-like structures that match destinations with table entries. So, given the destination for a certain message, there is some circuitry associated that decodes this information, and accesses the routing table to find the routing decision associated to that destination. The most conventional way to implement these tables is to use memory structures. The advantage of table-based routing is flexibility, as the information of routing decisions stored on routing tables could be the answer of more complex routing algorithms, that are not only based on logical or arithmetical assumptions. On the other hand, routing tables implementation suffers from scalability, area, power consumption, and latency problems. For example, there is a penalty time (that increases with table size) associated to accessing memory structures.

Also, *source routing* [51] is a method that allows moving a packet through a network in which the entire path is predetermined by the source. The path information is placed in the message header. When the message arrives at a switching device, no forwarding decision is necessary. The device looks at the path information in the packet header to determine the port to forward the packet. Source routing assumes that the source knows about the topology of the network, and can therefore specify a path. However, it is not always possible to expect the system's logic to learn a network's topology. Source-based routing requires larger headers in size wasting bandwidth, thus becoming a non-scalable solution.

Deadlock

A deadlock occurs when a message cannot advance toward its destination because the buffer requested by the message is full, being blocked by another message that is also waiting, all of them cyclically waiting. A cyclic set of such events could make the messages to be blocked permanently because each message involved in the situation requests a resource held by another one. As no one message will advance before getting its requested buffer granted we get a deadlock situation. That is the case in Figure 2.20 where four messages allocated in different input buffers request the buffer at the next hop cyclically. As those buffers are full, no one can get credits for the buffer and messages are blocked forever.

Schemes Classification

Regardless on how they are implemented, there are different taxonomies to classify routing algorithms. In the following paragraphs we provide a discussion for the most well-known types of routing algorithms which we use in this dissertation, mainly deterministic and adaptive routing algorithms.

In a 2D-Mesh topology, the most efficient routing algorithm (in terms of implementation complexity and power consumption) is dimension-order routing (DOR). The deterministic algorithm, DOR, is implemented on every switch and requires small logic blocks. The message first moves in the X dimension,

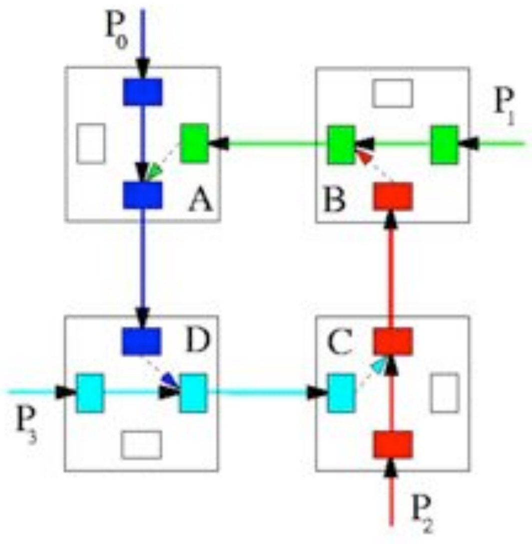


Figure 2.20: Deadlock situation when using XY and YX routing at the same time.

and once it reaches the destination column, then it moves through the Y dimension, always following minimal paths. The low complexity of DOR makes it very appealing for network on-chip (NoC) designers.

As commented in the previous chapter, DOR is not flexible, not allowing more than one path for every source-destination pair. Also, DOR does not tolerate a single failure without disconnecting several pairs of end nodes. Besides, DOR may lead to congestion in the network and it does not support alternative paths to alleviate congestion.

An alternative to DOR is the use of adaptive routing. In such a protocol, switches are able to select different output ports for the same destination, depending on the current status of those ports. Thus, local congestion can be alleviated by the use of adaptive routing. Typically, minimal paths are supported by adaptive routing, thus, the message gets closer to its destination for every hop. To avoid deadlocks, an acyclic escape path is implemented as a different virtual channel. If adaptive output ports are not available, then the escape path is taken [10].

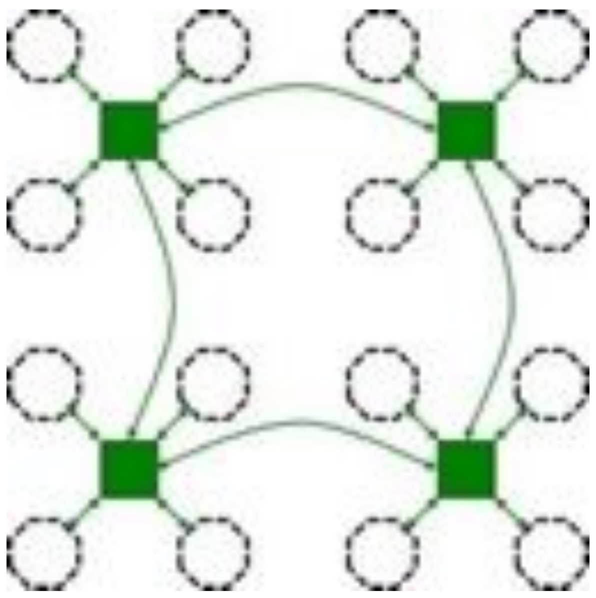


Figure 2.21: Concentrated mesh.

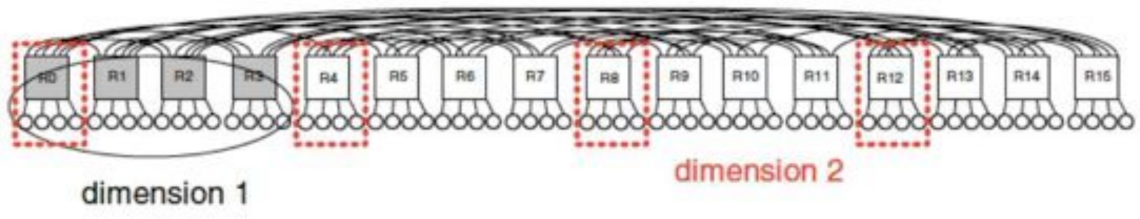
2.3 Related Work

In this section we deal with the related work for topologies and power saving strategies. These are two fields this thesis is focused on, and where this thesis tries to contribute. The related work for power gating strategies will be focused on techniques that try to switch off unused components.

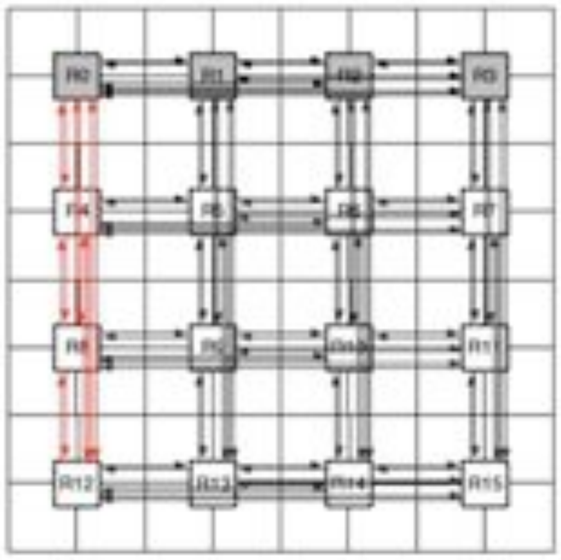
During the last years, different topologies for CMPs have been proposed in the literature. Initially, designs and proposals relied on rings [30] and 2D-Meshes [38], [35], [36]. From that moment, efforts focused on reducing hop count, thus reducing latency. This is the case of the C-Mesh [2] (Figure 2.21) and the flattened butterfly network [18] (Figure 2.22).

A concentrated mesh has a smaller diameter and area footprint than 2D-Mesh that results from improved resource sharing. While concentration is a key element in the design of scalable networks, it is not sufficient by itself when high traffic or congested situations arise.

The flattened butterfly is an update of an architecture known as a butterfly, which has been around since the 1960s. The name comes from the pattern of inverted triangles created by the interconnections, which looks like butterfly wings. Dally flattens the butterfly by combining columns of routers and linking each router to more processors. The new configuration halves the number of



(a) 1-Dimensional view



(b) 2-Dimensional view

Figure 2.22: Flattened butterfly.

router-to-router connections. Data traveling between the processors can now get to any other processor in fewer hops, even though the physical route may be longer, and that eliminates considerable latency.

The flattened butterfly can reduce the maximum number of hops. Unfortunately, the flattened butterfly is not truly scalable, as the channel count highly grows with the number of end nodes. In addition, the use of a large number of dedicated point-to-point links and the resulting high degree of wire partitioning leads to low channel utilization, even at high injection rates.

Other works [13] reduce hop count by relying on the concept of express channels where a switch is connected to several switches along each direction in the 2D-Mesh [7]. A complete analysis and comparison of several topologies mentioned in this section can be found in [13]. Besides, they propose a

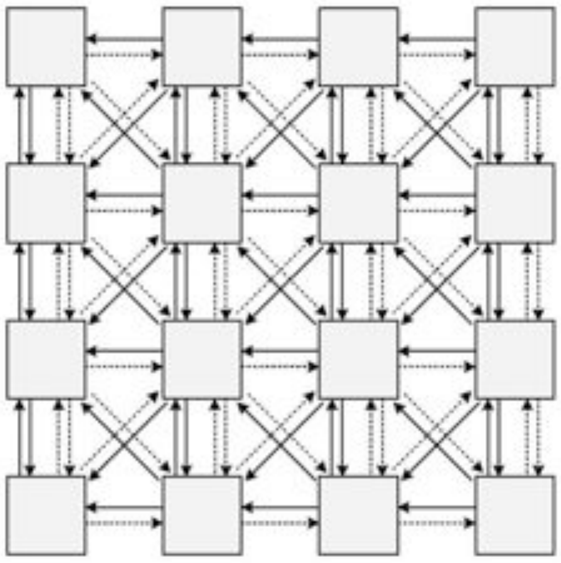


Figure 2.23: Diagonal Mesh.

new topology, called Multidrop Express Channels (MECS), that uses a one-to-many communication model enabling a high degree of connectivity in a bandwidth-efficient manner. In a 64-terminal network, MECS enjoys a 9% latency advantage over other topologies at low network loads, which extends to over 20% in a 256-terminal network.

One special case is the Diagonal Mesh (DMesh) topology proposed in [15]. See Figure 2.23. In such topology diagonal links are added between switches. However, all these topologies (including DMesh) rely on the fact that every end node is connected only to one switch. In this thesis, we take a different approach and connect nodes to multiples switches, thus providing much more flexibility.

Other works try to improve the performance by improving the switch architecture. Examples are [28] and [4]. In the first work, authors propose multiplane virtual-channel router which has multiple crossbar switches and a modified switch allocator is proposed to enhance the latency and throughput performance. The second work proposes and justifies the replacement of virtual channels by replicated channels, based on the abundance of wires expected in current and future deep sub-micron technologies.

In [2] authors also hint the use of two concentrated meshes for performance issues. Therefore, that contribution could resemble one of ours pro-

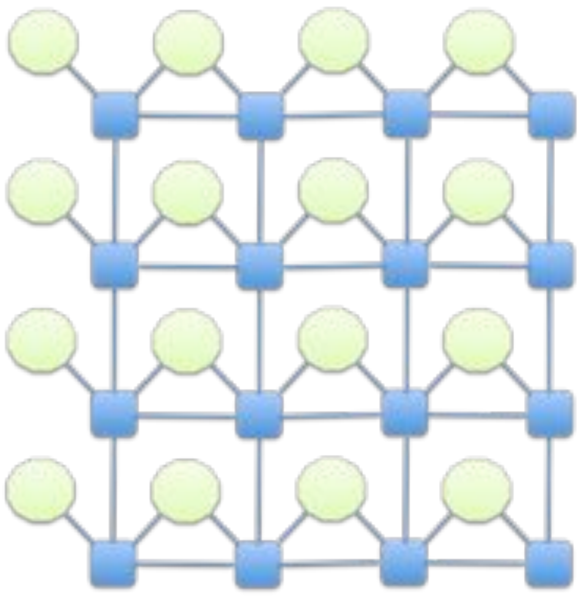


Figure 2.24: End nodes connecting two switches. Circles are nodes and squares are switches.

posals. However, there are major differences. The first one relies on the fact that in [2] no details are given to the network interface connected to different switches. This is of major importance as it may lead to excessive performance bottlenecks and power consumption issues. Secondly, in [2] only two concentrated meshes are used, instead of four in the PC-Mesh topology. Besides, the connection pattern between nodes and networks is not clear at all in this approximation.

In [43] (see Figure 2.24) an end node is connected to two switches in a 2D mesh configuration. The main goal of such approach is only for fault-tolerance purposes. In our case, our goal is to reduce power consumption without degrading performance and having a higher degree of fault tolerance. Besides, in our proposals in this thesis every end node is connected to four neighboring switches each belonging to a different independent sub-network. Every end node is connected to a varying number of parallel networks. We solve the heterogeneity problem by attaching every end node to exactly 4 networks.

Power gating (gated-Vdd) is a well-known technique to reduce static power consumption. In [31] a circuit technique was proposed to disconnect (by using a gating transistor) the power supply. Power gating can be applied at different levels, from complete execution units [16] down to single SRAM cells [6]. For NoCs different works applied the power gating technique. In [6] buffers are power gated and different policies are proposed. In [24, 25] power gating is applied to virtual channels. In [34] powering down links is proposed. In [14] static power consumption is reduced by using the concept of on/off links [34] with power-aware buffers [6, 24, 25]. The proposed power management algorithm also uses the concept of on/off links, where the voltage is dynamically adjusted in response to the network utilization. The different proposals for power gating usually rely on a standard 2D-Mesh topology, and potentially can be applied to every studied topology.

In this thesis, the HPC-Mesh contribution (which improve all the previous proposals) uses concentrated meshes dynamically. All the end nodes are connected to all the concentrated meshes. As a major difference to the previous works in the current state of the art, we evaluate a power-aware selection algorithm together with an on/off switch mechanism both working cooperatively, as the target is power consumption savings without end-to-end latency penalty, using a negligible and intelligent selection function in every end node.

As one of the proposals will be targeted for the 3D stacking concept in the HPC-Mesh topology, here we briefly introduce the state-of-the art regarding 3D NoCs. In [37] a trade-off in 3D systems on-chip between cost and performance is studied. Authors demonstrate in [29] how a 3D NoC overcomes a 2D NoC topology. In [21] authors propose vertical links through silicon vias (TSVs) and a flow design for 3D simulations while in [5] authors propose several techniques for traffic and thermal management aware in a 3D NoC. In [42] different policies for run-time thermal management are proposed. In [20] the authors share TSVs among neighboring switches to decrease the number of vertical links and in [32] a serialization of TSVs contributes to reduce also the number of vertical links and therefore, the manufacturing cost and the peak temperature.

2.4 Conclusions

In this section we have introduced the basic concepts of on-chip interconnection networks, starting from design factors, followed by the kind and examples of current topologies, tile-based designs, and the most important components in the networks as switches and links.

Also, we have referred the main related work existing for topologies and power gating mechanishms. In the next chapter, we begin with the first contribution called NR-Mesh (Nearest neighboR Mesh).

Chapter 3

NR-Mesh Topology

In this chapter we introduce the concept of using different injection and ejection ports for the end nodes. This is the main philosophy of this dissertation and, different proposals will emerge.

We first propose in this chapter a flexible network on-chip topology, referred to as NR-Mesh (Nearest neighboR Mesh). The topology allows for several alternative paths for most source-destination pairs. Thus, a message can be injected through different switches and can be received from different ones. This capability leads to higher flexibility when routing messages, not only at switches but also at network interfaces. The resulting topology has a lower diameter than the 2D-Mesh topology and provides efficient support for collective communication. Fault-tolerance is also increased with the new topology.

In parallel with the definition of alternative topologies, we tackle the power consumption issue in current on-chip networks. Complementary to the new topology, we propose a routing algorithm that is aware of network components that are switched off, thus saving static power consumption. The algorithm will maximize the time network components are switched off, thus maximizing power savings. The algorithm is based on the adaptive routing algorithm, with proper modifications to the selection function for the output ports.

In the first section we describe the NR-Mesh topology, focusing on its injection algorithm, routing algorithm, and implementation details. Then, we describe the complementary power management algorithm codesigned with

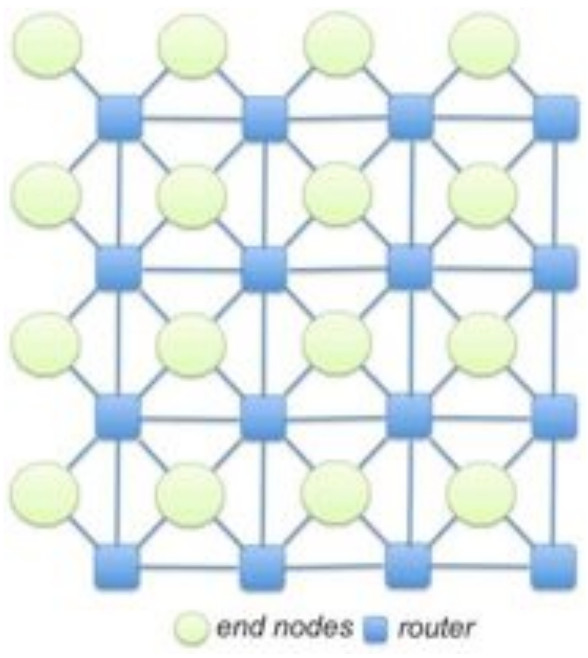


Figure 3.1: NR-Mesh on-chip network topology.

the routing algorithm. Then, we analize the topology and how it performs when coupled with the power management strategy.

3.1 NR-Mesh

Figure 3.1 shows the connection pattern of the topology between end nodes and switches. The network uses four links to connect every end node to four switches. Nodes at the boundary of the topology are connected, however, to fewer switches. In particular, the node at the top left-most corner is connected to a single switch and the remaining boundary nodes are connected to two switches.

Note that when compared with the 2D-Mesh topology, this topology offers a higher connectivity. It provides alternative paths as messages may be injected through up to four different switches and be received at final end nodes from up to four different switches. It is very important to emphasize that in-transit messages, however, cannot cross in-transit end nodes (end nodes are not used as switches).

A key property of the topology is the reduced diameter it provides. For example, NR-Mesh with 16 end nodes has the maximum distance (between the

Size	NR-Mesh D.	NR-Mesh B. BW	2D-Mesh D.	2D-Mesh B. BW
4x4	5 hops	8 flits/cycle	7 hops	8 flits/cycle
8x8	13 hops	16 flits/cycle	15 hops	16 flits/cycle
16x16	29 hops	32 flits/cycle	31 hops	32 flits/cycle

Table 3.1: Diameter (hops) and bisection BW (flits/cycle) comparison between NR-Mesh and 2D-Mesh topologies. Links with 1 flit/cycle bandwidth are assumed.

two most distant corners) of five switch hops while the 2D-Mesh has a distance of seven switch hops. This will reduce the average latency of messages, the execution time of applications, and the power consumption in the network.

Table 3.1 compares the diameter and bisection BW for the NR- and 2D-Mesh topologies for different sizes. Notice that bisection bandwidth is the same. However, in NR-Mesh there are certain number of nodes that are connected to both sides of the bisection, thus alleviating its traffic.

3.1.1 Tile-Based Design

The NR-Mesh can be adapted to a tile-based design. Figures 3.2 shows a possible example of a tiled organization where the NR-Mesh topology is supported. As can be seen in the figure, the switch is located at the bottom right-most part of the tile. The switch is connected to four different end nodes, three of them in a different neighboring tile.

Each end node includes the processor, the L1 data and instruction cache, a slice of the L2 cache, and a memory controller (potentially reaching a memory module if 3D stacking is used). Since the end node is connected to four switches, additional logic is required at the network interface of the switch to decide which output port to use. We have implemented and measured the required logic (Section 3.1.2) and we have obtained negligible overheads, except for the longer internal link required in the NR-Mesh which we have into account. The associated control logic is included at the end node (see Figure 3.2). Later, we will describe the algorithm used to control the selection input port function or *Select path* in Figure 3.2.

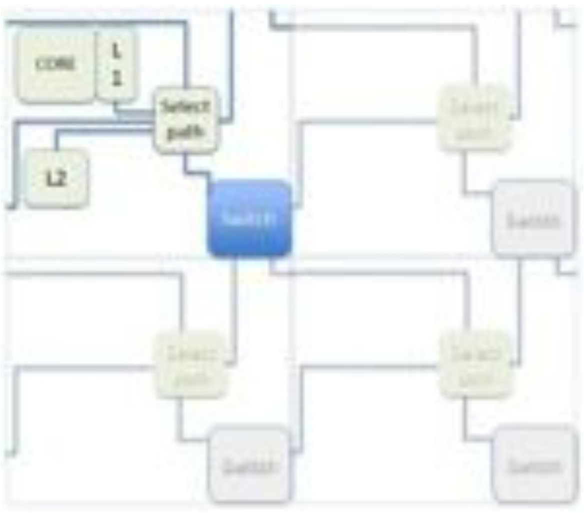
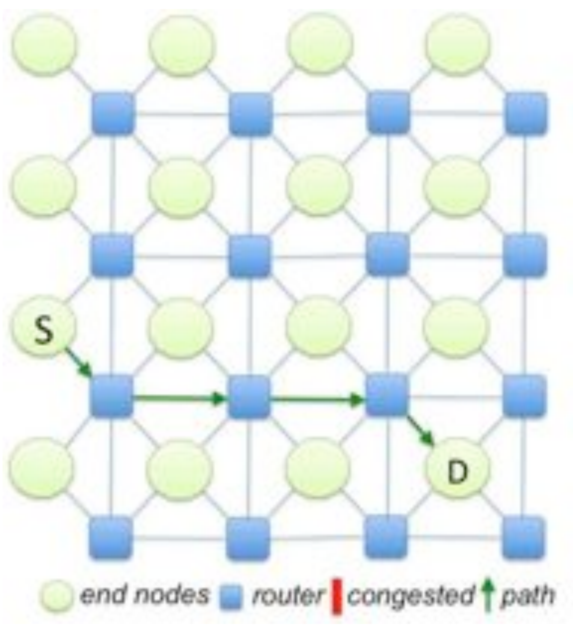


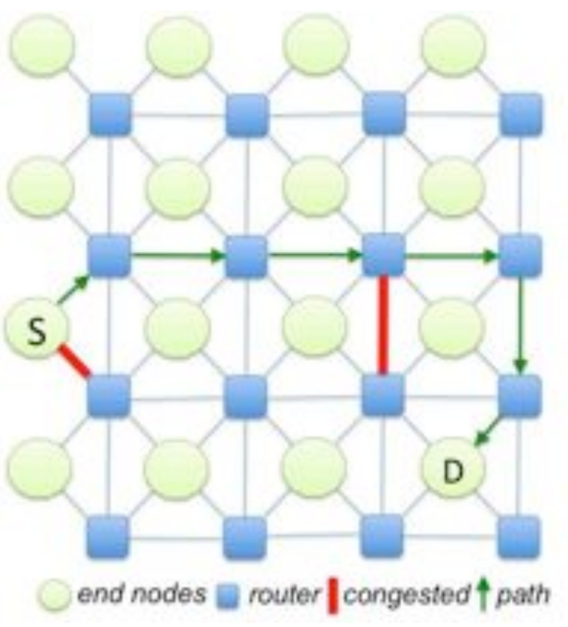
Figure 3.2: Tile design for the NR-Mesh topology.

Figure 3.2 shows the links connecting end nodes to switches. For links connecting end nodes, the figure shows how such links cross the tile boundary in order to reach other switches located at neighboring tiles. For links connecting switches among them, the usual 2D-Mesh link layout is used (not shown in the figure).

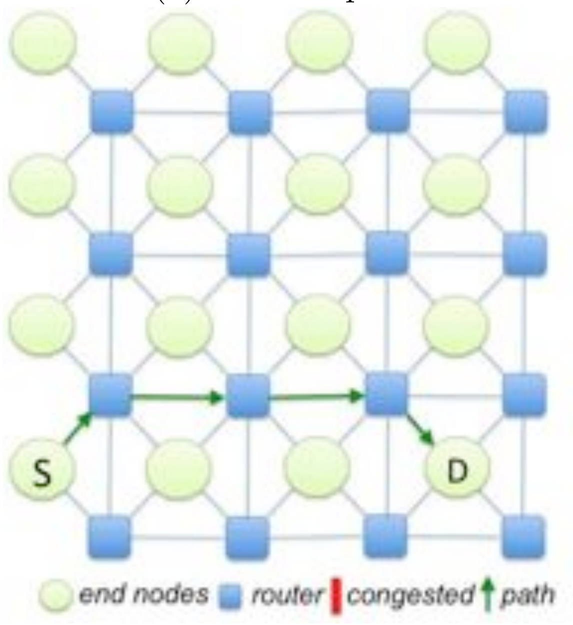
Notice that a tile-based design has the same number of end nodes and switches. This results in higher connectivity since only one end node is connected to a single switch. For the 16-tile design, nine end nodes are connected to four switches, six end nodes connected to two switches, and one end node connected to one switch. Indeed, the last row and column of switches could be removed without affecting connectivity. However, these switches provide flexibility when routing messages and will be in off mode most of the time (in low traffic conditions). We can see several examples showing the flexibility of the NR-Mesh in a non-congested and congested situation. In Figure 3.3.(a) we can see how a message can be sent through a minimal path from the end node S to the end node D. This is the desired case as minimal a path is enforced. However, in case of congestion we can use non-minimal paths, even at the injection, as Figure 3.3.(b) shows. In addition, different minimal paths can exist, as shown in Figures 3.3.(c) and 3.3.(d). In this case, the source and destination nodes are at the same row and have two alternative paths, both minimal. This flexibility is not available in the original 2D-Mesh network.



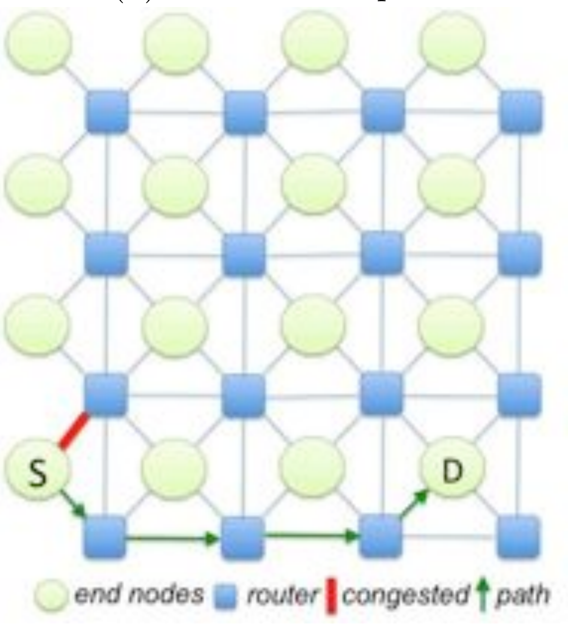
(a) Minimal path.



(b) Non-minimal path.



(c) Minimal path in the same row.



(d) Minimal path using the last row.

Figure 3.3: Using alternative paths in the NR-Mesh topology depending on traffic conditions.

3.1.2 Injection Algorithm

The injection algorithm is a key component of the topology. A message can be injected into the network through different ports. Notice, however, that depending on the final end node destination, some of the injection ports will

```
function injection-algorithm(dest) : port
var   p: port
var   mp: port_list
var   nmp: port_list
begin
  mp = min_av_ports(dest)
  nmp = non_min_av_ports(dest)
  p = random(mp)
  if (p == nil)
      p = random(nmp)
  end
  return p
end function
```

Figure 3.4: Injection algorithm for the NR-Mesh topology.

lead to minimal paths and others will lead to non-minimal (longer) paths. In order to provide full flexibility, the injection algorithm considers all the possibilities, although prioritizing ports that provide minimal paths. The proposed injection algorithm is shown in Figure 3.4.

The algorithm first creates a set of output ports that lead to minimal paths (*mp*; *min_av_ports* function) and a set that leads to non-minimal paths (*nmp*; *non_min_av_ports* function). These functions filter the ports that are not available. A port is not available if a message is already being injected through that port, or if no buffer (flow control) is available for the new message. According to this, these functions may return an empty list (none of the minimal or non-minimal ports are available).

Figure 3.5 shows two examples. At $S1$, for messages addressed to $D1$, two ports are included in the *mp* set, and two ports in the *nmp* set. For messages being injected at $S2$ and addressed to $D2$, only one port is included in the *mp* set and the remaining three ports in the *nmp* set. Notice that if none of the ports are available then, they are not included in any of the sets.

Once the set of ports is computed, the algorithm randomly selects an

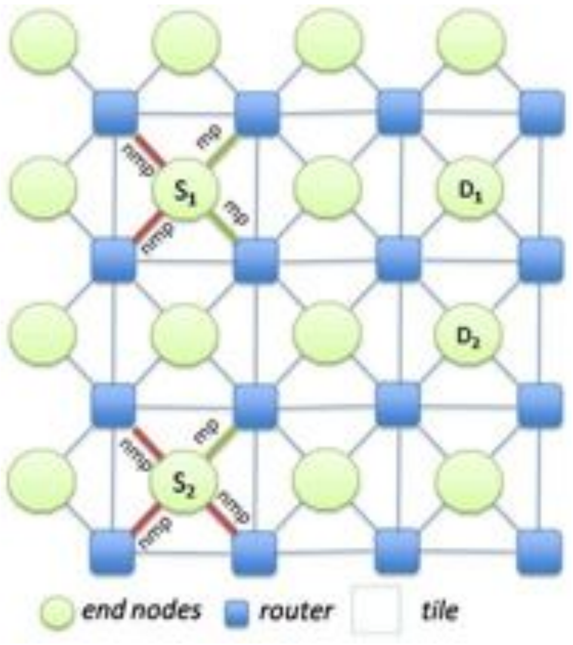


Figure 3.5: Example of minimal and non-minimal injection ports for the NR-Mesh topology.

output port. The function *random* chooses one output port among all the available ports. If there are no available ports leading to minimal paths, then a non-minimal path is randomly selected, always prioritizing shorter paths. Notice that the injection algorithm may return an empty result (p being *nil*). In that case, in the next cycle, the injection algorithm is checked again.

There are different methods to compute which set an injection port belongs to (either minimal path or non minimal path set). Each injection port of the end node must be labeled with its direction (North East NE; South East SE; South West SW, and North West NW). By comparing the coordinates of the destination node and the source node (not the switch coordinates), the direction of the destination node (with respect to the source node) can be obtained, being one of the following ones: NE, E, SE, S, SW, W, NW, N. Now, matching the direction of each output port and the direction where the destination is located provides an easy way of classifying each output port into minimal and non-minimal ports. For instance, an output port in NE direction is minimal whenever the destination end node is located either at N, E, or NE direction. On the contrary, the injection port must be labeled as non-minimal one. As deduced, this algorithm is straightforward and simple to implement.

3.1.3 Network Routing Algorithms

The previous section described the injection algorithm. In this section, we deal with the routing algorithm inside the network. Note that different routing algorithms are suitable for the new topology. If we remove the links connecting end nodes to switches, then we are left with a 2D-Mesh topology. Thus, deterministic routing, like DOR, can be used. Notice that the DOR algorithm is deadlock-free, as its channel dependency graph (CDG) is acyclic. Since Y $\rightarrow$ X transitions are forbidden, no cycles can be formed. Deadlock-freedom is guaranteed even if messages are injected to or extracted from the network through different locations. A source end node injects messages through four different switches, which is equivalent to four different end nodes injecting messages in an original 2D-Mesh network. The same applies to delivering a message to an end node from up four different switches.

However, the deterministic routing algorithm can be extended by deciding where to eject a message. Although DOR only provides one path per pair of end nodes, when applied to NR-Mesh, several alternative paths are now available. Specifically, there are up to 16 alternative paths given by four injection points and four ejection points. Figure 3.6 shows the average number of alternative paths per each communicating flow when assuming DOR routing in both 2D-Mesh and NR-Mesh topologies. As can be seen, as the topology gets larger, the number of possible paths approaches asymptotically 16, as the relative number of nodes with four injection/ejection ports increases in percentage. The figure also shows the average number of injection/ejection ports per end node. Although in a 4×4 NR-Mesh the number of alternative paths is small (around 4), we will see in the evaluation section that it is enough to reduce power consumption in the network without affecting performance.

As previously said, messages at switches can choose among different output ports based on the current port's status. Figure 3.7 shows the case of a message being routed from end node S to end node D. The message encounters a port at switch X (the north port) that is congested (busy). As the destination end node is connected to two switch columns, the message can be forwarded to the east switch and then move north. Next, the message reaches switch Y that is considered a destination switch (a switch where the destination end node is connected to). The message can leave the network through that switch or,

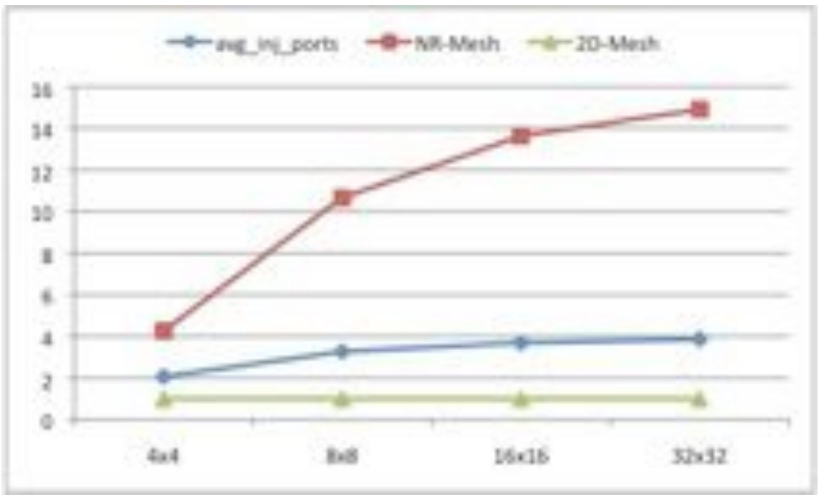


Figure 3.6: Average number of paths assuming DOR in NR-Mesh and 2D-Mesh networks. The figure also includes the average number of injection ports per node for the NR-Mesh topology. In the 2D-Mesh the number is one.

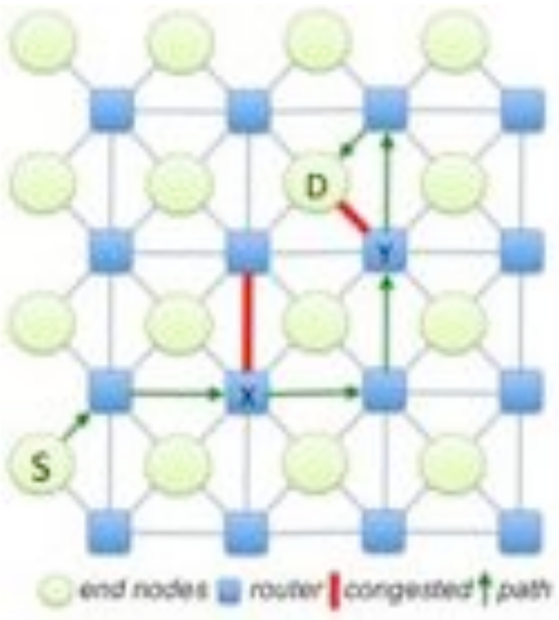


Figure 3.7: A non-minimal path in the NR-Mesh topology.

alternatively, if the output port is busy, it may visit the next switch (located to the north) and reach the end node through that switch. Notice that both alternative hops lead to non-minimal paths. Thus, such actions are performed only if the minimal paths are busy/congested. Note, however, that no Y-X turns are taken, therefore, deadlock-freedom is preserved.

```
function deterministic_routing(iport, cur, dest) : port
var    p: port
var    nmp: set of ports
begin
  p = minimal_xy_port(cur, dest)
  if (free[p]) return p
  if (iport == west and y(dest) < y(cur) and x(dest) == x(cur)+1) nmp = nmp + east + north
  if (iport == west and y(dest) > y(cur) and x(dest) == x(cur)+1) nmp = nmp + east + south
  if (iport == east and y(dest) < y(cur) and x(dest) == x(cur)-1) nmp = nmp + west + north
  if (iport == east and y(dest) > y(cur) and x(dest) == x(cur)-1) nmp = nmp + west + south
  if (iport == south and dest at north) nmp = nmp + north
  if (iport == north and dest at south) nmp = nmp + south
  return priority_select(nmp)
end function
```

Figure 3.8: Deterministic routing algorithm for the NR-Mesh topology.

The deterministic routing algorithm with support for non-minimal paths is described in Figure 3.8. The function is run on every switch and for every incoming message. The algorithm takes into account the input port that received the message (*iport*), the current location of the switch (*cur*), and the destination node (*dest*) of the message.

Figure 3.9 shows the node IDs and the switch IDs assumed by the routing algorithms. This is an important issue for the routing algorithms as the end nodes can be reached from different switches. The messages include in their headers the destination end node ID, which is used together with the switch IDs.

The first step of the algorithm is to find the output port using the DOR routing algorithm (function *minimal_xy_port*). Based on the current coor-

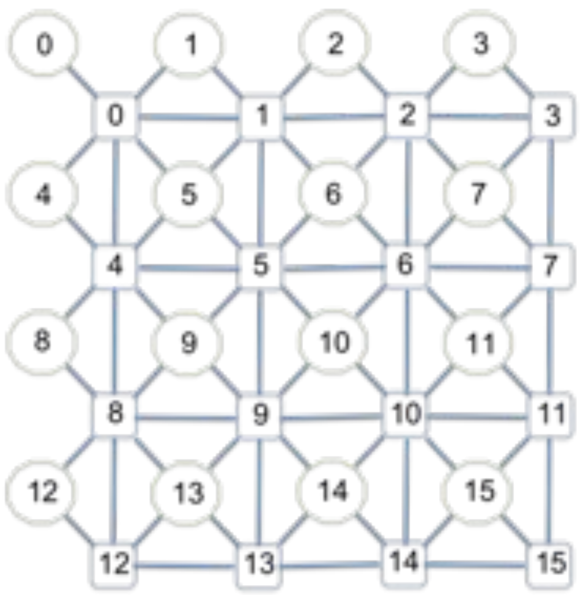


Figure 3.9: Node and switch IDs assumed by routing algorithms and messages in the network.

dinates and destination coordinates, the calculated port will be *north*, *east*, *west*, *south*, or *internal*. Notice that only one minimal port exists for every message at every switch, regardless of the destination of the message. However, in order to obtain the correct destination switch ID from the destination node ID, some slight modifications are needed. For instance, from Figure 3.9 we can deduce that a message sent from node 9 through switch 9 and addressed to node 11 should reach switch 10 instead of switch 11. Thus, from the destination node identifier, the switch needs to obtain the destination switch ID to proper compute the XY output port. The following precomputation is performed in the current switch:

```
dest_sw = dest_node
if x(cur)<x(dest_sw) x(dest_sw)=x(dest_sw)-1
if y(cur)<y(dest_sw) y(dest_sw)=y(dest_sw)-1
```

Notice that dest_sw is the ID of the destination switch where the destination node is connected to. To benefit from the extra connectivity of the end node to the network, at each direction one hop is potentially saved (e.g. if going east then the destination switch is one hop closer).

After finding the output port, the algorithm checks if the port is available. If the port is available ($free[p]$), then it is selected. However, if the port (either internal or switch-to-switch port) is busy, the algorithm checks if an extra hop is possible (non-minimal routing). Six cases are considered, none of which introduces a Y $\rightarrow$ X turn. For example, if the message is coming from the *west* port and the destination is in the north-east quadrant, then the message can still be routed either *east* or *north*. An end node is considered to be in the north-east quadrant if it can be reached through the current Y column or through the next Y column. The same reasoning is used for messages coming from the east port.

For messages coming through the south port, they can still move *north*, but cannot go *east* or *west* as this would lead to a Y $\rightarrow$ X turn. The same occurs for messages coming through the north port. Notice also that U turns are not allowed by the algorithm.

As a final step, the *priority_select* function returns only one output port from the set of non-minimal ports computed. The function gives priority to X ports (east, west) over Y ports (north, south). If no valid output port is selected, the routing algorithm is executed again at the next cycle.

In Figure 3.10 we can see an example of all the possible output ports the message can take when moving from end node S to end node D. As can be seen, at the columns near the destination different alternative choices can be taken.

The topology also allows the use of adaptive routing. In this case, virtual channels are needed to decouple adaptive paths from escape paths. One alternative is to use one virtual channel to route messages adaptively and one virtual channel to route messages through the escape path.

Figure 3.11 shows the adaptive routing algorithm proposed for the topology. The algorithm returns an output port and a virtual channel to be used at the next switch. First, the set of output ports providing minimal paths to the destination is retrieved (function *min_adap_and_av_ports*), which returns the available ports closer to the destination when they are not busy. Notice that only output ports currently available are considered. If at least one port is available, then the algorithm takes the port and outputs the adaptive virtual channel that must be used for the message (*adaptive_vc*). However, if all the adaptive minimal paths are busy, then the algorithm takes the deterministic routing path. In this case, the previous function (*deterministic_routing*) is

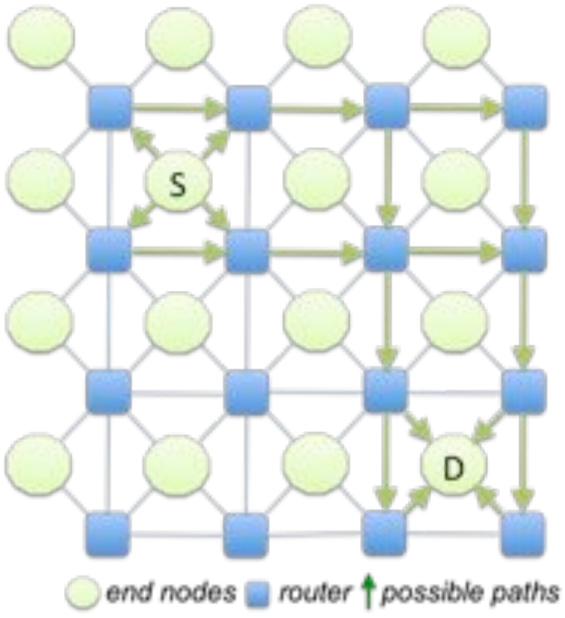


Figure 3.10: All possible paths in the NR-Mesh.

called. The virtual channel to be used in this case is the deterministic one (*deterministic_vc*).

Messages may take adaptive virtual channels and deterministic virtual channels along their path in any order. In wormhole switching networks, however, this may lead to deadlocks. This happens as indirect channel dependencies are introduced that potentially lead to cycles in the CDG. However, this is not possible with virtual cut-through switching. If wormhole switching is used, then the algorithm must prevent messages moving from deterministic virtual channels back to adaptive virtual channels.

In the evaluation section, we evaluate the NR-Mesh topology with both routing algorithms, deterministic and adaptive. The adaptive routing algorithm will be later modified in order to be used in a system where input ports and links are switched on and off dynamically.

3.1.4 Topology Properties

In this section, we explore the main properties of the NR-Mesh topology. The first one is its flexibility when injecting and ejecting messages. An end node may decide which injection port should be used. At the destination, the routing algorithm is able to deliver the message through one of four different

```
function adaptive_routing(iport, cur, dest) : port, virtual_channel
var     ap: set of ports
begin
   ap = min_adap_and_av_ports(cur, dest)
   if (empty(ap))
         return deterministic_routing(iport, cur, dest), deterministic_vc
   else
         return random(ap), adaptive_vc
end function
```

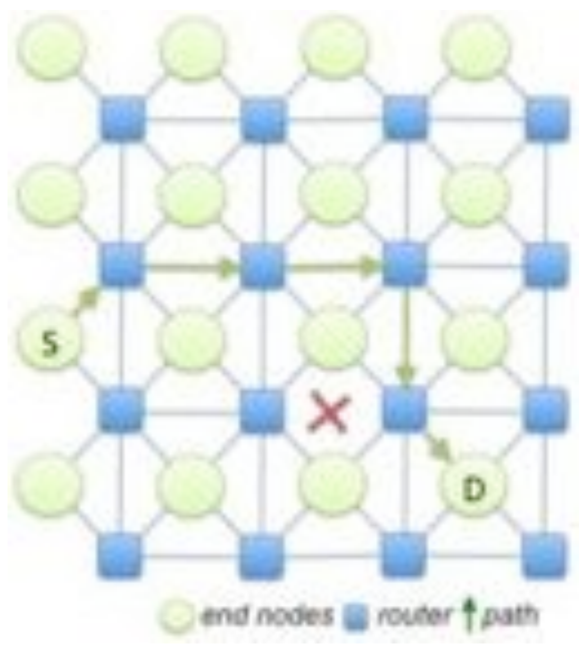
Figure 3.11: Adaptive routing algorithm for the NR-Mesh topology.

switches. This flexibility is not available in the original 2D-Mesh topology. This property will be exploited by the power management technique that switches off resources (ports and links) during idle periods. Also, congestion within the network can be alleviated.

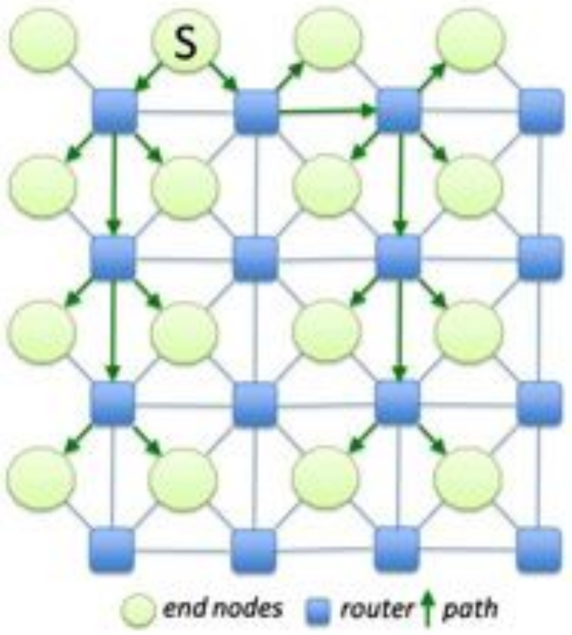
A second property of the NR-Mesh topology is its lower diameter. The diameter of an $N \times N$ NR-Mesh network is $2N - 2$, since at the injection and ejection points, one hop per dimension is saved. This will also lead to lower message forwarding latencies and lower power consumption within the network. This property will, however, loose ground with larger configurations.

The NR-Mesh topology provides a high fault tolerance degree. Network links and even switches could fail and still valid paths from sources to destinations can be available. In contrast, the 2D-Mesh case with DOR routing does not tolerate a single link or switch failure. Figure 3.12(a) shows an example where a link failure is supported by the topology. The S end node is still able to send messages to end node D through the highlighted path.

Collective communication can be efficiently implemented in the NR-Mesh topology. A single unicast message may reach four destinations if they coincide with the end nodes connected to the destination switch. Thus, localized multicast can be efficiently implemented. When communicating to larger sets of end nodes through multicast, or even when dealing with broadcast, the topology is advantegeous. A broadcast operation that sends a single message



(a) Fault-tolerance.



(b) Collective (broadcast) communication.

Figure 3.12: Examples of additional properties of the NR-Mesh topology.

to all the end nodes may be reduced in that a message is only replicated at some switches and then forwarded only through a few rows and columns. At every visited switch, the message is forwarded to all the neighboring internal ports. Traffic overhead is, then, significantly reduced. Figure 3.12(b) shows an example. Only five links between switches, out of 24, are used to reach all the end nodes. In a 2D-Mesh most of all the external links would be required.

Although such opportunities exist for the NR-Mesh topology, we do not evaluate them. These features will further exacerbate the benefits of NR-Mesh topology.

3.2 Power Management Algorithm

In this section, we propose a power management algorithm that works in cooperation with the adaptive routing algorithm presented before. The basic goal of the algorithm is to maximize the time that switch input ports and input links are powered down. Due to the large power consumption in the input buffers this technique potentially achieves large savings in static power

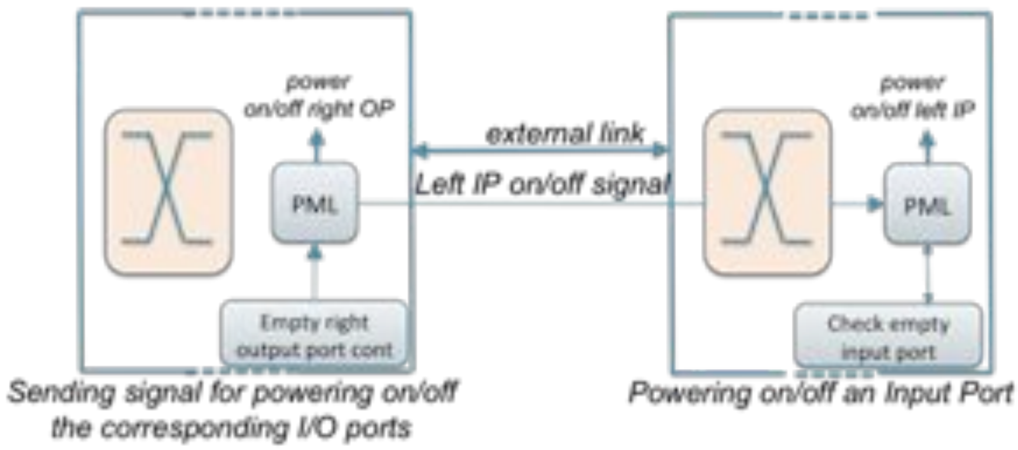

power on/off right OP
external link
Left IP on/off signal
PML
Empty right output port cont
power on/off left IP
PML
Check empty input port
Sending signal for powering on/off the corresponding I/O ports
Powering on/off an Input Port

Figure 3.13: Powering on/off the next IP (input port) from the previous switch.

consumption. The routing algorithm is modified in order to avoid, whenever possible, the need to use input ports that are switched off, and to rather route messages through other alternative minimal or non-minimal paths. Obviously, depending on the message's destination, this will not always be possible and the port will need to be powered on.

The algorithm relies on three key steps. The first one is deciding when an input port can be powered down. The second is making the routing algorithm aware of which ports are switched off. The third one is related to deciding when an input port needs to be activated again. We describe each step next.

Notice that an input port will be highly coupled with the associated output port at the upstream switch. They will work in tandem, indeed, whenever an input port switches off it will notify the associated output port.

3.2.1 Algorithm for Switching Off an Input Port

Every switch in the network includes a new Power Management control Logic (PML). This logic is in charge of switching *off* and *on* every input port. These actions are made locally and independently from the other switches in the network. Only input traffic is taken into account from the previous output port. Figure 3.13 shows the PML located at the switch to power off the next input port from the previous output port.

For switching off a port, the PML logic measures the port utilization,

```
procedure check_port(current_output_port)
begin
  if (cycle_empty_cnt[current_output_port] > threshold)
     send $IP\_off\_signal$ to the corresponding input_port into the next switch
     cycle_empty_cnt[current_output_port] = 0
  else
     cycle_empty_cnt[current_output_port]++
  end
end procedure
```

Figure 3.14: Algorithm for *switching off* an input port.

computed as the number of cycles the port is unused. When a given threshold is reached, the port is switched off. Figure 3.14 shows the control algorithm.

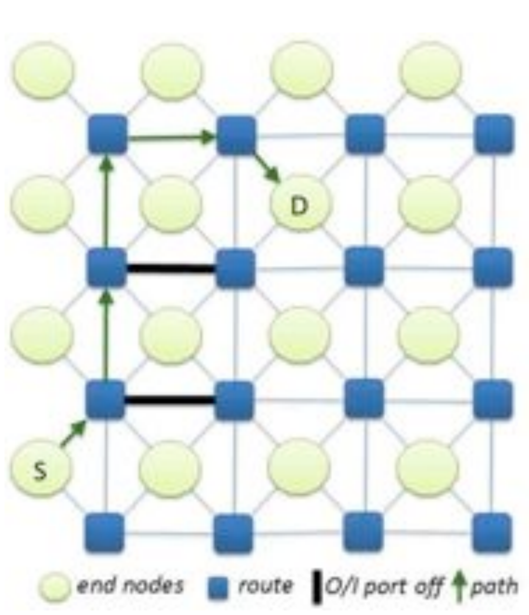
Therefore, an internal control signal (IP_off_signal) is sent to the associated input port to notify that the logic must be switched off (through power gating). When all the input ports in a switch are powered down and there is no left flit in the switch, the logic switches off the entire switch, including the clock.

The process to power down a block may consume some power, however this power consumption is compensated by the power consumption saved when the port is powered off. To save power, the port needs to be powered off for at least ten cycles. However, there is no power penalty for waking up the port earlier. We have followed the recommendations in [16] for these numbers.

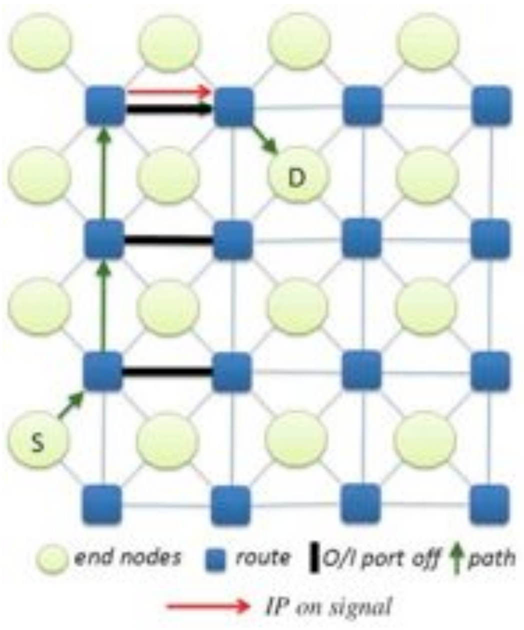
3.2.2 Algorithm for Routing Messages

The routing algorithm described in Section 3.1.3 is modified in order to support powered down components. Two changes are required, the first one deals with the need of the algorithm to know which ports are in off state and which are not. This can be easily achieved by extending the concept of a busy port. A port is busy if it is transmitting a message, there is no credit available, or the associated input port at the other end of the channel is powered down.

The second change deals with the need for resuming an input port. This



(a) Avoiding powered off ports.



(b) Waking up a powered off port.

Figure 3.15: Routing algorithm assuming the status of the network.

may happen when a message ends up requesting the output port. Notice that the *adaptive_routing* algorithm may not find an available adaptive output port, and, in that case, a deterministic output port is required. When the port is disabled, the logic at the disabled next input port is notified by the PML in the previous switch (IP_on_signal signal in Figure 3.13), and after the wake up delay (explained in the next subsection), the message is forwarded. Figure 3.15(a) shows, on the one hand, how the algorithm avoids powered down or busy ports and, on the other hand, Figure 3.15(b) shows how an input port is woken up as the algorithm can not take another path.

3.2.3 Algorithm for Switching On a Port

In this dissertation, we assume the conservative approach of three cycles to wake up a switch input port (obtained from [16]). To efficiently achieve this, the PML logic wakes up an input port when a control signal from a connected device (neighbor switch or attached end node) is received. This leads to three-cycle penalty when switching on an input port attached to an end node. When a signal is sent from a switch, there is one-cycle penalty (the VA_SA and ST stages are overlapped in this case). See Figure 3.16 for details of the switch

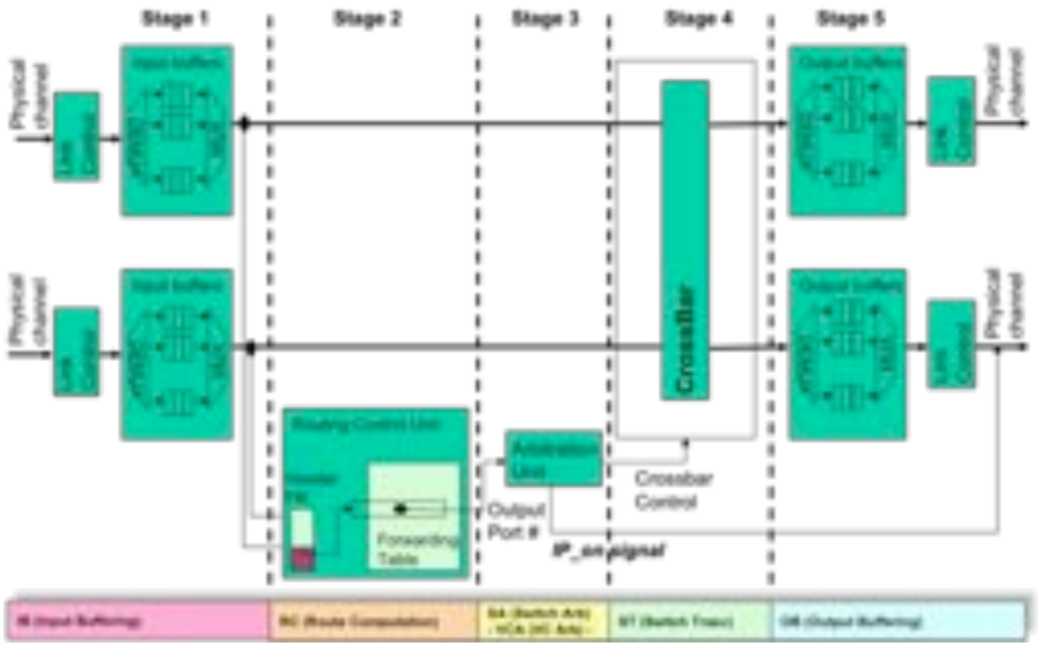


Figure 3.16: Switch stages.

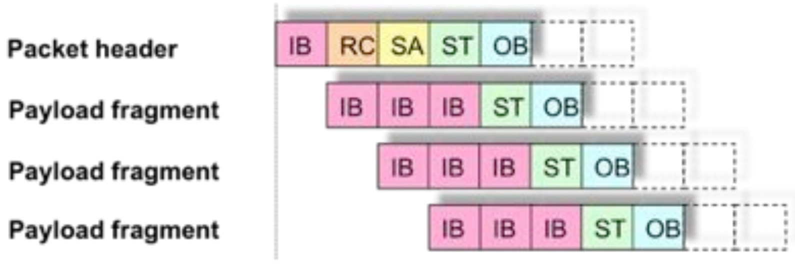


Figure 3.17: Pipeline of the switch stages.

stages.

Figure 3.17 shows the pipeline of the stages when a packet is traversing a switch. As we can see, the *Packet header* traverse all the stages in the router. First, it is stored into the input buffer at the switch. Secondly, it computes the routing stage to see whether the routing is possible in the current cycle or we have to wait until the next cycle. In the next stage, the switch arbitrates the *header* to provide a virtual allocation for the packet. Next, the header traverse the switch and is ready to cross the link. Rest flits of the packet (*Payload fragment*) need to be buffered in the switch and compute the swich traverse. For each packet, the *route computation* and *switch arbitration / virtual allocation* is taken only once in the *Packet header*.

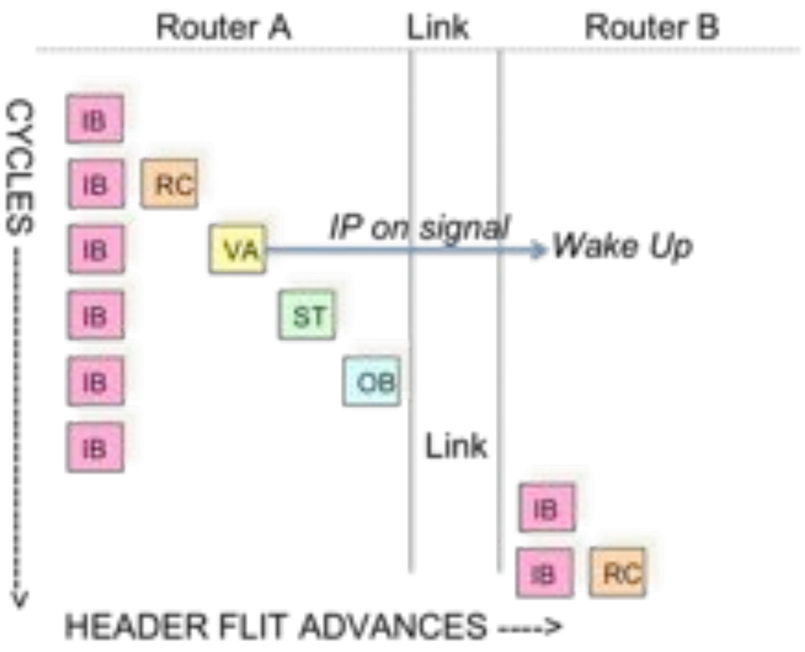


Figure 3.18: First stages of the pipeline when an input port is being powered on.

Figure 3.18 shows, on the other hand, the pipeline stages when an input port is being powered on by the previous switch. Once a header flit reaches the arbiter (VA stage), the IPon signal is forwarded. At the same time, the header flit crosses the ST and OB stages and the link. By the time the header flit arrives to the input port, the port is awake. However, *OB* represents in this case one-cycle penalty as we do not use this stage in our simulator.

3.3 Performance Evaluation

In this section, we evaluate the new topology in different scenarios together with the different routing algorithms, deterministic and adaptive (or power saving algorithm). We also analyze, for comparison purposes, the 2D-Mesh network. Finally in this chapter, we evaluate the performance of the NR-Mesh topology compared with other similar topologies as the Concentrated Mesh topology or the NR-Mesh using only two injection ports.

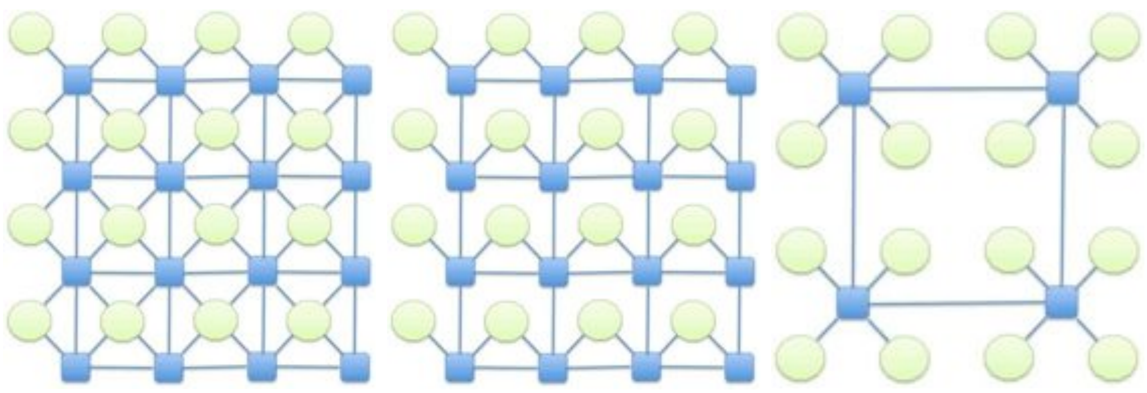
3.3.1 Evaluated parameters

The comparison has been performed in terms of average message latency (cycles), network throughput (flits/cycle/tile), power consumption (W), network energy consumption (power consumption during the entire execution of each application) and execution time (cycles). For the routing algorithms, we have used the deterministic DOR algorithm and the adaptive routing algorithm described in Figures 3.8 and 3.11, respectively. In both topologies, we use the same number of virtual channels; for the deterministic algorithm we use two VCs, and for the adaptive algorithm we use one adaptive VC and one escape VC (implementing the deterministic routing algorithm). To avoid protocol level deadlocks, we use virtual networks, where basically separate queues are used for each virtual network. Notice that we use VCs inside each virtual network.

Components in the network are switched off and on when using the adaptive algorithm in both topologies. Notice that when using the 2D-Mesh topology the algorithm will have less options for skipping powered down links as end nodes will be connected only to one switch. The number of cycles for switching the ports on and off has been obtained from [16]. We assume that logic requires more than nine cycles to switch off a port in order to save power. The delay to power on a port is three cycles. The threshold to switch off a port is set to 100 and 200 cycles for 16-node and 32-node CMP systems, respectively. These values exhibited a good trade-off between performance and energy savings.

Power estimates have been obtained from the Orion-2 power model [17], assuming 45nm technology, with a network frequency of one GHz and a 1.1V supply voltage.

Besides comparing the 2D-Mesh and NR-Mesh topology, we compare the NR-Mesh with other similar topologies, namely, the one proposed in [43] (referred to as NR/2-Mesh) and the concentrated mesh (C-Mesh). Figure 3.19 shows the cited topologies. We assume both a 16-tile system and a 32-tile system.



(a) NR-Mesh. (b) NR/2-Mesh. (c) C-Mesh.

Figure 3.19: The NR-Mesh and other topologies.

3.3.2 Simulation Model

We use different types of synthetic traffic patterns, which allow us to explore a wider scenario:

- Uniform: Destination of messages are spread uniformly among all the end nodes.
- Bit-complement: Unary operation that performs logical negation on each bit, forming the ones' complement of the given binary value. Digits which were 0 become 1, and vice versa.
- Bit-reversal: Bit reversal is the permutation where the data at an index n, written in binary with digits $b_4b_3b_2b_1b_0$ (e.g. 5 digits for N=32 inputs), is transferred to the index with reversed digits $b_0b_1b_2b_3b_4$.
- Hot-spot: One node in the network is receiving much more traffic than the others. In our case, 6% and 12% additional traffic is injected to the *hotspot* node for a 16- and a 32-node system, respectively.

The acccepted traffic and power consumption metrics for synthetic traffic are calculated as follows: the accepted traffic is the number of flits by cycle and tile that the network is able to deliver to the end nodes for a given injection rate. The power consumption is the average power by cycle in the network for a given injection rate, using the Orion-2 model [17].

In addition to synthetic traffic patterns, we use SIMICS [22] and GEMS [23] to model a complete system. GEMS is a General Execution-driven Multiprocessor Simulator. It is a subset of modules for SIMICS that enables detailed simulation of multiprocessor systems, including Chip-Multiprocessors (CMPs). Simics is a full system simulation platform, capable of simulating high-end systems with sufficient fidelity and speed to boot and run operating systems and commercial workloads. One of the most important modules of GEMS is Ruby. The GEMS Ruby module provides a detailed memory system model, where each end node includes an in-order processor core, an L1 data and instruction cache, an L2 cache bank, and a directory/memory controller. L1 cache, L2 cache, and the directory are connected to the switch, and to four switches for the NR-Mesh topology. One internal port is used to connect each end node to the switch in the 2D-Mesh and C-Mesh topologies (four in the NR-Mesh topology and two in the NR/2-Mesh one).

We have replaced the network simulator inside Ruby by a cycle-accurate event-driven network simulator. We have developed an in-house network simulator (gNoCsim). gNoCsim is an event-driven cycle-accurate network simulator that models the pipelined structure of the gNoC switch (designed in Verilog) and the network interface. This simulator is used in both scenarios, real applications and synthetic traffic patterns.

Table 3.2 shows the main end node parameters and cache coherency protocol parameters. A directory-based protocol is used whereby five virtual networks are required (to avoid protocol-level deadlock). Table 3.3 shows the network parameters. The flit size has been set to eight bytes. For internal and external links (links connecting to end nodes and links between switches, respectively) one cycle delay is assumed for both topologies, except for the internal links in the NR-Mesh case, where the link length has been modeled with two cycles delay, because on average the switches are more distant from the end node. Buffer size at switches is set to 10 flits to support virtual cut-through switching (VCT) and one cycle is assumed for each switch stage (four stages in total).

Several Splash-2 [40] applications and a commercial workload (when comparing with other topologies) [1] have been run. Only the parallel section for the Splash-2 applications have been measured, while 5000 transactions for

Parameter	Value	Parameter	Value
L1 size	128 KB private	L1 hit latency	3 cycles
L2 size	8MB shared	L2 hit latency	6 cycles
Coherency protocol	Directory-based	Virtual Networks	5
Processors	16 & 32		

Table 3.2: End node and cache coherency protocol parameters.

General network parameters		Router parameters	
Flit size	8 bytes	Buffer size	10 flits
Externals links	1c delay (2c C-Mesh)	VCs per VN	2
Internal links	1c delay (2D-Mesh)	Routing	1c delay
Internal links	2c delay (NR-Mesh)	Arbiter	1c delay
Internal links	2c delay (C-Mesh)	Crossbar	1c delay
Internal links	2c delay (NR/2-Mesh)	Transmit	1c delay

Table 3.3: Network parameters.

the commercial workload have been considered. For this analysis, we combine applications with low traffic loads (FFT, BARNES and LU) and with higher traffic loads (RAYTRACE, RADIX, and APACHE). We show also the average results for all the topologies.

The FFT kernel is a complex 1-D version of the radix-$\sqrt{2}$ six step FFT algorithm, which is optimized to minimize interprocessor communication. The data set consists of n complex data points to be transformed, and another n complex data points referred to as the roots of unity. RAYTRACE renders a three-dimensional scene using ray tracing. A hierarchical uniform grid (similar to an octree) is used to represent the scene. A ray is traced through each pixel in the image plane that reflects in unpredictable ways off the objects it strikes. The BARNES application simulates the interaction of a system of bodies (galaxies or particles, for example) in three dimensions over a number of time-steps, using the Barnes-Hut hierarchical N-body method. The LU kernel factors a dense matrix into the product of a lower triangular and an upper triangular matrix. The integer RADIX sort kernel is based on the method described in [3]. The APACHE commercial workload is a popular open-source web server used in many internet/intranet environments.

Component	Power consumption
Injection	2.5 μW
3-port switch	34.63 mW
4-port switch	49.57 mW
5-port switch	63.11 mW
6-port switch	76.42 mW
7-port switch	88.37 mW
8-port switch	102.13mW

Table 3.4: Power consumption of the different components for the NR-Mesh topology. Target frequency set to one GHz.

3.3.3 Implementation Results

We have designed and synthesized the network interface, and a 5-stage pipelined switch with the 45nm Nangate open-source library [44]. We have modeled switches with different number of ports, all of them with the same target frequency (1GHz). We have obtained the total power consumption using the Power Compiler tool from Synopsys [49] after place and route using Encounter [46] Place&Route tool from Cadence [48]. Results are shown in Table 3.4. We have measured dynamic power for different traffic loads (from no load to one flit/cycle/input port). Results did not vary significantly as the static power is the main contributor. Furthermore, we have included the average power consumption in the results shown in the table.

As can be deduced from Table 3.4, additional power consumption at the network interface (due to the injection algorithm) is negligible when compared with the switch power consumption. Also, we can see how power consumption almost doubles when moving from a 5-port (as required in a 2D-Mesh) switch to a 8-port switch (as required in the proposed topology). This is mainly due to the addition of the buffer resources. However, it should be noted that the listed power assumes that the entire switch is in on-state. The NR-Mesh topology, combined with the adaptive routing algorithm will switch off ports and links, thus saving power.

Table 3.5 shows the latency and area overheads of the injection algorithm

Component	critical path (ns)	area (μm^2)
Injection	0.21	57.40 μm^2
3-port switch	1.00	36,417.96 μm^2
4-port switch	1.00	49,954.71 μm^2
5-port switch	1.00	64,354.10 μm^2
6-port switch	1.00	71,827.22 μm^2
7-port switch	1.00	95,188.55 μm^2
8-port switch	1.00	111,724.72 μm^2

Table 3.5: Area and delay overheads for the different components. Target frequency set to one GHz.

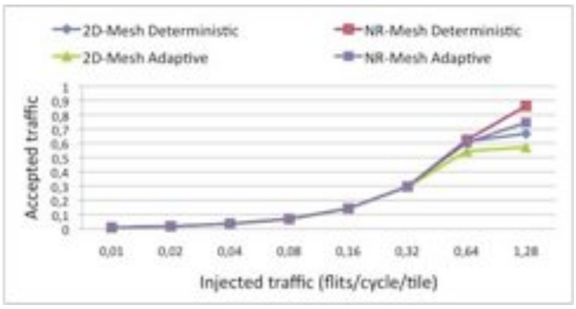
at the network interface as well as of the switch designs. The delay of the injection algorithm is much smaller than the critical path of the switch and therefore does not set the bottleneck in the transmission path of the message. As expected, the area needs of the injection algorithm are also negligible when compared to the switches. Less than 1% is really needed. This device will provide much gains when combined with the power management mechanism within the network.

In the next section, we evaluate the performance achieved for the different applications when using both topologies (2D-Mesh and NR-Mesh) and deterministic versus adaptive routing algorithms. When using the adaptive algorithm, the power-saving algorithm is enabled, which leads to little performance penalty. In the two next subsections, we analyze the accepted traffic and the power consumption using synthetic traffic. Later, we analyze the execution time and power consumption using GEMS.

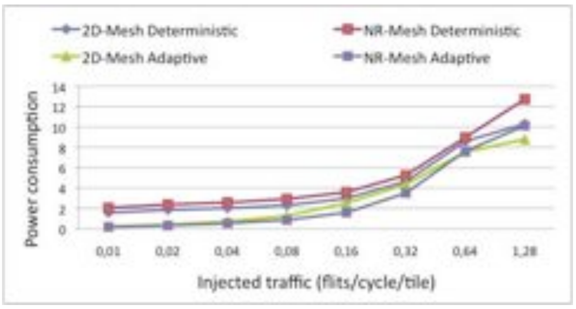
3.3.4 Analysis with Synthetic Traffic

Figure 3.20 shows the performance and power consumption for a 16-tile system (4×4 topologies) under synthetic traffic patterns (uniform, bit-reversal, and bit-complement). Figures a, c, and e show accepted traffic and figures b, d, and f show the power consumption.

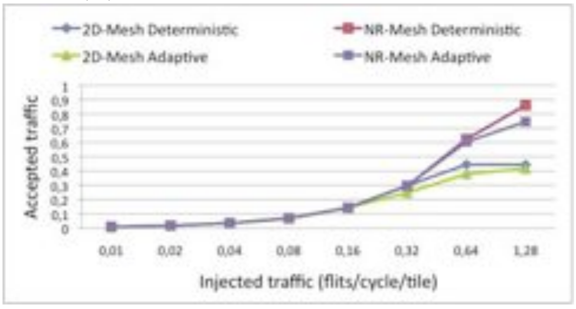
One thing to notice is the higher throughput achieved by the NR-Mesh



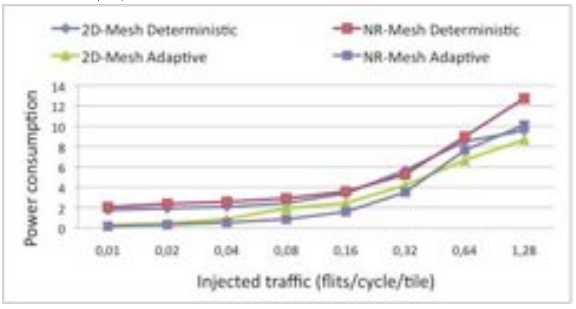
(a) accepted traffic, uniform



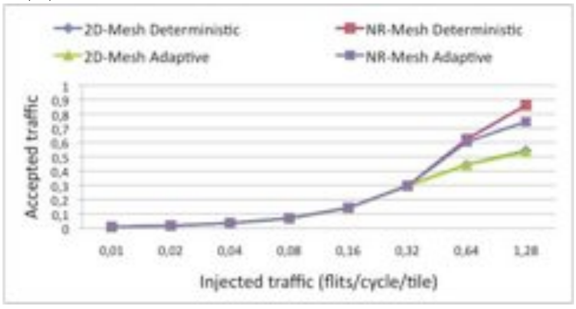
(b) power per flit, uniform



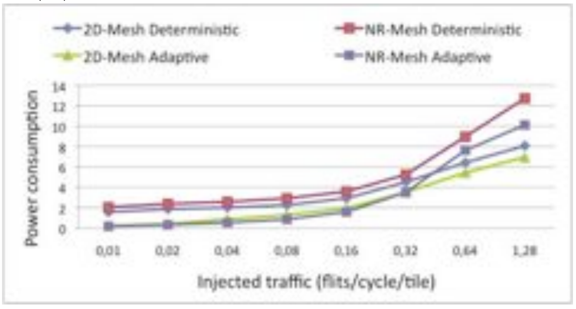
(c) accepted traffic, bit-complement



(d) power per flit, bit-complement



(e) accepted traffic, bit-reversal

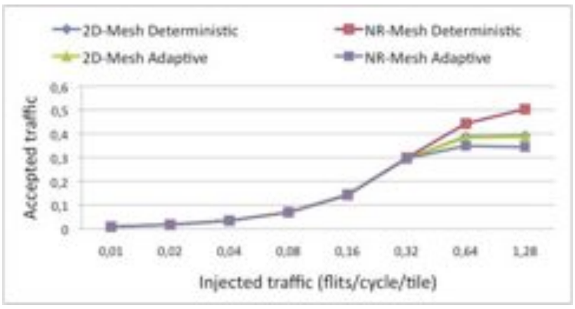


(f) power per flit, bit-reversal

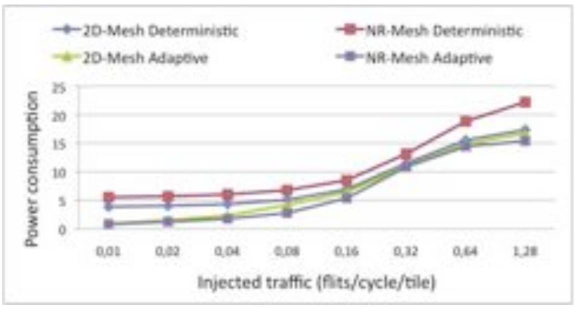
Figure 3.20: Accepted traffic and power consumption for 16-node systems. Synthetic traffic patterns.

topology for the three synthetic traffic patterns. This is because of the higher bisection bandwidth (as some nodes are connected to both sides of the bisection) and the lower diameter of the topology. In bit-complement traffic, the throughput of the 2D-Mesh is doubled. It can also be noticed that the adaptive routing algorithm does not sustain the maximum throughput of the deterministic algorithm in high traffic rates. This is due to the extra latency when switching on components and the use of non-minimal paths. However, the power reduction plays a good trade off between low and high traffic rates.

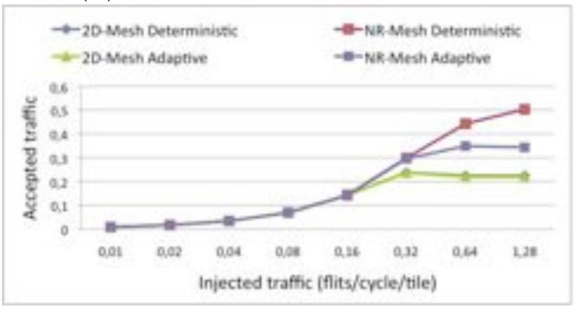
At low loads, we can see how the adaptive algorithm is able to signifi-



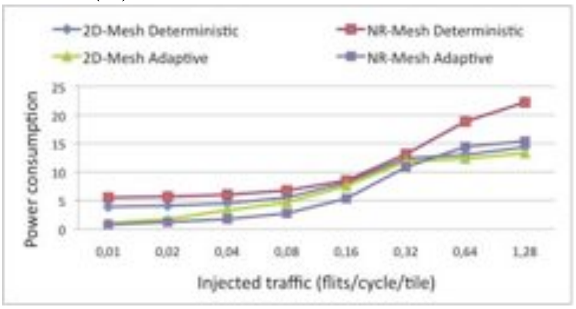
(a) accepted traffic, uniform



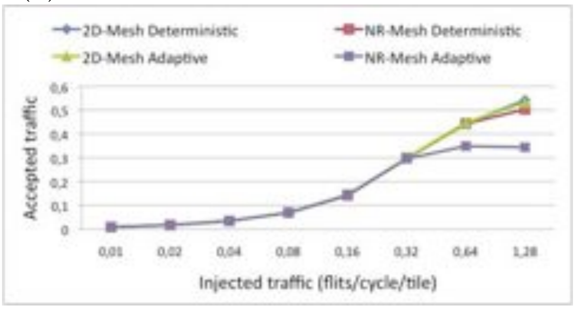
(b) power per flit, uniform



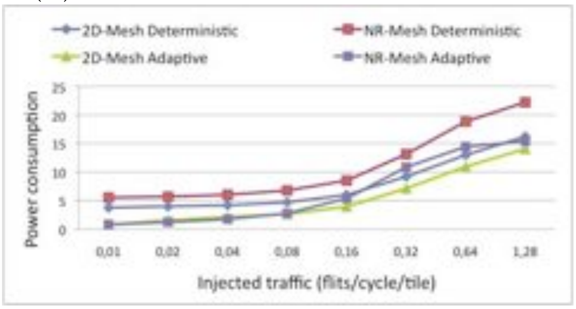
(c) accepted traffic, bit-complement



(d) power per flit, bit-complement



(e) accepted traffic, bit-reversal

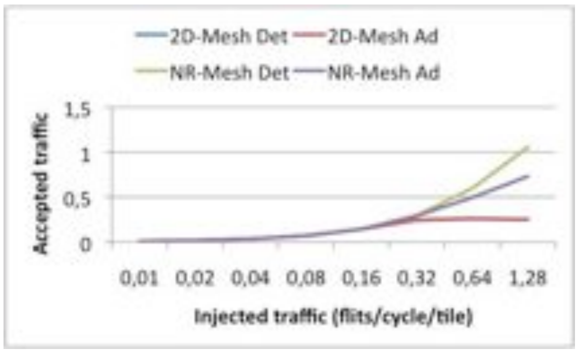


(f) power per flit, bit-reversal

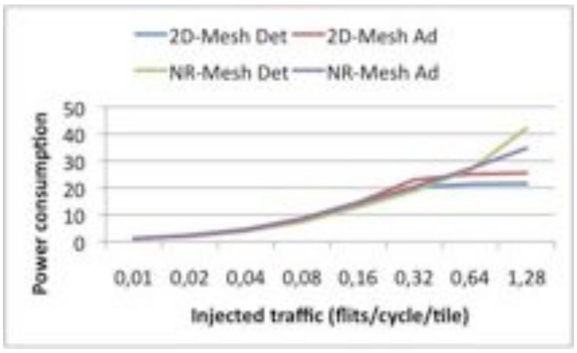
Figure 3.21: Accepted traffic and power consumption for 32-node systems. Synthetic traffic patterns.

cantly reduce the power per bit metric (reduction factor larger than 2x) in both topologies. For higher traffic rates (near saturation), the adaptive algorithm still achieves power reductions, although to a lesser degree. The best combination with respect to performance and power is, therefore, the NR-Mesh with adaptive routing.

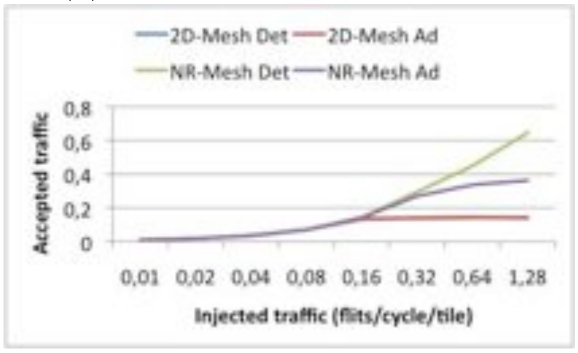
Figures 3.21 show the same results, but now for 32-tile systems (4×8 topologies). The tendency in this case is different, specially, for the NR-Mesh. The deterministic case takes more power to achieve good throughtput (accepted traffic). The adaptive case achieves lower power consumption with



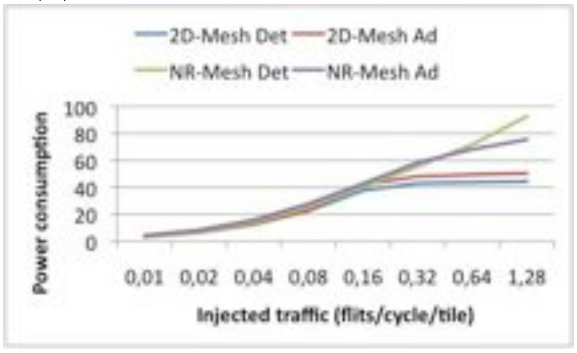
(a) 16 nodes. Accepted traffic.



(b) 16 nodes. Power consumption.



(c) 32 nodes. Accepted traffic.



(d) 32 nodes. Power consumption.

Figure 3.22: Performance and power consumption for 16- and 32-node systems. Hot-spot scenario.

lower traffic loads. Power consumption increases with higher loads as in the previous case. In some traffic patterns, when the network saturates, the accepted traffic in the NR-Mesh topology decreases compared to the 16-node case. Thus, there is an expected trade off between power and performance.

Next, we use synthetic traffic to examine the impact of a hot-spot. In this case, the NR-Mesh topology will be able to deliver more traffic. Consequently, it will take less time to deliver the traffic and, with it, less power will be consumed. Therefore, the higher power consumption, in this case, will be compensated with the increase in performance.

Hot-spot Scenario

Figure 3.22 shows a hot-spot scenario where a significant amount of traffic to one node in the center of the network is injected.

The measurements have been done for 16- and 32-node systems. Figure

3.22 shows accepted traffic and power consumption for the 2D-Mesh topology and the NR-Mesh with both deterministic and adaptive routing.

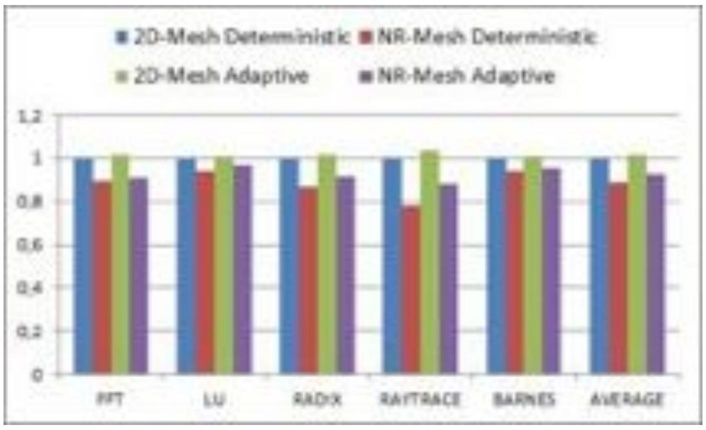
In this scenario, the NR-Mesh is clearly superior to the 2D-Mesh topology. And not only in terms of accepted traffic, but also in terms of power consumption.

We could expect in NR-Mesh an increase in power consumption, because of the additional internal links. However, this effect is compensated by the fact that traffic is delivered faster, that is, a message spends less time in the network, thus, saving power.

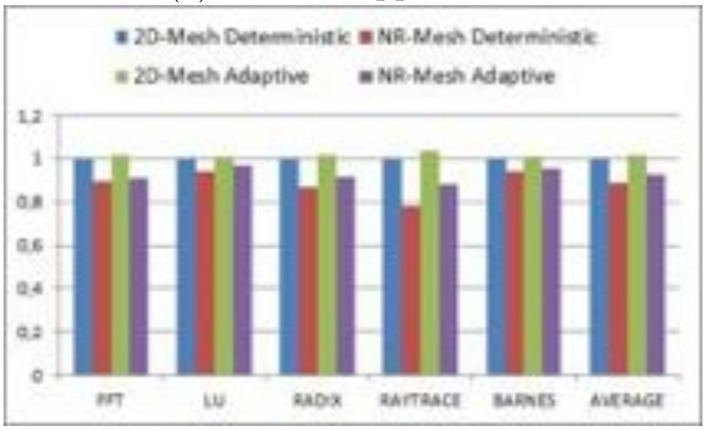
Though deterministic and adaptive routing perform similar in the 2D-Mesh topology, this is not the case for the NR-Mesh one. Deterministic routing performs better in a hot-spot scenario than adaptive routing, however, it requires more resources and then, wastes more power. This behavior can be easily explained as follows: the deterministic routing in both 2D-Mesh and NR-Mesh network on-chip topologies do not power off ports and/or switches obtaining less latency to reach the destination. For this reason, in a saturated condition, the messages, when using the deterministic routing algorithm, do never encounter off ports thus not having extra latencies. This is the opposite when using the adaptive routing algorithm, where messages sometimes face an off port and need to wait the port being woken up, although saving lot of power. Therefore, this is a trade off between performance and power consumption. Even though, the adaptive routing in the NR-Mesh greatly outperforms both deterministic and adaptive routing to the 2D-Mesh topology by 50% and 20% in high traffic scenarios for 16- and 32-node systems, respectively.

3.3.5 Analysis with Applications

Let's now turn our attention into performance when running real applications. Figure 3.23(a) shows execution time for the 16-node CMP system, for each topology/routing algorithm. Results are normalized for the 2D-Mesh/deterministic case for each application. As we can see, execution time when using the adaptive routing algorithm increases slightly in both topologies. In this configuration, components are switched off and on depending on traffic conditions. Some ports are switched off and few cycles later turned on to receive an incoming message (as the other links are busy). As the



(a) 16-Node applications



(b) 32-node applications

Figure 3.23: Normalized execution time for different applications under different topologies and routing algorithms.

link sometimes needs to wait several cycles to be woken up (as we described before), some latency penalty is introduced. Also, avoiding turned off links forces the traffic over non-minimal paths increasing the associated message delay. Though, there is a trade-off between execution time (performance) and power savings, the penalty is less than 2% more execution time on average in the 2D-Mesh case. Although in the adaptive NR-Mesh the penalty is slightly higher when comparing with the deterministic case in the same topology, the execution time is always significantly lower than for the 2D-Mesh case.

Looking at the NR-Mesh (using either deterministic or adaptive routing), we can see large reductions of execution time, up to 12% in Raytrace, when compared with the 2D-Mesh. Regardless of the algorithm (deterministic or adaptive), the NR-Mesh topology achieves better performance than the 2D-

Mesh topology.

Figure 3.23(b) shows the execution time for a 32-node CMP system. Results are similar to the previous case. On average, the NR-Mesh outperforms the 2D-Mesh by 14% for the deterministic case, and by 8% for the adaptive one. In addition, the adaptive version for 2D-Mesh topology increases execution time, by 2%. As we said before, even if the execution time in the adaptive version of the NR-Mesh is increased, it is still better than the 2D-Mesh deterministic case.

Now we compare the total NoC power consumption for each application between both routing algorithms and topologies by combining: the average energy consumption[1] and the total execution time of applications. Figure 3.24 shows the results. Results are normalized to the 2D-Mesh deterministic case for each application.

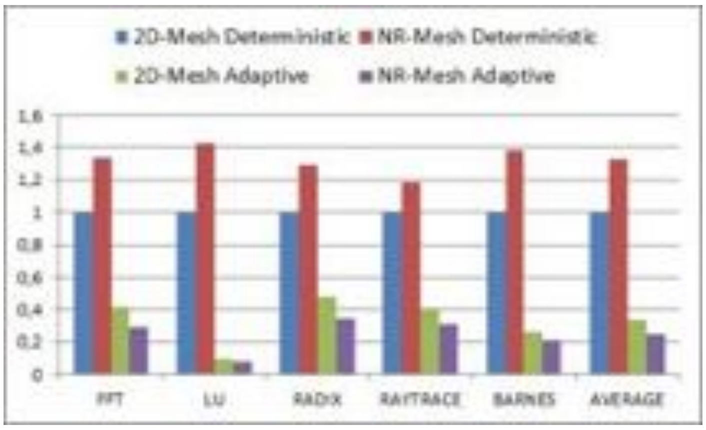
Large savings are obtained when using both the NR-Mesh topology and the adaptive algorithm: 75% on average for the 16-Node CMP system. For the 32-node CMP system results achieve a 69% improvement. Although the 2D-Mesh benefits greatly from the use of the adaptive algorithm (switching off unused components), the NR-Mesh further increases improvements. For instance, the Radix application with 16 nodes gets an additional benefit of 14% in energy reduction when compared with the 2D-Mesh case with adaptive routing. Therefore, the flexibility provided by NR-Mesh significantly improves the effectiveness of the power management algorithm.

3.3.6 Additional Performance Comparisons and Analysis

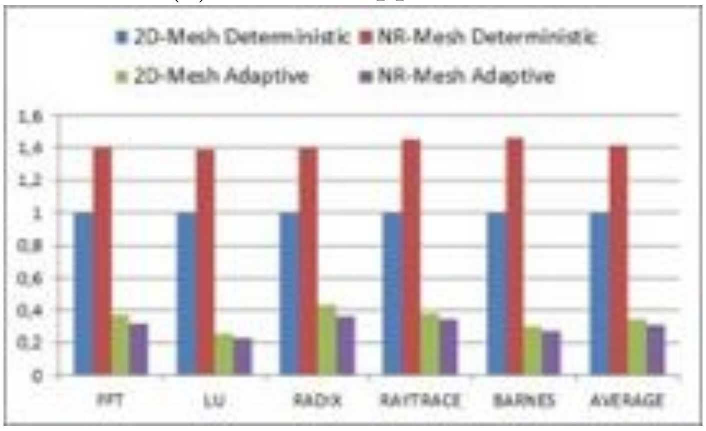
Now we focus on particular aspects of the NR-Mesh topology. First we analyse the variability in performance and power consumption when changing the number of injection ports. Then, we focus our attention into similar topologies, comparing the total execution time of several applications.

We know that the potential of the NR-Mesh relies on the number of injection links and a proper power management mechanism using adaptive routing. To verify this assumption, we simulate all the topologies shown in Figure 3.25.

[1]The energy consumption is calculated by adding the power consumed in the network every cycle during the whole execution of the application.



(a) 16-Node applications



(b) 32-node applications

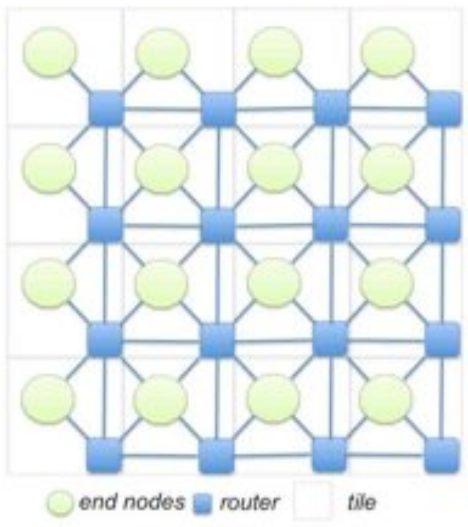
Figure 3.24: Normalized energy consumption for different applications under different topologies and routing algorithms.

We vary the topologies by removing an injection link from all the end nodes until the NR-Mesh becomes a 2D-Mesh.

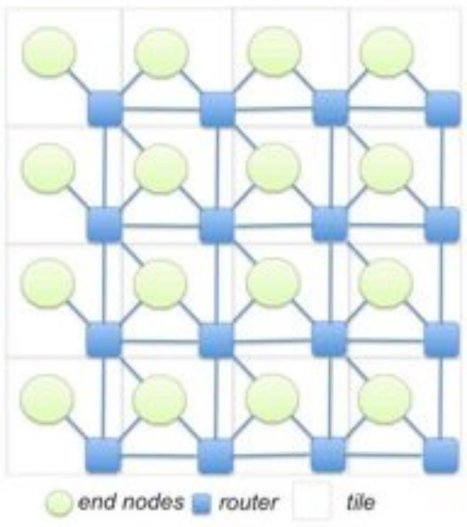
Performance Analysis with Hot-Spot Traffic

Now, we perform the same experiment as in the previous one but now varying the number of injection links at the end nodes. We want to see the behavior when varying the number of input ports. The final goal is to show the flexibility provided when having 4 different injection ports.

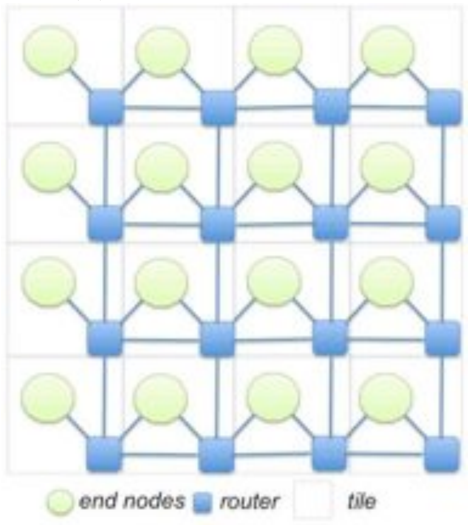
Figure 3.26 shows performance achieved for a 16-node system and Figure 3.27 shows results for a 32-node system. When looking at the accepted traffic, we can see that with fewer injection links, less traffic is accepted in the



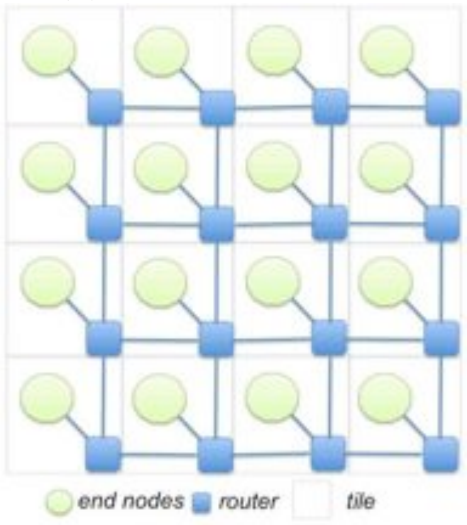
(a) four link injection.



(b) three link injection.



(c) two link injection.

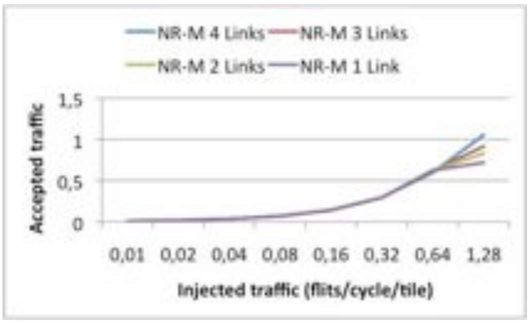


(d) one link injection.

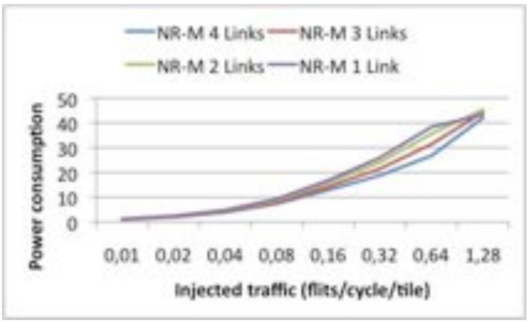
Figure 3.25: Link injection variation in the NR-Mesh.

saturated scenarios. A special case is adaptive routing for 32 nodes, where we achieve the best results with two internal links, although the percentage difference is not significant, however, by consuming more power. Fortunately, having more internal links, we are able to decrease power consumption due to the high number of alternatives to route a packet, achieving a better trade off between performance/power.

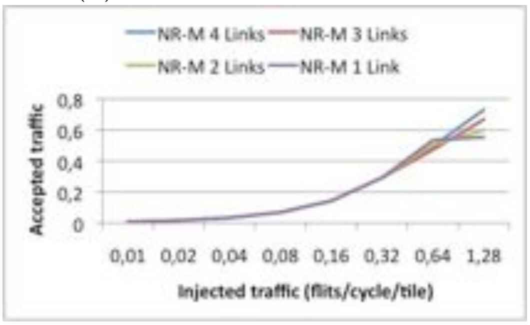
The power consumption is also worse in both 16 and 32-node system as we decrease the number of injection links, most notable in the adaptive routing case. Also, note that by having four or three injection links, the behavior is



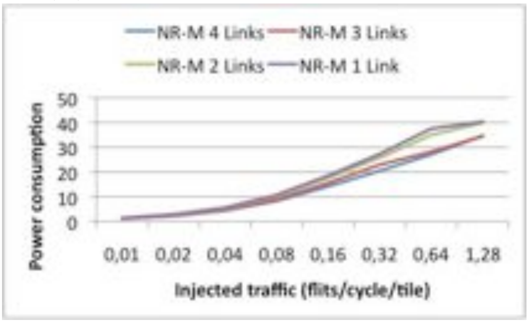
(a) Accepted traffic. Det.



(b) Power consumption. Det.



(c) Accepted traffic. Adap.



(d) Power consumption. Adap.

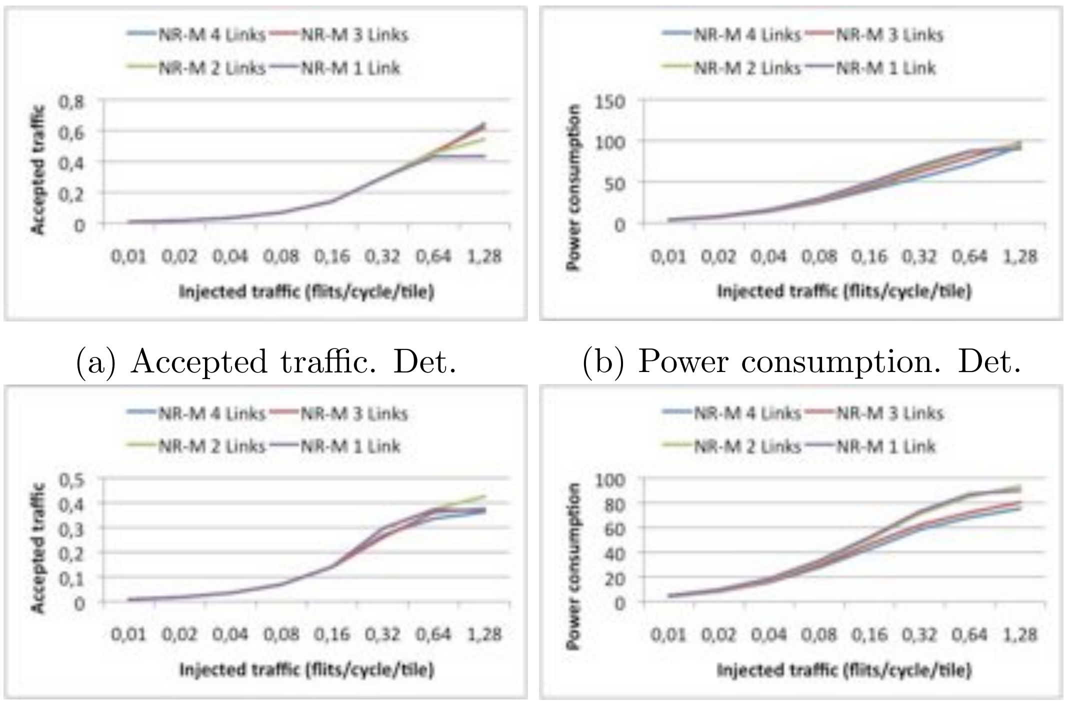
Figure 3.26: Accepted traffic and power consumption for 16-node systems. Injection variation in a hot-spot scenario.

quite similar, and the same when having two or one injection links, although the best option is always with four injection link.

Performance Analysis with Applications

Figure 3.28 and Figure 3.29, show a comparison between a NR-Mesh with 16 and 32 nodes, respectively, with four injection links and the same topology with one injection link per node removed in every new figure (for both 16- and 32-node systems). That is, graphs are shown for four links versus three links, four links versus two links and four links versus one link. For every comparison, we give a graph for the 16- and one for the 32-node system.

Comparing the last case (four links versus one link) for 16 and 32 nodes, we can see how the execution time increases about 10% on average in a 16- and 32-node systems using deterministic routing, and 13% and 14% using adaptive routing, respectively. The energy consumption increases about 20% in deterministic routing for both types of systems and 58% and 29% for adaptive



(a) Accepted traffic. Det. (b) Power consumption. Det.

(c) Accepted traffic. Adap. (d) Power consumption. Adap.

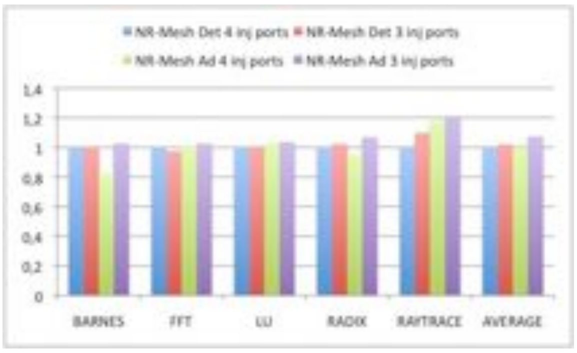
Figure 3.27: Accepted traffic and power consumption for 32-node systems. Injection variation in a hot-spot scenario.

routing for 16 and 32 nodes, respectively. We can see that as the number of internal links is lower, the NR-Mesh topology performs worse, as expected.

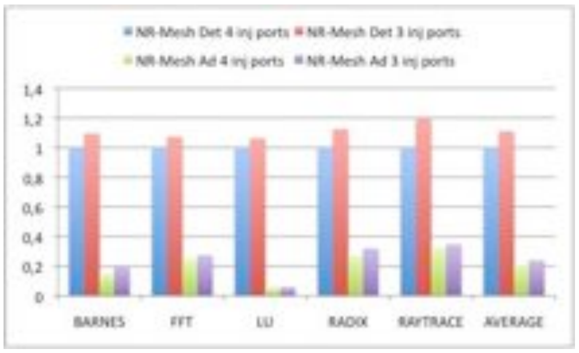
To clearly understand the behavior when we vary the number of injection links, we provide Table 3.6 where we show how performance decreases (normalizing results for 4 injection links in deterministic and adaptive routing).

Further Comparisons with Other Topologies

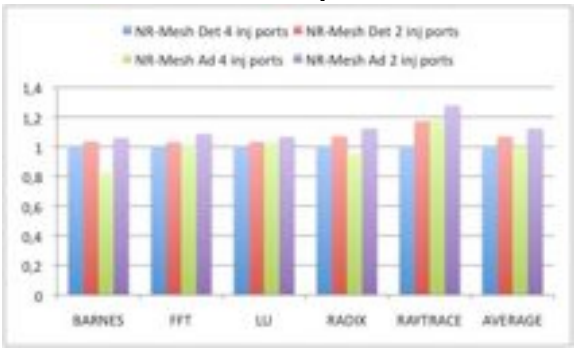
Finally, we compare the performance achieved when assuming different topologies and using the deterministic routing algorithms. Also, notice that in this evaluation links/switches are not switched off. The goal of this evaluation is to analyze the performance among similar topologies. In particular, the NR/2-Mesh and the concentrated mesh topology (C-Mesh) are compared with the NR-Mesh topology. Figure 3.30(a) shows the normalized execution time (relative to the NR-Mesh) for different applications and the average results achieved for a 16-tile system. In all the applications, the use of the NR-Mesh topol-



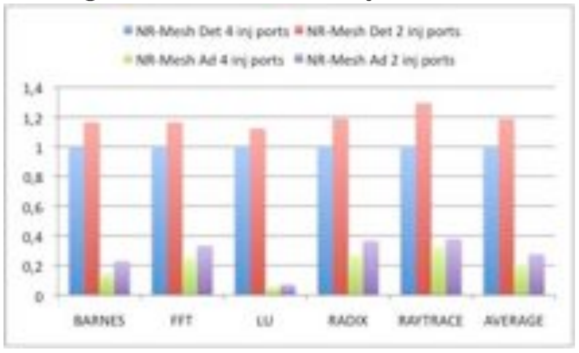
(a) Execution time comparison using four and three injection links.



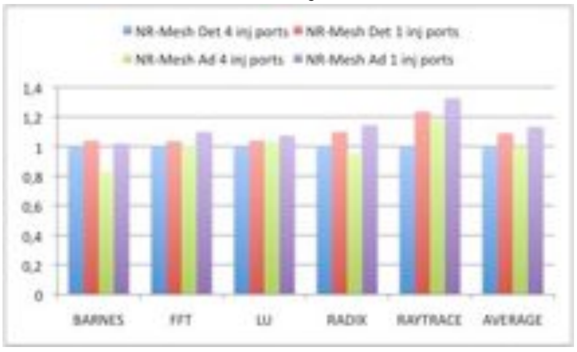
(b) Energy consumption comparison using four and three injection links.



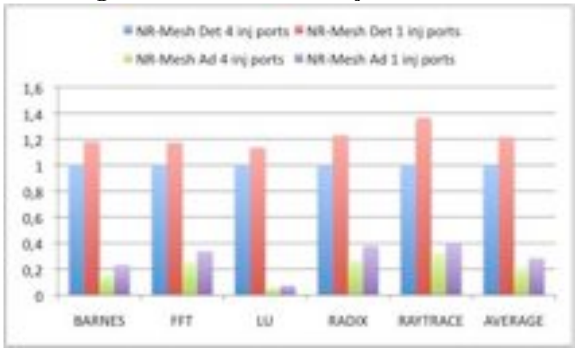
(c) Execution time comparison using four and two injection links.



(d) Energy consumption comparison using four and two injection links.



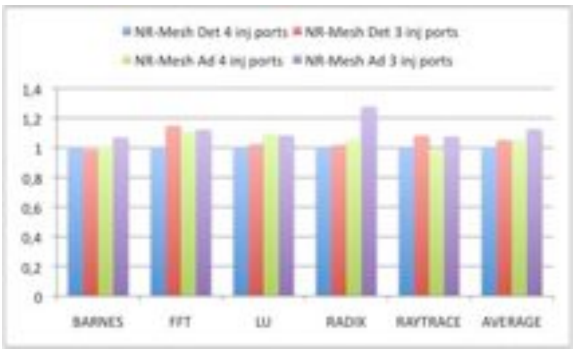
(e) Execution time comparison using four and one injection link.



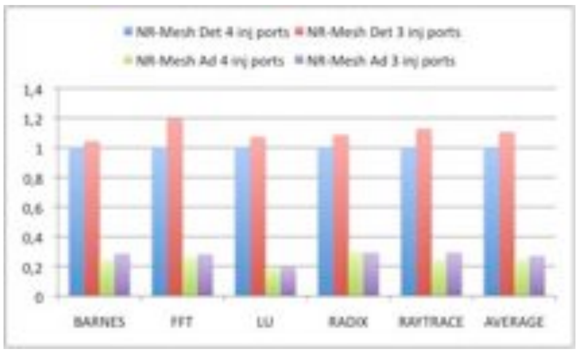
(f) Energy consumption comparison using four and one injection link.

Figure 3.28: 16-node link injection variation in the NR-Mesh.

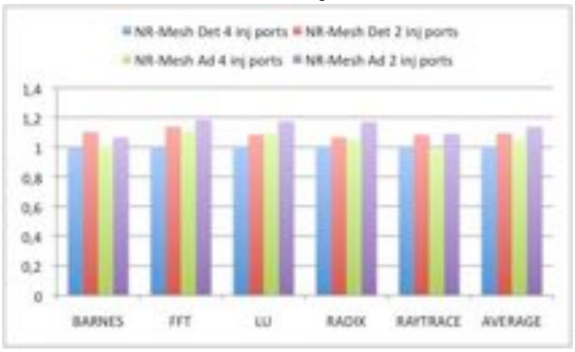
ogy helped in reducing execution time, on average by 14% when compared to the NR/2-Mesh and by 20% when compared to the C-Mesh topology. In the Apache application, execution time is reduced by 23% when compared with the NR/2-Mesh and by 52% when compared with the C-Mesh topology. The C-Mesh topology is the one with the lowest performance. Although it behaves



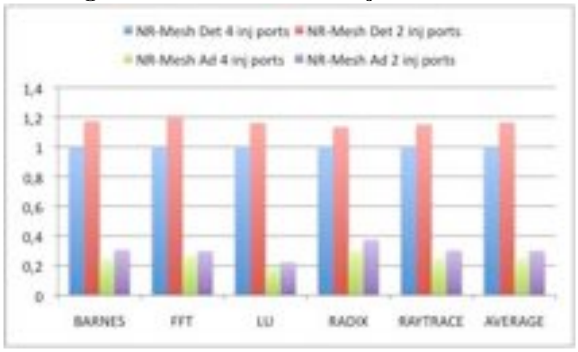
(a) Execution time comparison using four and three injection links.



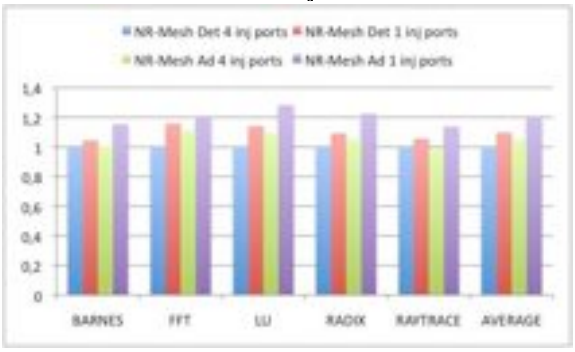
(b) Energy consumption comparison using four and three injection links.



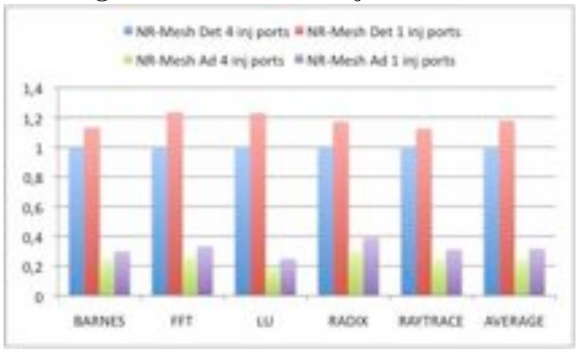
(c) Execution time comparison using four and two injection links.



(d) Energy consumption comparison using four and two injection links.



(e) Execution time comparison using four and one injection link.



(f) Energy consumption comparison using four and one injection link.

Figure 3.29: 32-node link injection variation in the NR-Mesh.

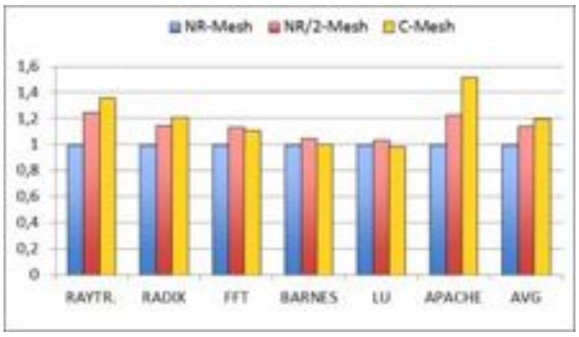
better in applications with low traffic requirements (e.g. Barnes), when traffic requirements increase (e.g. Apache), the lower bisection bandwidth of the C-Mesh behaves as a bottleneck and higher contention levels arise.

Figure 3.30(b) shows the same experiments but using a 32-tile system (4×8). Although relative performance of the different topologies is often

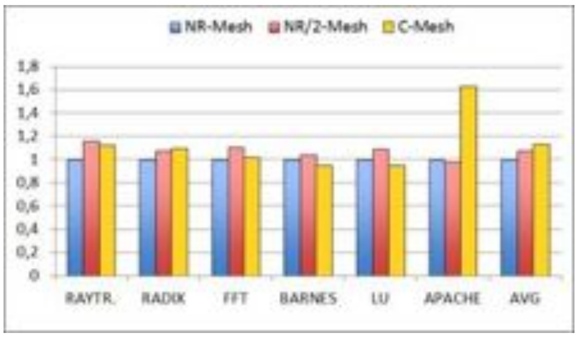
Deterministic Routing				
Injection Links	16N Time	16N Power	32N Time	32N Power
4 Link	1	1	1	1
3 Links	1.02	1.11	1.05	1.11
2 Links	1.07	1.19	1.09	1.17
1 Links	1.09	1.22	1.10	1.18

Adaptive Routing				
Injection Links	16N Time	16N Power	32N Time	32N Power
4 Link	1	1	1	1
3 Links	1.08	1.36	1.07	1.10
2 Links	1.12	1.56	1.08	1.22
1 Links	1.13	1.58	1.14	1.29

Table 3.6: 16- and 32-node execution degradation when removing injection links from the NR-Mesh in real applications.



(a) 16 (4 × 4) cores



(b) 32 (4 × 8) cores

Figure 3.30: Normalized execution time for Splash-2 applications and commercial workloads using different topologies.

similar to the 16-tile case, we can observe that differences between NR-Mesh and the other topologies are lower, on average a reduction of 8% and 13% when comparing with the NR/2-Mesh and C-Mesh respectively. This can be the result of a higher number of hops that messages need to take in the NR-Mesh topology. However, still, in systems of this size the NR-Mesh topology improves performance. Notice, however, that reducing performance is a secondary goal of the NR-Mesh topology. The main benefit of the topology is its ability to switch off components while exhibiting adaptiveness without degrading performance, as we analyzed in the previous sections.

3.4 Conclusions

In this chapter we have presented a flexible network on-chip topology referred to as NR-Mesh (Nearest-neighboR mesh). Each end node in this topology is connected to four different switches, which enables significant benefits when compared to other topologies. Using the NR-Mesh topology, the average latency in the network decreases significantly. Network contention is also reduced. Other benefits are fault tolerance and more efficient collective communication support. The benefit explored is the higher flexibility exhibited by the topology to inject and receive messages, enabling power aware routing algorithms.

We also proposed a power-gating mechanism that is able to switch off most components in the network. When combined with the NR-Mesh topology large energy savings are obtained. The NR-Mesh topology is fully exploited when adaptive routing is used. Adaptive routing has been combined with the power gating mechanism. To do that, power gating is used to switch on/off input ports and links in the switches. Due to the low utilization, as the NR-Mesh topology enables multiple alternative paths, energy savings (when compared to the deterministic 2D-Mesh topology) are clearly superior (75% on average for a 16-node CMP system reducing the execution time by 7% on average). Similar results were obtained for 32-node CMP system.

To sum up, the NR-Mesh topology provides more flexibility than the 2D-Mesh when both performance and power consumption become decisive. Most of the topologies proposed so far have the end node connected to a single switch, thus power consumption management becomes complex.

However, one of the potential drawbacks of the NR-Mesh topology is that we cannot decouple the network in smaller subnetworks, although we are using several injection ports.

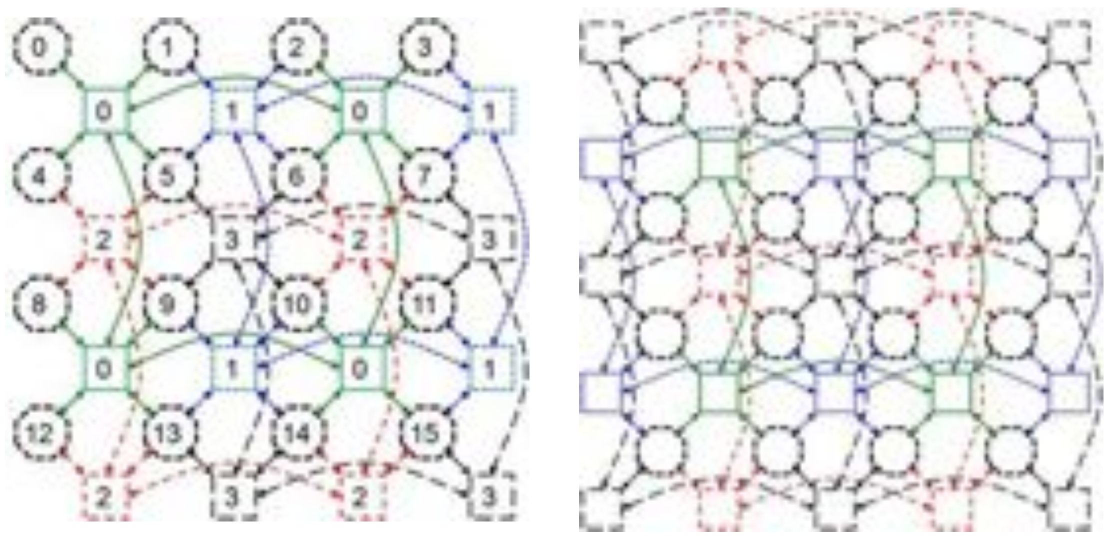
In the next chapter we propose a new network on-chip topology with the same philosophy, but in this case we are able to decouple the network in several smaller ones, improving the NR-Mesh topology results. This additional decoupling will provide more flexibility and an improved trade-off between performance and power consumption.

Chapter 4

PC-Mesh Topology

In this chapter, we take as a reference design the concentrated mesh (C-Mesh) network due to its good performance with low traffic loads (low latency). Moreover, we address the C-Mesh topology capacity limitations in high traffic loads with an alternative approach. We extend the C-Mesh topology in order to obtain savings in power with low traffic loads, as in the C-Mesh topology, and to avoid congestion with high traffic loads, achieving similar bisection bandwidth as of the 2D-Mesh, but still exhibiting low end-to-end latency. The proposed network on-chip topology is the parallel concentrated mesh (PC-Mesh). The new topology, as its own name indicates, is composed by different concentrated meshes. Specifically, we have four different concentrated meshes (or subnetworks). Each subnetwork is used only when necessary, otherwise the subnetwork is in off state and every switch is powered off. The key differentiating point is the fact the end node is connected to different subnetworks, thus, having more opportunities for power savings and performance optimization.

In order to achieve large power saving values, the PC-Mesh topology is enriched by an injection (selection) algorithm at the network interface of each node. The algorithm manages the parallel networks in order to maximize power savings without compromising performance. Similar to the NR-Mesh design, this algorithm works in parallel and closely with a distributed on/off power saving mechanism at every switch. The complementarity of both mechanisms is the key to achieve large power saving values. One key differing point



(a) PC-Mesh (b) FTPC-Mesh

Figure 4.1: Switch IDs and Node IDs in PC-Mesh network (left side), and Fault-Tolerant PC-Mesh network (right side).

from the previous NR-Mesh topology is the fact that now we have separate parallel networks and that traffic can be routed through separate networks without affecting/needing the other networks. In contrast, NR-Mesh represents a single network with multiple injection/ejection ports.

One important design point in a NoC system is also the routing algorithm. The routing algorithm, highly coupled with the topology, determines on every switch which output port every message must take in order to reach its destination. We assume the use of DOR algorithm by default, but with the new topology, every source-destination pair will have more than one alternative X-Y path, thus providing higher fault-tolerance rates than Mesh and C-Mesh topologies. Indeed, in order to maximize fault-tolerance, and still using the DOR algorithm, two different configuration layouts will be provided for the PC-Mesh topology, each one achieving similar characteristics. However, because of the different connectivity pattern, they will show different fault tolerance degrees. Notice that NR-Mesh also allowed different XY paths. However, now those paths will be totally decoupled and disjoint, thus enhancing the fault-tolerant degree of the final network.

4.1 PC-Mesh

Figure 4.1(a) shows the PC-Mesh topology. End nodes are represented as circles and switches as squares. When focusing only on switches, they build four disjoint subnetworks, thus, every switch belongs only to a single subnetwork. In our example, four 2×2 networks are build (each with a different link color and pattern). Notice that switches of different subnetworks are not connected between them. Although it is possible, short links would be used, it would increase the radix of the switch quite significantly, with the known problems this carries on [33]. Also, as we plan to switch off entire subnetworks, they should be isolated from the other networks in order to get an efficient power management strategy.

When focusing on end nodes, we can see that they have different connectivity patterns. Nine nodes are connected to four switches and the remaining ones to fewer switches. In particular, the top left-most node is connected to a single switch and the remaining boundary nodes are connected to two switches. This is the same pattern as the NR-Mesh topology. As end nodes are connected to some switches, additional logic is required at the network interface. The logic will implement an algorithm that selects the switch (network) to use on a per packet basis (described in the next section). As the case of NR-Mesh, this injection algorithm has negligible impact.

To increase the fault tolerance deegree of the network, we add a row and a column of extra switches, leading to the topology shown in Figure 4.1(b). Now, the four C-Mesh topologies are different in size and number of switches, however, every end node is connected now to the four parallel subnetworks. This configuration will be referred to as Fault-Tolerant PC-Mesh (FTPC-Mesh). It is important to note that the FTPC-Mesh configuration uses more switches than end nodes, and at first sight this will render more power consumption. However, it is necessary to note that these extra switches will be also dynamically powered on and off, thus, used only when really needed.

4.1.1 Tile-Based Design

One important aspect in CMP systems is to use a tile-based design as it significantly reduces the design effort. Thus, the proposed PC-Mesh topology

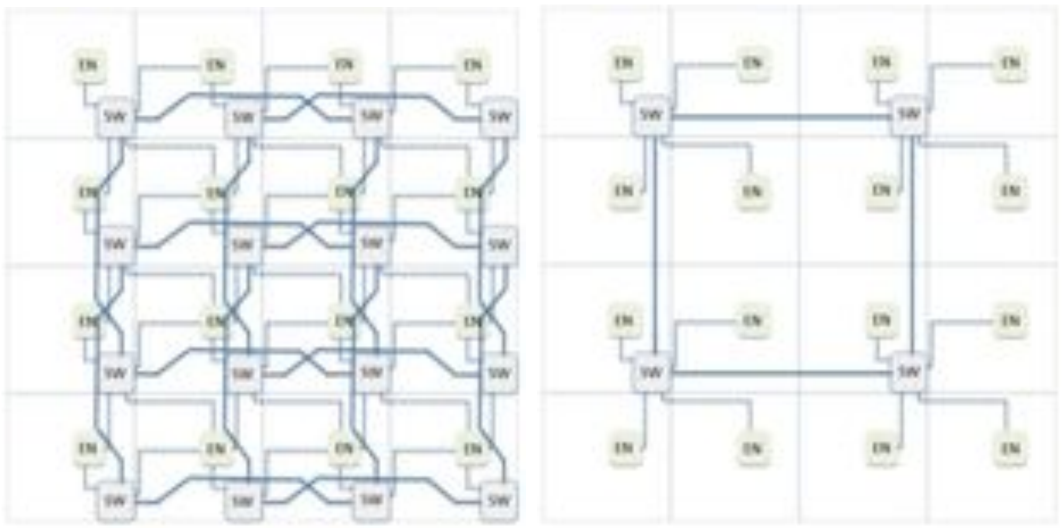


Figure 4.2: 16-tile design assumed for the PC-Mesh topology (left side) compared to the C-Mesh one (right side).

must be adapted to a tile-based structure. Indeed, the PC-Mesh topology is similar to the C-Mesh topology. The key differentiating points are the connection of each end node to more than one subnetwork and the existence of a switch on every tile (as the 2D-Mesh). Figure 4.2 shows a possible tile design for the PC-Mesh topology compared to the C-Mesh one. The differences with the C-Mesh topology lay on the extra links connecting end nodes to switches and the higher number of switches. Overlapped links can effectively be routed over the logic through higher metal layers.

4.1.2 Injection Algorithm

In this section we describe the injection algorithm assuming a 4×4 PC-Mesh topology. The goal of the algorithm is to adapt the use of subnetworks to the current injection load of the end node. The four subnetworks are used in an ordered way and the algorithm is used on a per packet basis. The logic has been modeled in Verilog and results in terms of power, area and delay show negligible overheads (compared to a switch) as was the case for the NR-Mesh topology.

As XY routing is assumed and an end node now can be reached through four subnetworks, now we need to compute the final switch destination through

Subnetwork	X	Y	Switch	Node ($b_3b_2b_1b_0$)
N_0	1	1	3	1010
N_1	0	1	2	1010
N_2	1	0	1	1010
N_3	0	0	0	1010

Table 4.1: Example for different switches which can reach the node 10.

each possible subnetwork. The labels of end nodes and switches are depicted in Figure 4.1(a). End nodes are labeled from 0 to 15 and switches belonging to the same subnetwork from 0 to 3. As can be seen, end node 10 can be reached through switch 3 in subnetwork 0 (green), 2 in subnetwork 1 (blue), 1 in subnetwork 2 (red), and 0 in subnetwork 3 (black).

Upon reception of a packet to be injected into the network, the algorithm proceeds with three stages computed in parallel. In the first stage, to compute the switch destination reaching the end node, we derive the following equations (assuming destination end node is coded in four bits $b_3b_2b_1b_0$, the network to use is represented by four signals n_x, and x is the subnetwork id):

$X = b_1 \times ((n_1|n_3) \times b_0)$

$Y = b_3 \times ((n_2|n_3) \times b_2)$

In particular, this equation represents the following cases:

$X_{N0} = b1$

$Y_{N0} = b3$

$X_{N1} = b1 \times b0$

$Y_{N1} = b3$

$X_{N2} = b1$

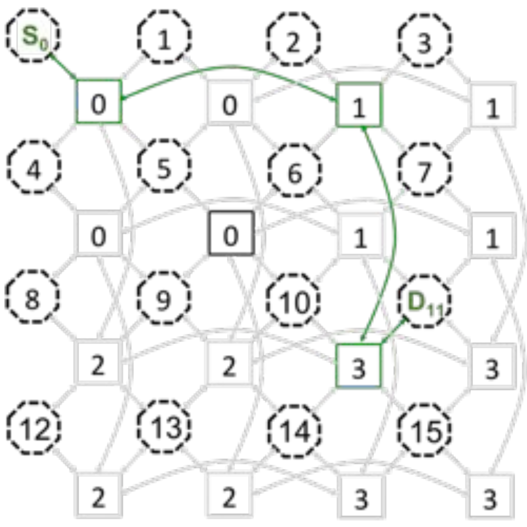
$Y_{N2} = b3 \times b2$

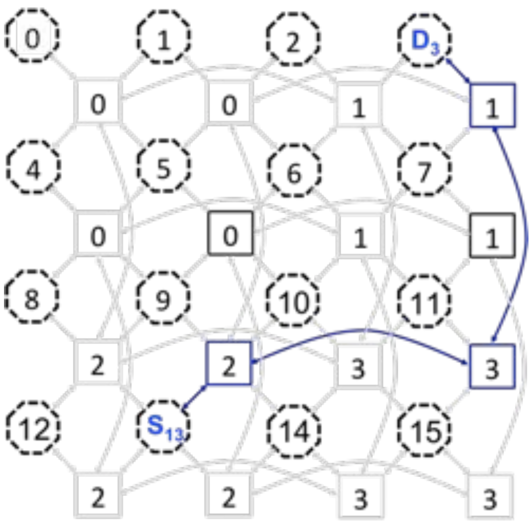
$X_{N3} = b1 \times b0$

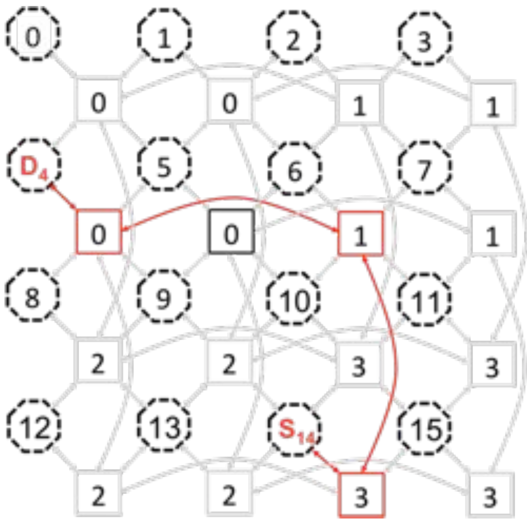
$Y_{N3} = b3 \times b2$

As an example, Table 4.1 shows the destination switches for reaching node 10 (coded as 1010) through the different subnetworks.

This equation is compatible with the lower connectivity of the end nodes along the first row and column of switches. A similar equation can be deduced


(a) Subnet N_0 ($S_0 - D_{11}$)


(b) Subnet N_1 ($S_{13} - D_3$)


(c) Subnet N_2 ($S_{14} - D_4$)

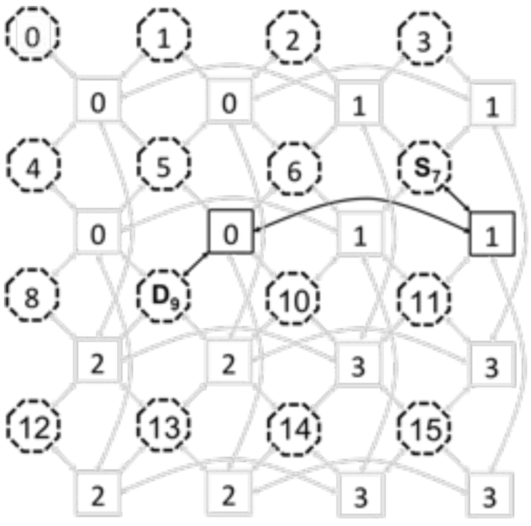

(d) Subnet N_3 ($S_7 - D_9$)

Figure 4.3: X-Y in every graph is followed from S_x to D_x represented with the same color as the subnetwork, through switches of the same color.

for the FTPC-Mesh topology. Figure 4.3 shows an example of 4 pairs of sources and destinations end nodes, each one taking a different subnetwork. Every source follows X-Y through the subnetwork with the same color.

In parallel, at the second stage, the number of enqueued flits F_x for each subnetwork is checked. A threshold register is used (T_h) to compare with. Several thresholds have been tested, and the chosen values are explained in the evaluation section. For each subnetwork x, a T_x signal is computed representing whether the threshold has been exceeded or not ($T_x = F_x > T_h$). This threshold is used for low injection rates of the node (when injection is below 20%). For high injection rates (superior to 20%), the T_h threshold is considered to be zero (enabling all the networks to be used). This last value was chosen after large simulation tests. We have experimented that when experiencing injection rates superior to 20%, the PC-Mesh performs better with this situation (enabling all subnetworks).

As different path lengths are available from the same source-destination pair, depending on the subnetwork used[1], in the third stage the end-to-end node distances are computed. The manhattan distance (MD_x) for each subnetwork is obtained by computing the distances along X and Y dimensions: $MD_x = abs(X_x - XC_x) + abs(Y_x - YC_x)$ where XC_x and YC_x are the coordinates of the switch attached to the injection node for the particular x subnetwork (these values are constant and, thus, there is no need to compute them at every packet injection).

After the three parallel stages, at the fourth stage, the different manhattan distances are compared in order to prepare the selection of the final subnetwork. To do this, three CMP_x signals are computed, where manhattan distances are compared (the CMP_i signal just checks if the subnetwork i provides equal or shorter path for the destination end node):

$CMP_1 = MD_1 <= MD_0$
$CMP_2 = MD_2 <= min(MD_1, MD_0)$
$CMP_3 = MD_3 <= min(MD_2, MD_1, MD_0)$

In the final stage, the subnetwork to use is computed, based on the CMP_x, T_x and S_x signals. The S_x signals indicate whether the attached switch in the x subnetwork is powered on or not. SEL_x signals are computed appropriately as follows:

$SEL_0 = (T_0 \times S_0) + (\overline{SEL_1} \times \overline{SEL_2} \times \overline{SEL_3})$
$SEL_1 = \overline{SEL_0} \times T_1 \times CMP_1 \times S_1$
$SEL_2 = \overline{SEL_1} \times \overline{SEL_0} \times T_2 \times CMP_2 \times S_2$
$SEL_3 = \overline{SEL_2} \times \overline{SEL_1} \times \overline{SEL_0} \times T_3 \times CMP_3 \times S_3$

Networks are prioritized based on the previous equations. Notice that,

[1]As an example, end nodes 5 and 10 are one hop away through subnetwork 3, two hops away through subnetworks 1 and 2, and three hops away through subnetwork 0.

in order to optimize the most frequent case (low injection rates), if T_0 is set (meaning enqueued flits for the first subnetwork is below the threshold), then the remaining logic can be switched off.

With the designed algorithm, the PC-Mesh network will allow most of the subnetworks to be switched off as traffic will be directed to only a subnetwork, thus resembling a C-Mesh network for low traffic. For high traffic rates, the PC-Mesh network will use an increasing number of subnetworks, thus, maximizing performance. The benefit of PC-Mesh lies on the fact of enabling subnetworks only when necessary.

Therefore, the PC-Mesh topology will always use only one subnetwork in low traffic scenarios. When the threshold is reached, then, all the subnetworks will be enabled. The subnetwork used will depend on the number of flits enqueued in the end node. Also, the injection path chosen will depend on the distance to destination and the availability of the port, always prioritizing the best scenarios, that is, more enqueued flits and shorter distances.

4.2 Power Management Algorithm

As done previously, now we combine the injection algorithm with a power management algorithm. The goal is to switch off complete switches (not only the ports of the switch) while they detect no traffic activity. Switches, however, will cooperate with neighboring switches and attached end nodes. The basic goal of the algorithm is to maximize the time switches and links are powered down, thus enabling large savings in static power. The designed injection algorithm will avoid, whenever possible, the use of powered down switches. We focus exclusively on the switches, considering both, power and clock gating.

We elaborate the power management mechanism assuming a canonical 4-stage pipelined switch, virtual channel support, and stop&go flow control implemented. The first stage (IB) of the switch allocates the flit in the input buffer of the port, the second stage (RT) performs the XY routing computation, the third stage (VA_SA) the virtual channel allocation and the switch allocation, and the final stage the effective switch traversal (ST).

The switch has been enriched with the PML (power management control logic). The PML is in charge of deciding when powering down the entire

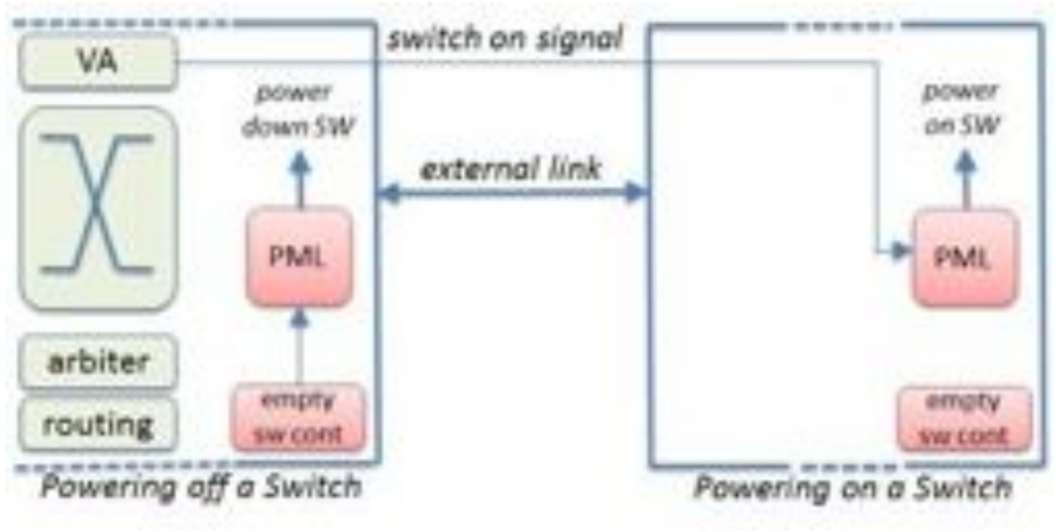


Figure 4.4: Logic for the power management algorithm. The main difference with the PML in the previous chapter is that here we power off the entire switch.

switch and its attached clock. These actions are made locally within the switch. Figure 4.4 shows the new logic. For switching off, PML measures the number of cycles the switch is empty of flits (buffer use of every input port). Upon reaching a threshold (T_{swoff}) the switch is powered down as well as the clock feeding the switch. The key difference with NR-Mesh now lies on the fact that before we powered down at the port level and now we do it at the switch level. This is because now we have decoupled networks.

The assumptions on cycles to power down and on components, as well as the number of cycles to save power, are the same as the ones used in the previous chapter for NR-Mesh. The PML logic is waken upon arrival of a control signal from the upstream connected devices (neighbor switches or attached end nodes). In order to overlap 2 of the 3-cycles needed to wake up the switch, we have augmented the VA_SA stage of the switch to trigger the $switch_{on}$ signal, reaching the next switch at the same cycle (see Figure 3.16 for a description of the switch).

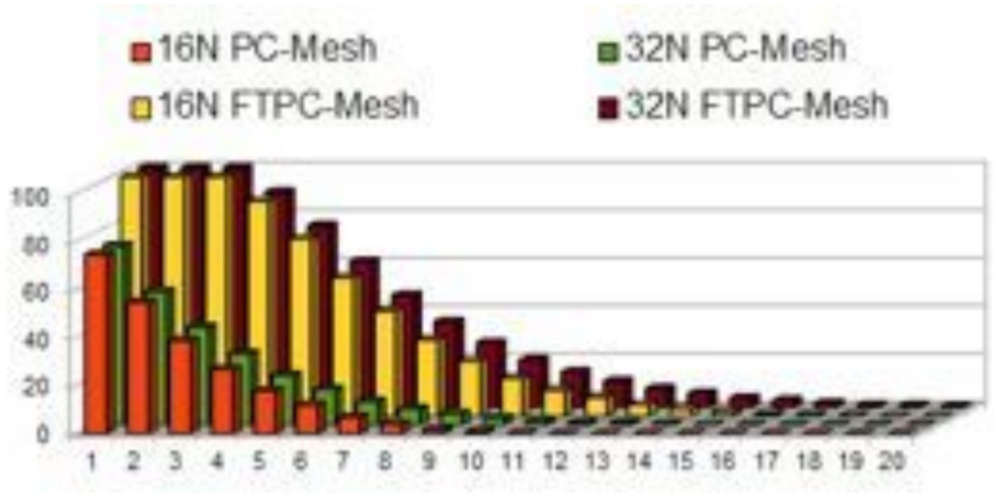


Figure 4.5: Faul tolerance comparison for the PC/FTPC-Mesh topologies. X-Axis shows the number of faulty switches. Y-Axis shows the fault tolerance percentage for each number of faulty switches in each topology.

4.3 Fault tolerance

One of the key properties of the PC-Mesh network is its fault-tolerance degree (which should be higher than that of the NR-Mesh topology). Providing parallel networks increases significantly the reliability of the entire network. In this section we analyze this property by computing the probability that one or more failed component (switches and the attached links) leave the end nodes unconnected. We provide the analysis for both PC-Mesh and FTPC-Mesh network topologies and assume the use of the DOR routing algorithm.

The hypergeometric distribution (which is a discrete probability distribution that describes the probability of k events in n draws) has been used to obtain the fault tolerance for the PC-Mesh and FTPC-Mesh topologies. Figure 5.6 shows the obtained results for both versions of the topology: PC-Mesh and FTPC-Mesh, when using 16 and 32 cores. As can be seen, FTPC-Mesh is able to tolerate 3 failures as has four complete parallel networks. However, the PC-Mesh version obtains only 75% of fault-tolerance for a single switch failure. Indeed, having one end node connected to a single switch prevents achieving 100% fault tolerance coverage for the one-switch failure case. Even though, when compared to the C-Mesh and 2D-Mesh topologies results are good, since these topologies are not able to tolerate, assuming the DOR routing algorithm, a single switch failure in any case.

The figure also shows the graceful degradation of the fault tolerance for the FTPC-Mesh network. For 6 switch failures, the network is still able to achieve a level of 60% in fault tolerance.

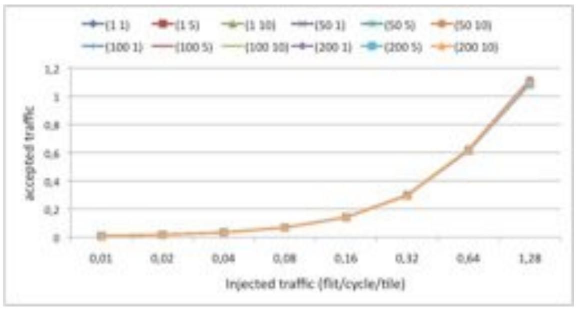
4.4 Performance Evaluation

Now we evaluate the new topology and the injection algorithm at the network interface. We compare the topology with the 2D-Mesh and C-Mesh topologies. DOR is used in all the topologies and with the same number of virtual channels (VCs). VCs are used to avoid protocol-level deadlocks induced by the cache coherency protocols.

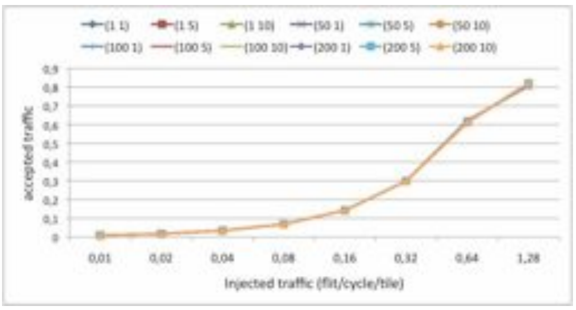
In all the topologies switches (including clock signal) are powered on and off, following the algorithm provided previously. The number of cycles for powering switches on and off has been obtained from [16]. The delay to power on the switch again (including the clock signal) is fixed to 3 cycles. The threshold to decide a switch needs to be powered off will depend on the current traffic and will be applied on a per switch basis (distributed and local off actions). In a first analysis we evaluate the robustness of the thresholds used in the injection algorithm (T_h) and the threshold used in the switches (T_{swoff}). This analysis will allow us to fix those thresholds. Once fixed, we analyze the different topologies under a wide range of traffic, with synthetic traffic patterns and real applications.

4.4.1 Threshold Analysis

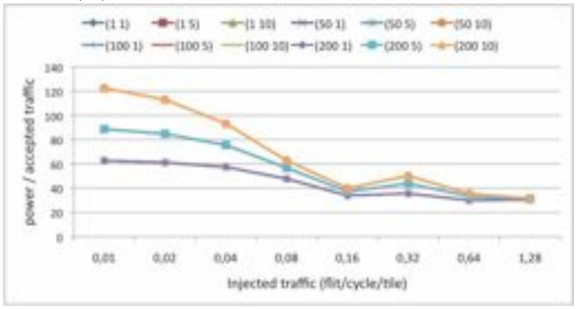
The PC-Mesh network has two key threshold parameters. T_h (shown in the Figure 4.6 and Figure 4.7) is used on every end node to decide which subnetwork has to be used to inject a message. Whenever a queue reaches that threshold the next subnetwork is set as a candidate to be used. This threshold is analyzed under four different values: 1, 50, 100, and 200. The threshold is measured as flit occupancy of the injection queue. The second parameter (T_{swoff}) relates to the switches being off. It indicates the number of cycles the switch needs to have all its queues empty in order to be switched off. We evaluate three values for this threshold: 1, 5, and 10 cycles. Figure 4.6 shows throughput and power consumption when using uniform distribution of messages as the traffic pattern in a 4×4 and a 8×4 network. Twelve combinations of threshold values have been explored. Each case is labelled with the T_h value followed by the T_{swoff} value.



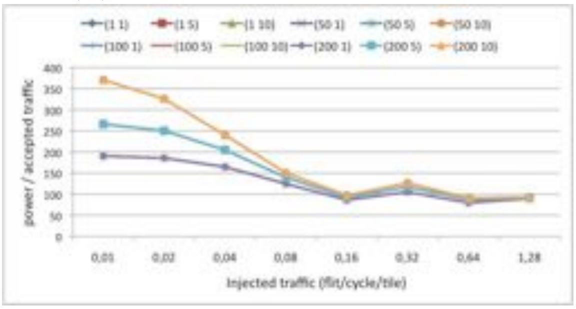
(a) accepted traffic, 16 nodes



(b) accepted traffic, 32 nodes



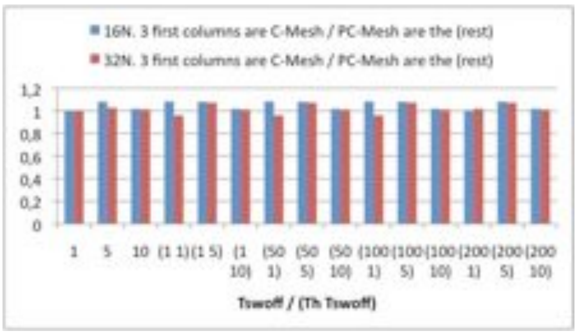
(c) power per flit, 16 nodes



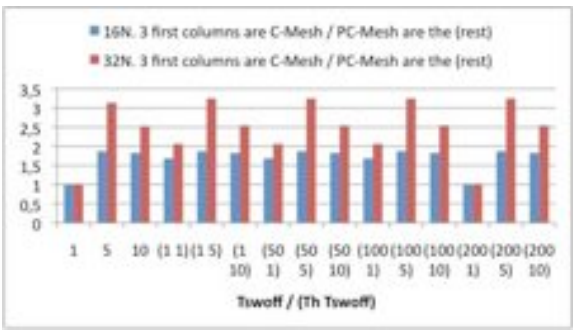
(d) power per flit, 32 nodes

Figure 4.6: Threshold analysis. Synthetic traffic patterns (uniform distribution of message destinations). Accepted traffic measured as flits/cycle/tile and power consumption as W.

As can be seen, network throughput is largely insensitive to the threshold values. Only marginal differences are seen for very high traffic loads. In any case, the network is always ready to accept messages and the mechanism to switch on the network components is efficient. Thus, no delays are incurred. However, differences are much greater in terms of power consumption. Indeed, for very low traffic conditions the threshold selection impacts power consumption heavily. The T_{swoff} parameter is the one that impacts the highest in power consumption in very low traffic conditions. Indeed, a threshold value of 1 allows to save the maximum power from switches. A threshold of 1 achieves 100% power reduction when compared with a threshold of 10. The value of 5 for T_{swoff} lies in between, as expected. For the T_h value we can see that for



(a) Execution time



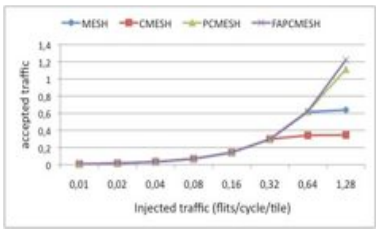
(b) Energy consumption

Figure 4.7: Different subnetwork thresholds cases. Results are normalized for 16- and 32-node system to the C-Mesh topology where $T_{swoff} = 1$. Then, T_h is shown for C-Mesh case and (T_{swoff} T_h) are shown for PC-Mesh.

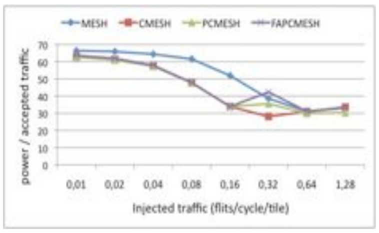
the same T_{swoff} value all perform the same both in performance and power consumption.

One special case is the 0.32 injection rate point where power consumption per flit slightly increases. At this point switches are toggling between on and off modes thus incurring in a small penalty. Prior to this injection point only the first sub-network is working and the rest of sub-networks are down. After this critical injection point all the sub-networks are stable and working because of the higher traffic demand.

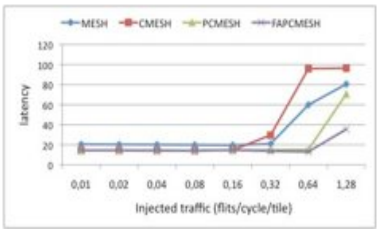
From these results we could conclude the best threshold values would be 200 for T_h and 1 for T_{swoff}. However, notice that the injected traffic is steady and does not have burstiness. In order to better assess the correct threshold values we also run an application (FFT) with different threshold values, and compare the results with the C-Mesh network also with varying T_{swoff} values (notice that C-Mesh has only one injection link thus there is no T_h). Figure 4.7 shows the results for the FFT application, both the execution time and the power consumption. As can be seen, the results with T_{swoff} set to 1 are the ones that achieve the largest amount of power saving. However, for T_h we get differing results. Indeed, FFT consumes the same power in the network with values of 50 and 100, however it consumes less power with a value of 200. This is due to the burstiness of the application that leads to opening many subnetworks thus having larger consumption levels. From these results, and in combination with synthetic traffic results we conclude that a safe value for



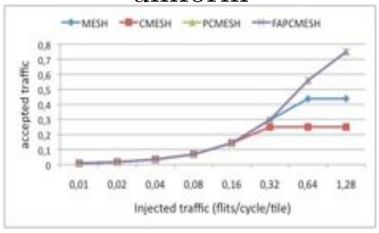
(a) accepted traffic, uniform



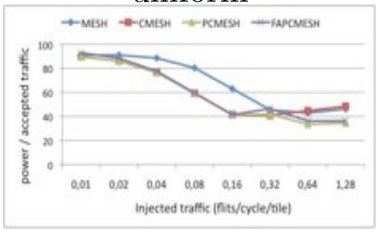
(b) power per flit, uniform



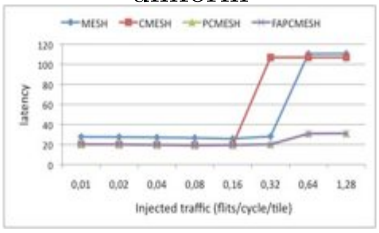
(c) average latency, uniform



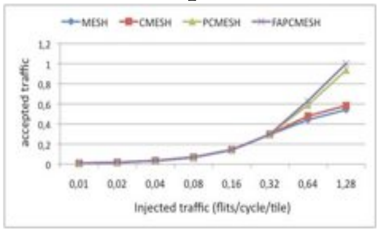
(d) accepted traffic, bit-complement



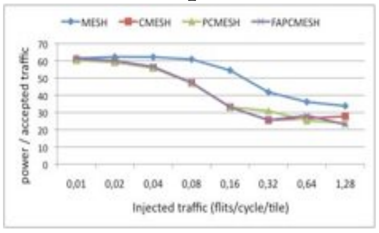
(e) power per flit, bit-complement



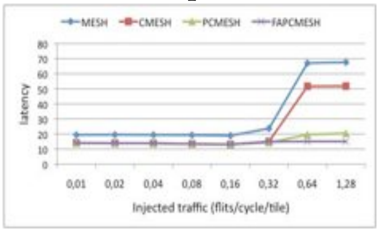
(f) average latency, bit-complement



(g) accepted traffic, bit-reversal



(h) power per flit, bit-reversal



(i) average latency, bit-reversal

Figure 4.8: 16-node synthetic traffic comparison. Accepted traffic measured as $flits/cycle/tile$, power consumption as W and latency as cycles.

T_h is 200.

4.4.2 Synthetic Traffic Results

Once we set the threshold, we compare the PC- and FTPC-Mesh with the 2D- and C-Mesh. Figure 4.8 shows accepted traffic and power results for a 16-node system (4×4 topology) under synthetic traffic patterns (uniform, bit-reversal, and bit-complement). Figures in the left show accepted traffic, figures in the center show the power consumption per flit unit (namely, overall power consumed divided by accepted traffic) and figures in the right side show the average latency.

First thing to note is the higher throughput of both PC-Mesh and FTPC-

Mesh, in the three scenarios. The PC-Mesh and FTPC-Mesh topologies exhibit shorter paths and provide parallel paths not available in the 2D-Mesh network, thus lowering contention in high traffic conditions. On the other hand, the limited bisection bandwidth of the C-Mesh topology limits its maximum throughput, low below the 2D-Mesh network (except for the bit-reversal traffic pattern where is almost equal).

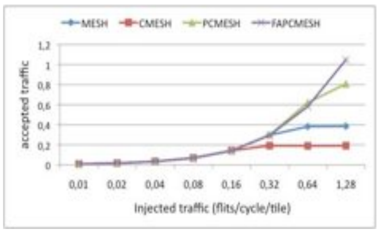
Now, looking into the power consumption figures, we can see how PC-Mesh, FTPC-Mesh and C-Mesh topologies exhibit a similar power consumption rate per accepted traffic unit. The 2D-Mesh topology achieves worse results due to its higher number of switches traversed. Also, at high injection rates, both PC-Mesh and FTPC-Mesh obtain in some cases a small reduction in power when compared with the 2D-Mesh and C-Mesh topologies. Anyway, we can conclude that PC-Mesh and FTPC-Mesh exhibit the same degree of power efficiency (per delivered flit) as C-Mesh. It is worth mentioning that this does not mean all the topologies deliver the same amount of traffic (as seen in the previous figures).

The results for the latency shows how the 2D-Mesh is the worst topology in low traffic conditions, although the C-Mesh performs still worse in high traffic scenarios. Then, looking at the figures it is clear that PC/FTPC-Mesh is the most suitable topology due to its adaptability for every situation.

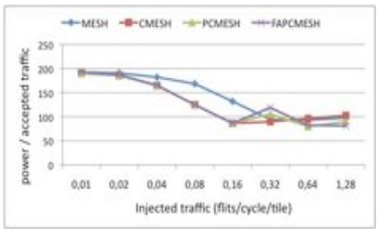
Similar results among different networks for a 32-node system (4×8 topology) have been achieved (shown in Figure 4.9), although there is a general drop in accepted traffic. Also, power and latency increases for every topology.

4.4.3 Application Execution Time and Power Results

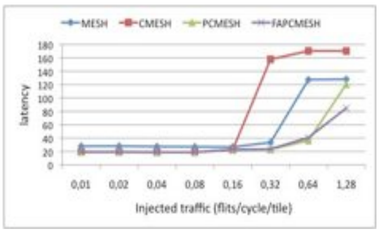
In this section we evaluate application's performance, and power savings, when real applications are run. Figure 4.10 shows normalized execution time and network energy consumption of different applications when using different topologies for the 16- and 32-node configurations. The execution time is reduced when using the C-Mesh, PC-Mesh, FTPC-Mesh. On average, a reduction of 20% is achieved when compared with the 2D-Mesh topology. This is mainly due to the reduction in the path length. The take away message here is that performance of the application is not sacrificed in the new topologies. The benefit comes also with the network energy consumption values. These



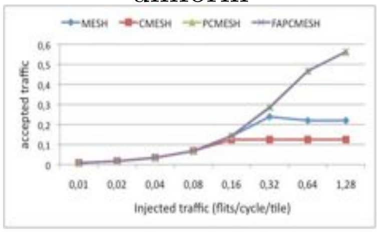
(a) accepted traffic, uniform



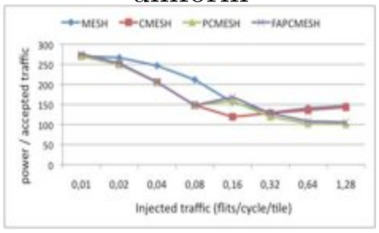
(b) power per flit, uniform



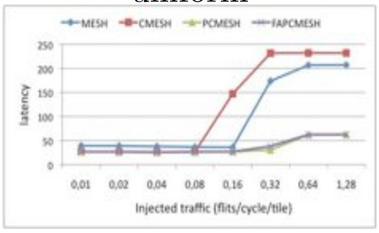
(c) average latency, uniform



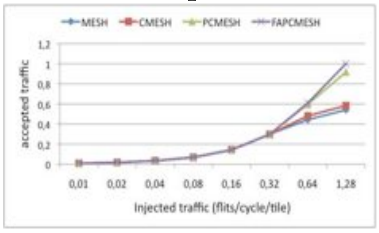
(d) accepted traffic, bit-complement



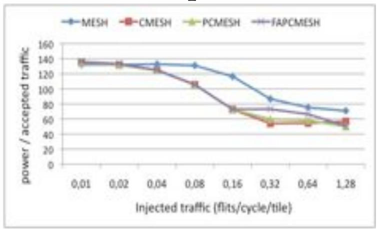
(e) power per flit, bit-complement



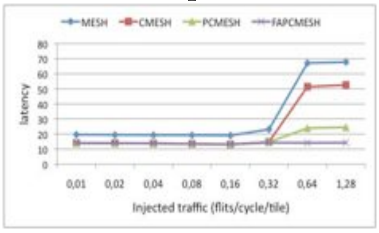
(f) average latency, bit-complement



(g) accepted traffic, bit-reversal



(h) power per flit, bit-reversal



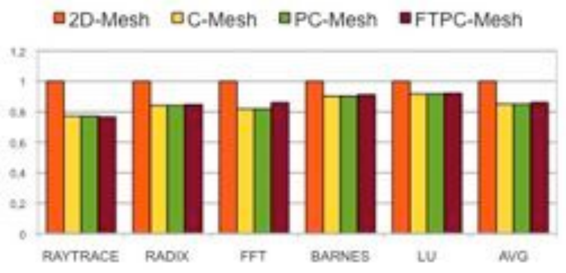
(i) average latency, bit-reversal

Figure 4.9: 32-node synthetic traffic comparison. Accepted traffic measured as flits/cycle/tile, power consumption as W and latency as cycles.

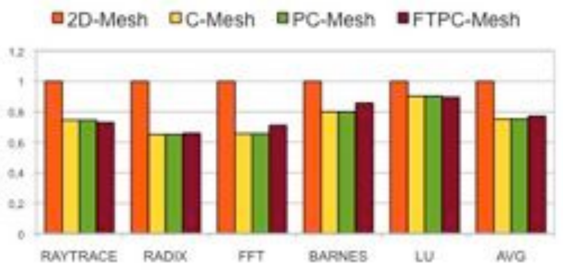
results demonstrate the on/off mechanism can be used effectively in the new topologies, together with the injection algorithm at the network interfaces. Average results indicate savings of 20% when compared with the 2D-Mesh topology.

4.4.4 Results with Overloaded Systems

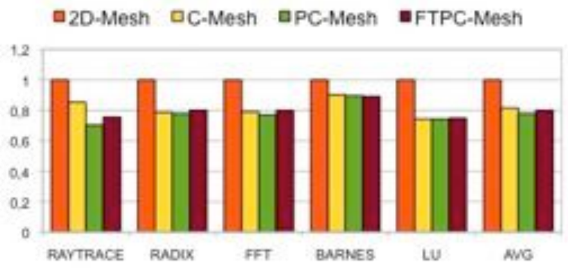
The previous results clearly indicate the PC-Mesh topology behaves like the C-Mesh network. This is due to the low injection rates of the tested applications. In order to test the network in a much stressed scenario, and to identify the potentials of the proposed networks, we evaluate every topology with a background synthetic traffic in addition to the application's traffic. This traf-



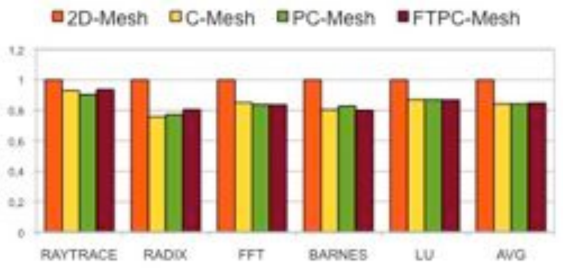
(a) 16-cores, execution time



(b) 16-cores, energy consumption



(c) 32-cores, execution time

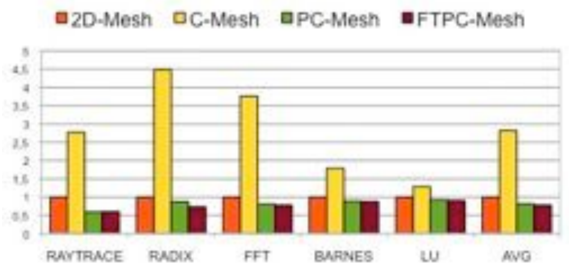


(d) 32-cores, energy consumption

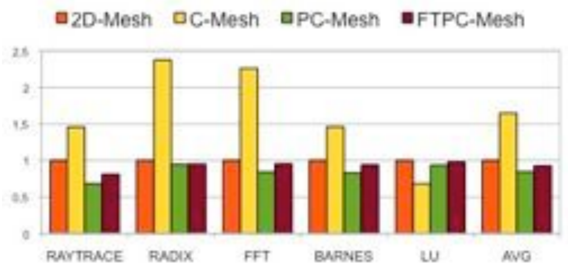
Figure 4.10: Application execution time and network energy consumption.

fic model represents a more stressed system where multiple applications will be running at the same time. In such scenario the network will need to facilitate much higher bandwidth. Figure 4.11 shows the performance and power consumption values for both 16-node and 32-node systems. Background uniform traffic of 12% (for 16-node systems) and 6% (for 32-node systems) of total injection rate is injected. As can be seen, the C-Mesh network is not able to run efficiently the application's traffic. Application's execution time is severely impacted and in most cases, more than doubled. On average, execution time is tripled. For power consumption values, on average, the C-Mesh network now achieves a 50% higher consumption values than the other topologies. When comparing 2D-Mesh with the PC-Mesh topology, and with the FTPC-Mesh one, the 2D-Mesh (normalized result) achieves higher execution time and larger power consumption values. Therefore, the PC-Mesh topology is the one that achieves better performance. In particular, PC-Mesh reduces execution time and power consumption by 20% on average for both 16- and 32-node systems.

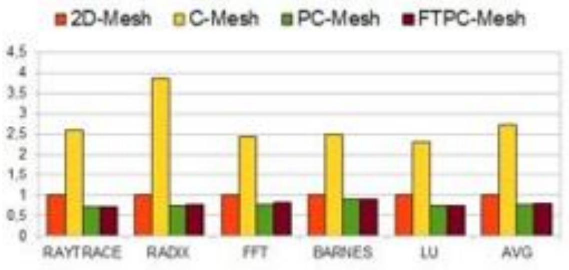
Another interesting point is when different traffic classes can be mapped on different networks in PC-Mesh. Indeed, this is a key property that is not



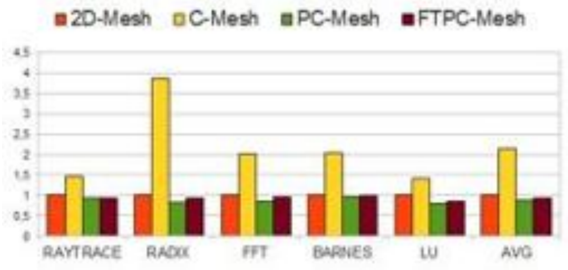
(a) 16-cores, execution time



(b) 16-cores, energy consumption



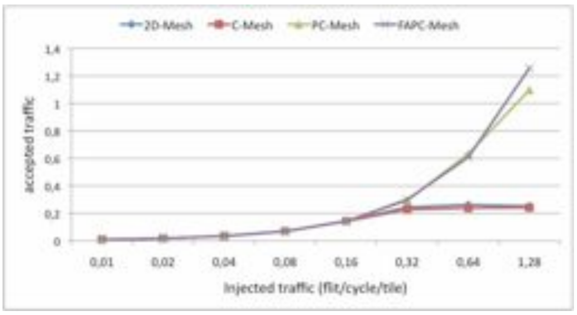
(c) 32-cores, execution time



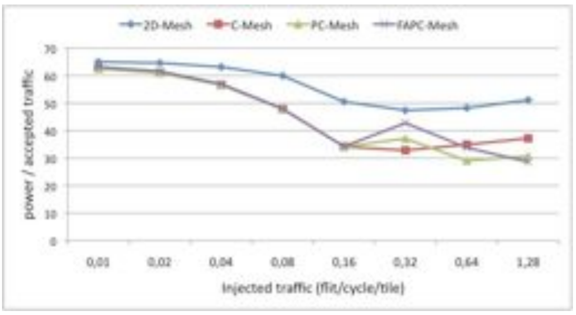
(d) 32-cores, energy consumption

Figure 4.11: Application execution time and network energy consumption with a background of 12% (16-node) and 6% (32-node) of synthetic traffic.

available in 2D-Mesh and C-Mesh topologies. To demonstrate the potential, we have analyzed the impact of mapping a congested traffic on a PC-Mesh subnetwork while injecting the full range of traffic through other networks. Figure 4.12 shows the performance and the energy consumption in hot-spot scenarios for a 16-node configuration using uniform traffic. As can be seen, the mapping of the hotspot traffic on a particular PC-Mesh subnetwork allows to attain much larger performance levels, even with similar energy consumption values (per accepted flit). In particular, the 2D-Mesh network reduces the maximum accepted traffic by 160% and C-Mesh by 40%, compared with the figure 4.8(a) (uniform traffic without a hot-spot scenario). The PC-Mesh topology does not get impacted by the hotspot scenario. Similar results have been obtained also for the 32-node configuration. These results open the door to a smart use of the four subnetworks available in the PC-Mesh network on-chip topology.



(a) Accepted traffic, hotspot



(b) power per flit, hotspot

Figure 4.12: Accepted traffic and power consumption in hot-spot scenarios. 16-node system. Accepted traffic measured as flits/cycle/tile and power consumption as W.

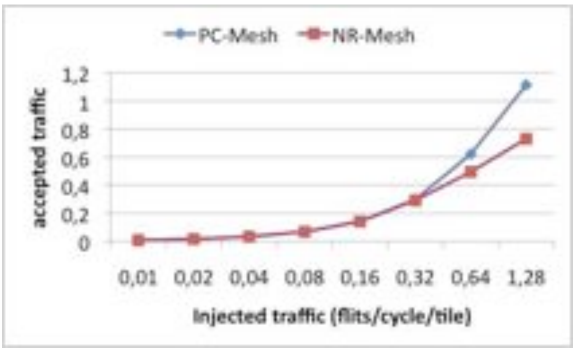
4.5 NR-Mesh versus PC-Mesh topology

Now in this section, we provide results comparing the NR- and PC-Mesh topologies. We show results for synthetic traffic and real applications, both of them using 16- and 32-node systems.

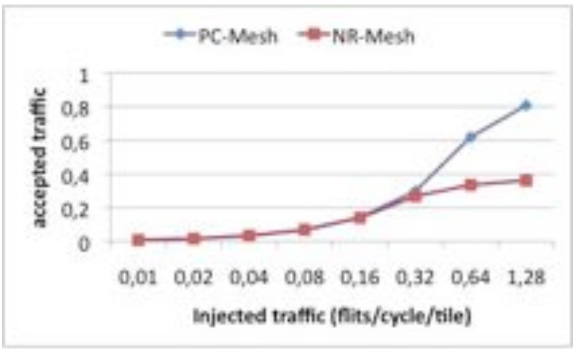
The purpose of this section is to demonstrate which topology performs better. Once we obtain the results, we can continue improving the best one. However, this is left for the next chapter.

4.5.1 Uniform Synthetic Traffic

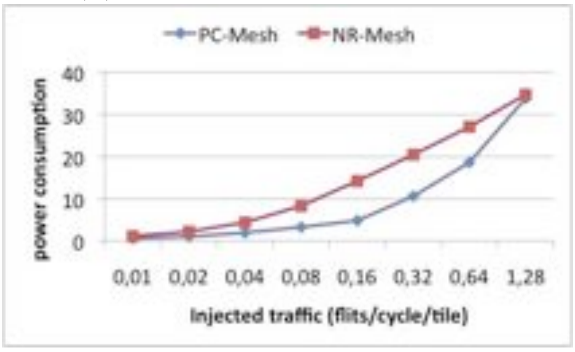
First, looking at the accepted traffic (Figure 4.13(a)(b)) with 16 and 32 nodes, respectively, we can see how the PC-Mesh outperforms the NR-Mesh topology when a good quantity of traffic arises. When the injected traffic is low, both network on-chip topologies are able to accept the same traffic. However, in Figure 4.13(c)(d) for 16 and 32 nodes, respectively, the power consumption is clearly higher in the NR-Mesh in almost every injected traffic range. This is due to PC-Mesh provides separated networks that are able to power down themselves when not needed, while the NR-Mesh finds more difficult to power down entire components. Besides, Figure 4.13(e)(f) shows how the latency degrades as the injected traffic increases in the NR-Mesh, compared with the PC-Mesh topology.



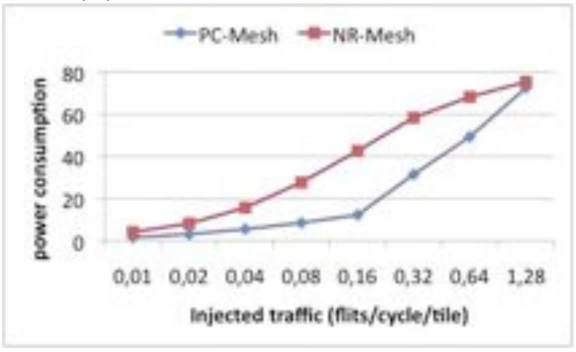
(a) 16-cores, accepted traffic



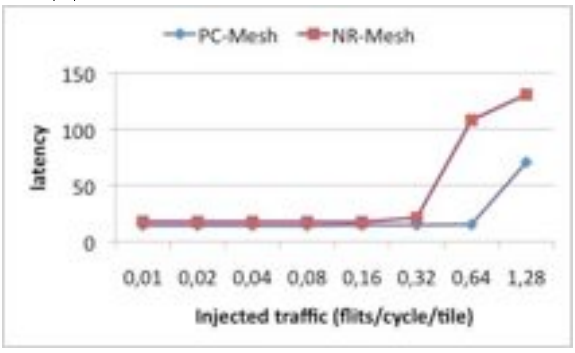
(b) 32-cores, accepted traffic



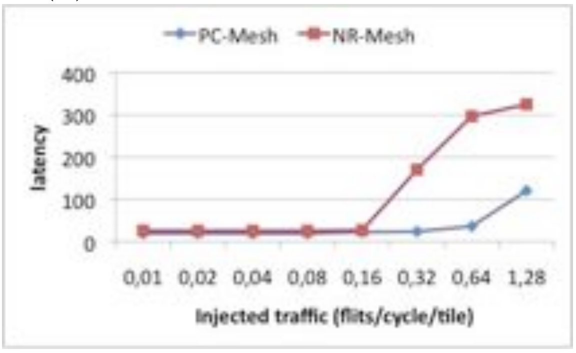
(c) 16-cores, power consumption



(d) 32-cores, power consumption



(e) 16-cores, latency

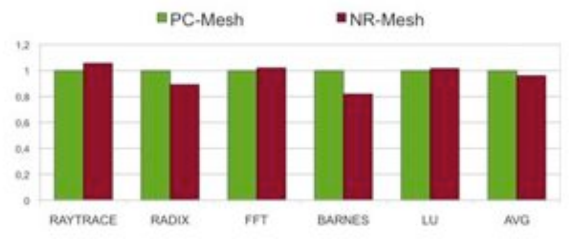


(f) 32-cores, latency

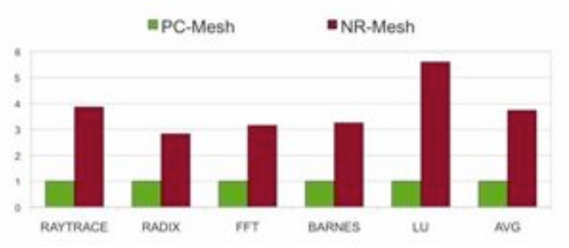
Figure 4.13: Accepted traffic, power consumption and latency comparison between NR- and PC-Mesh topologies using uniform synthetic traffic.

4.5.2 Real Applications

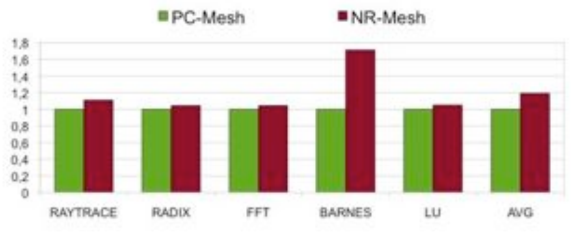
Figure 4.14 shows a similar reasoning in real applications as in the previous section. We can see that the execution time is very similar in both topologies due to the low traffic in the SPLASH-2 applications. However, the energy consumption is much higher in the NR-Mesh topology. As we saw in the previous section, the power consumption is much lower in almost all traffic ranges



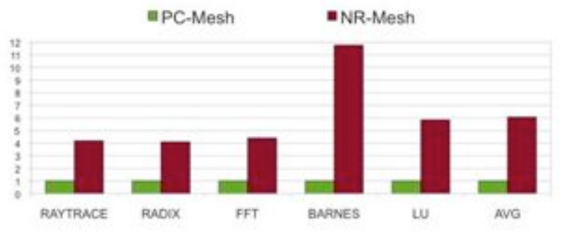
(a) 16-cores, execution time



(b) 16-cores, energy consumption



(c) 32-cores, execution time



(d) 32-cores, energy consumption

Figure 4.14: Normalized application execution time and normalized network energy consumption comparison between NR- and PC-Mesh topologies using real applications.

for the PC-Mesh topology. That is the reason why the energy consumption (average power consumption added every cycle during the application's execution) is much lower in the PC-Mesh network. Although the traffic is low in real applications, the power consumption is always higher in the NR-Mesh network topology.

4.6 Conclusions

We have proposed two alternative topologies, PC- and FTPC-Mesh, to address the increasing network power consumption in CMP systems. The proposed topologies rely on extending the connectivity of the nodes to different subnetworks. Parallel networks allow an efficient on/off mechanism to cooperatively work with an injection algorithm at the network interface. Switches are powered down in a distributed manner and subnetworks are used based on the traffic injection requirements. When using the concentrated mesh as a subnetwork, network latencies are kept low, when compared with the 2D-Mesh topology.

Experimental results with both synthetic traffic patterns and real applications, using 16- and 32-node system configurations demonstrated the high benefits of the new topologies, achieving large savings in network power consumption without increasing path lenghts nor execution time of applications.

As a result of the comparison between NR-Mesh and PC-Mesh topologies, in the next chapter we continue studying the PC-Mesh one exploiting its flexibility, improving some additional properties, as its fault tolerance, but without adding extra resources (switches/links) as we performed in the FTPC-Mesh topology to increase the fault tolerance.

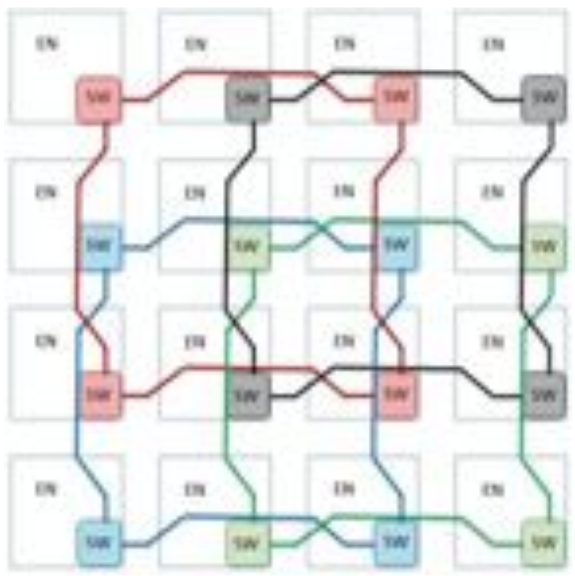
Chapter 5

HPC-Mesh Topology

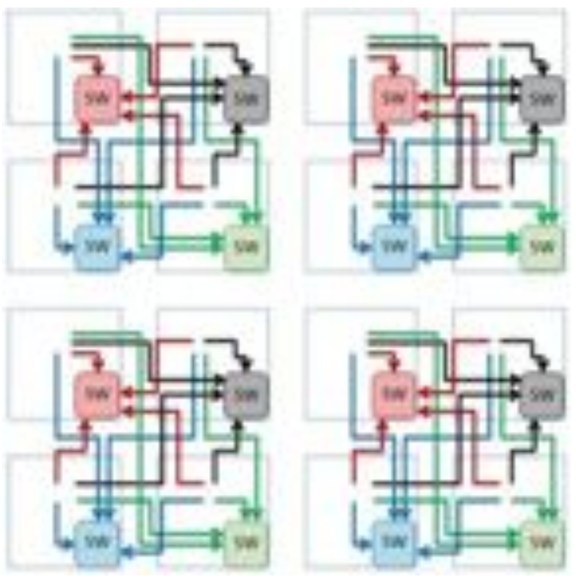
In the previous chapters we have proposed the NR-Mesh topology and the PC-Mesh topology. Also, we showed the superior performance of PC-Mesh when compared with the NR-Mesh. The benefit comes from the fact of using parallel networks and having disjoint paths through those networks.

Now, we propose a new topology for NoCs that, similar to PC-Mesh, it offers reduced hop count latencies as the C-Mesh network and enables the use of alternative paths when necessary but improving the fault tolerance of the PC-Mesh topology . The new topology will be homogeneous, all the nodes are connected to the same number of switches, in contrast to the PC-Mesh that can be considered a non-homogeneous topology. Therefore, the new topology will have the same fault tolerance degree as the FTPC-Mesh, but having the same number of switches and nodes like in PC-Mesh.

As done in previous chapters, we show an injection algorithm that allows every end node to decide the best subnetwork to use in order to achieve low end-to-end latencies and minimized power consumption values. We explain results extracted from detailed simulations including all the possibilities when faulty components are present. Also, we show a possible implementation for the new topology in a 3D structure with a relative minimal effort, that is, without adding complex routing algorithms or a prohibitive number of extra resources.



(a) Connection pattern between switches (external links)



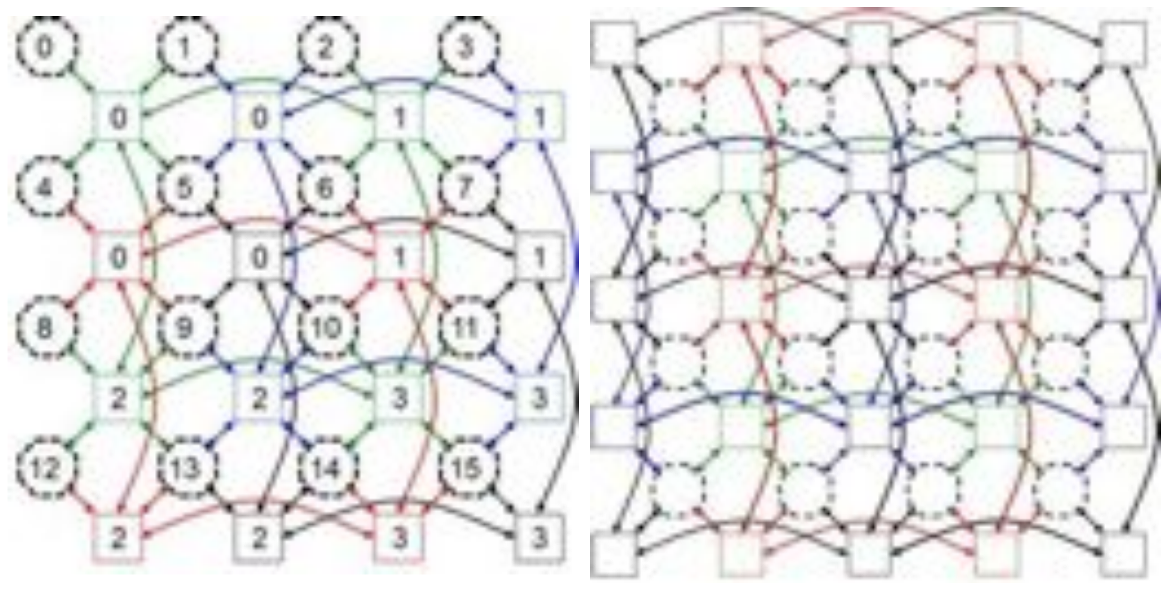
(b) Connection pattern between both switches and end nodes (internal links)

Figure 5.1: HPC-Mesh topology.

5.1 HPC-Mesh

Figure 5.1 shows the HPC-Mesh topology: (a) the connection pattern between switches in the same subnetwork or concentrated mesh and (b) the connection pattern between switches and end nodes. Every end node is connected to four switches each one belonging to a disjoint network. When focusing only on switches, they build four disjoint networks, thus, every switch belongs only to a single network. In our example, four 2×2 subnetworks are build (each with a different color). Notice that switches of different networks are not connected between them. As we plan to switch off entire subnetworks, they should be isolated from the others in order to get an efficient power management (as was done for the PC-Mesh).

When focusing on end nodes, Figure 5.1.(b), we can see that they have the same connectivity pattern. Every node is connected to 4 switches, each one in a different subnetwork. The four end nodes located in the north-west quadrant of the chip are connected to the four switches of the same quadrant. The rest of end nodes follow the same approach in their quadrant. As an end node is connected to four switches, the logic for the injection of packets



(a) PC-Mesh (b) Fault-tolerant PC-Mesh

Figure 5.2: PC-Mesh topology.

will implement an algorithm that selects the switch (subnetwork) to use on a per packet basis (described later). The injection algorithm will have a similar overhead of the previous one for PC-Mesh topology, thus, having a negligible impact.

The PC-Mesh topology is shown again in Figure 5.2 for comparison purposes. The major problem of the previous topology is the connectivity pattern having several nodes connected to less than 4 subnetworks and decreasing fault tolerance. For example, the worst case scenario is when the top left-most switch fails. The entire network will be inoperable because of the first end node is only attached to the first switch.

To provide higher fault tolerance degree, the PC-Mesh topology requires an additional row and column of switches. In addition to this extra overhead, the HPC-Mesh topology solves this problem by a smart connection pattern between end nodes and switches (Figure 5.1).

An overview of the overall topology is shown in Figure 5.3.

5.1.1 Tile-Based Design

One important aspect in CMP systems is to use a tile-based design as it significantly reduces the design effort. Thus, the proposed HPC-Mesh topology

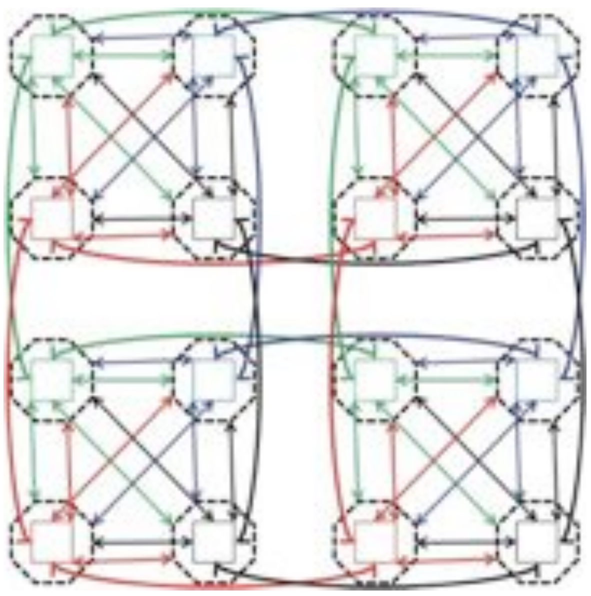


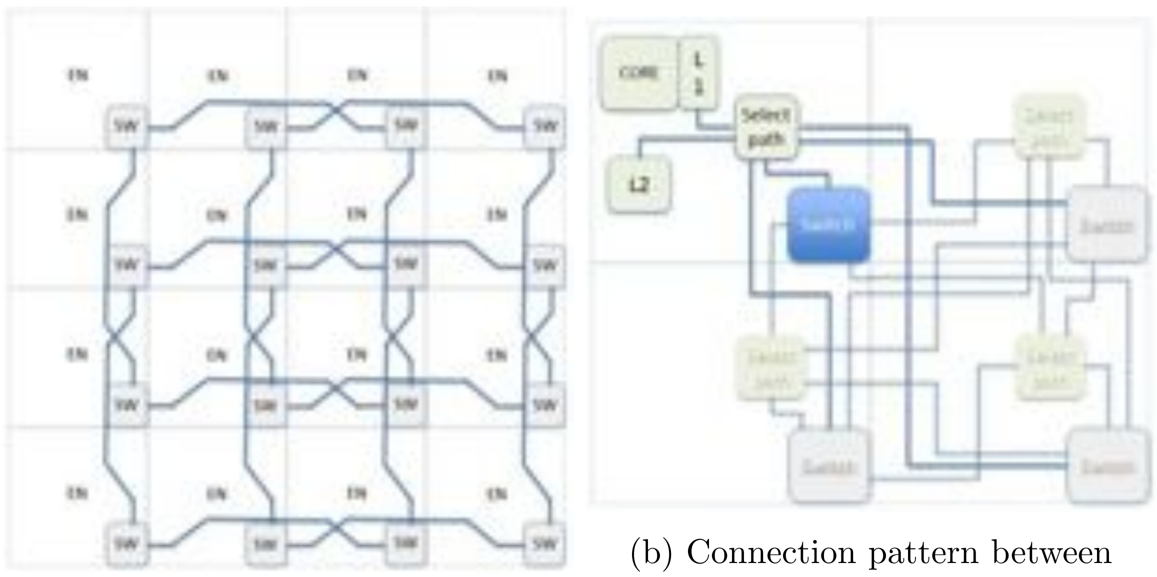
Figure 5.3: HPC-Mesh overview. Octagons are the end nodes and squares the switches.

must be adapted to a tile-based structure. Figure 5.4 shows a possible tile-design for the HPC-Mesh taking into account both the external links (Figure 5.4.(a)) and the internal links (Figure 5.4.(b)). As can be seen, tiles exhibit an homogeneous design for both the external and internal links. Upper metal layers must be used to route properly those links. External links are homogeneous (have the same layout pattern) at tile level. However, internal links are homogeneous (have the same layout pattern) at 2×2 tile level. Anyway, we can use the mirror effect for building neighbour tiles.

5.1.2 Injection Algorithm

In this section we describe the injection algorithm assuming a 4×4 HPC-Mesh topology. The goal of the algorithm is to adapt the use of subnetworks to the current injection load of the end node. The four subnetworks are checked in an ordered way and the algorithm is used on a per packet basis.

As XY routing is assumed and an end node now can be reached through four subnetworks, we need to compute the final switch destination through each possible subnetwork. However, as can be seen in Figure 5.1, each end node is attached to four switches with the same location in their subnetwork.



(a) Connection pattern between switches (external links)

(b) Connection pattern between switches and end nodes (internal links)

Figure 5.4: HPC-Mesh topology tile design.

For instance, the top left-most node is attached to switches located at the first row and column of its subnetwork. This is different from the PC-Mesh network where switch IDs to reach the same end node were different at each subnetwork.

In the algorithm, the number of enqueued flits F_x for each subnetwork is checked, that is, for every injection port at every network interface. By default, the subnetwork with the lowest number of enqueued flits is taken. However, to save power in low traffic conditions a threshold value is used (T_h). Several thresholds have been tested, and the chosen value was 200 flits because it presented the best trade-off between power and performance. It means that the four networks can only be used when the flits enqueued in the first subnetwork (F_0) overcomes the T_h value. In this situation, we open the remaining subnetworks and use them in an ordered way (we analyzed the effects of gradually opening every subnetwork obtaining worse results). Otherwise, when $F_0 < T_h$, the chosen subnetwork is the first one and the rest of the networks are powered off (once all the enqueued flits are injected, that is, F_x becomes 0). Figure 5.5 shows the injection algorithm.

With the designed algorithm, the HPC-Mesh network will allow most of

```
function injection-algorithm($F_0$, $F_1$, $F_2$, $F_3$) : subnetwork
var   s: subnetwork
var   th: threshold
begin
  s = 0
  if ($F_0$ >= th) then
     if ($F_1 <= F_0$) then s = 1
     if ($F_2 <= F_1$) then s = 2
     if ($F_3 <= F_2$) then s = 3
     else s = 0
  return s
end function
```

Figure 5.5: Injection algorithm for the HPC-Mesh topology.

the subnetworks to be switched off as the traffic will be directed to only the first subnetwork, thus resembling a C-Mesh network for low traffic minimizing the power consumption. However, for high traffic rates the HPC-Mesh network will use the four subnetworks by prioritizing queues with less number of flits, thus, maximizing performance and power.

5.2 Fault tolerance

One of the key properties of the HPC-Mesh network is its fault tolerance degree. Providing parallel networks increases significantly the reliability of the entire network. In this section we analyze this property by computing the probability that one or more failed component (switches and links) leave end nodes unconnected. This study assumes the faulty switches are identified during the initialization phase. We provide the analysis for 2D-Mesh, C-Mesh, PC-Mesh and HPC-Mesh network topologies and assume the use of the DOR routing algorithm.

5.2.1 Faulty subnetworks

Table 5.1 shows the fault tolerance for the 2D/C-Mesh networks and for the PC/HPC-Mesh subnetworks. As can be seen, the 2D-Mesh and C-Mesh do not support any failure. Otherwise, the PC-Mesh does not support a failure in the first subnetwork. In contrast, the HPC-Mesh support failures in up to 3 subnetworks. Notice that any subnetwork could perform as network 0 because of the homogeneous design of the HPC-Mesh.

All the faulty combinations where subnetwork 0 is affected means PC-Mesh will be impacted. On the contrary, the homogeneous design of HPC-Mesh allows the system to keep connectivity. For 2D-Mesh and C-Mesh cases, as there is no any redundancy in each subnetwork, the case for one failure on each subnetwork leads to every configuration to fail.

FNS	0	1	2	3	0-1	0-2	0-3
2D-M	N	N/A	N/A	N/A	N/A	N/A	N/A
C-M	N	N/A	N/A	N/A	N/A	N/A	N/A
PC-M	N	Y	Y	Y	N	N	N
HPC-M	Y	Y	Y	Y	Y	Y	Y
FNS	1-2	1-3	2-3	0-1-2	0-2-3	1-2-3	0-1-2-3
2D-M	N/A	N/A	N/A	N/A	N/A	N/A	N/A
C-M	N/A	N/A	N/A	N/A	N/A	N/A	N/A
PC-M	Y	Y	Y	N	N	Y	N
HPC-M	Y	Y	Y	Y	Y	Y	N

Table 5.1: Faulty Network Support. The used acronyms in the table are FNS (Faulty Network Support) which shows the failed networks separated by a dash. For instance, 1-2 means they are failed components in subnetworks 1 and 2. N/A means Not Applicable (never supported) and Y (supported failures) or N (unsupported failures).

5.2.2 Faulty switches

As done in the previous chapter, the hyper-geometric distribution has been used to obtain the fault tolerance for both PC-Mesh and HPC-Mesh topologies. Figure 5.6 shows the obtained results for both versions: PC-Mesh and HPC-Mesh using 16 and 32 cores, respectively. As shown in the figure, the HPC-Mesh topology is able to tolerate up to 3 failures as it has four complete

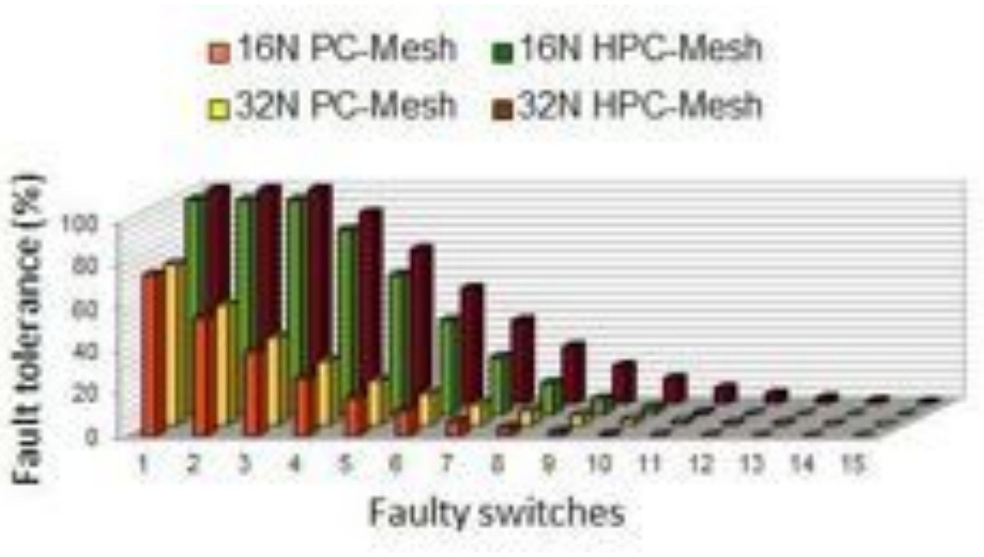


Figure 5.6: Fault tolerance comparison for PC-Mesh and HPC-Mesh using 16 and 32 nodes.

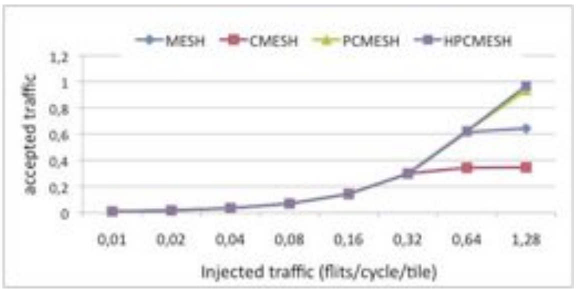
disjoint networks.

However, the PC-Mesh topology obtains a degree of 75% of fault tolerance for a single switch failure. Indeed, having one end node connected to a single switch prevents achieving 100% fault tolerance coverage for the one-switch failure case, although when compared with the 2D-Mesh and C-Mesh topologies results are also good, since these topologies are not able to tolerate, assuming the DOR routing algorithm, a single switch failure in any case.

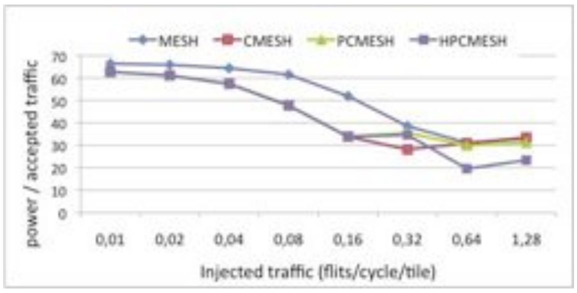
Figure 5.6 shows the graceful degradation for the fault tolerance in the PC-Mesh and the HPC-Mesh topologies. The results (as explained before) demonstrate how the HPC-Mesh topology achieves a higher degree of fault tolerance than the PC-Mesh one.

5.3 Performance Evaluation

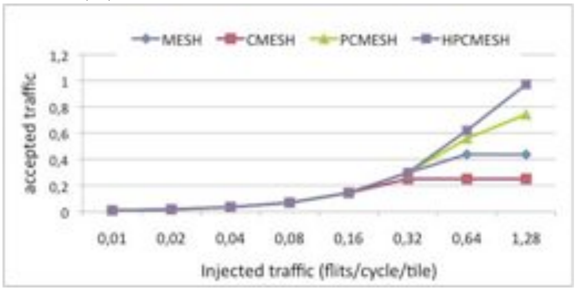
In this section we evaluate the new topology. We compare the HPC-Mesh topology with the 2D-Mesh, C-Mesh and PC-Mesh ones. DOR is used in all of them with the same number of Virtual Channels (VCs), only one in this case. Virtual Networks (VNs) are used to avoid protocol-level request-reply deadlocks induced by the cache coherency protocols (for the scenarios evaluated with applications). The same scenarios of the previous chapter are



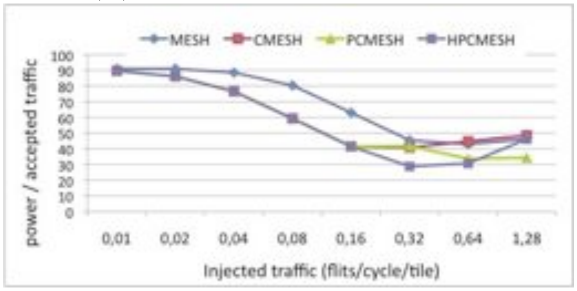
(a) accepted traffic, uniform



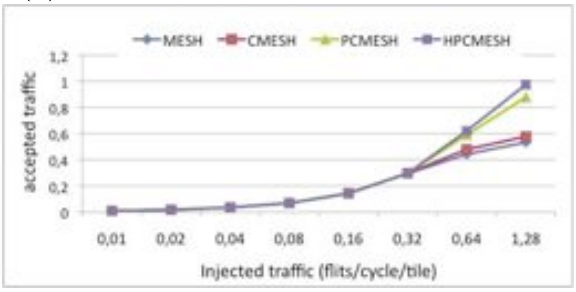
(b) power per flit, uniform



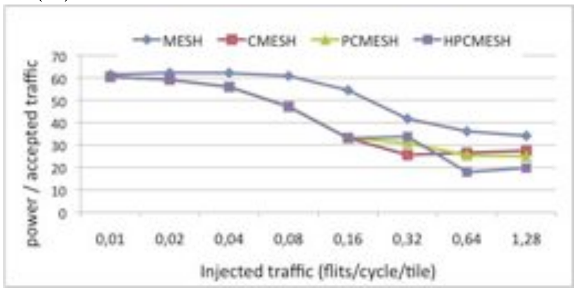
(c) accepted traffic, bit-complement



(d) power per flit, bit-complement



(e) accepted traffic, bit-reversal



(f) power per flit, bit-reversal

Figure 5.7: 16-node synthetic traffic comparison (accepted traffic measured as flits/cycle/tile).

used, please refer to that part for details of each scenario.

5.3.1 Synthetic Traffic Results

Figure 5.7 shows the performance and power results for a 16-tile system (4×4 topology) under synthetic traffic patterns (uniform, bit-reversal, and bit-complement). Figures a, c, and e show accepted traffic and figures b, d, and f show the power consumption per flit unit (namely, overall power consumed divided by accepted traffic).

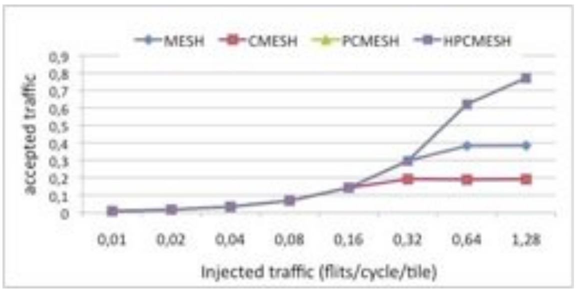
First thing to note is the higher throughput for both HPC-Mesh and PC-Mesh, in the three scenarios. The HPC-Mesh and the PC-Mesh topologies exhibit shorter paths and provide parallel paths not available in the 2D-Mesh network, thus lowering contention in high traffic conditions. Besides, the HPC-Mesh network throughput is higher than the PC-Mesh for high traffic rates as all the end nodes have four disjoint subnetworks, not provided in the PC-Mesh. On the other hand, the limited bisection bandwidth of the C-Mesh topology limits its maximum throughput, low below the 2D-Mesh network (except for the bit-reversal traffic where is almost equal).

Now, looking into the power consumption figures, we can see how all the topologies (except the 2D-Mesh which performs worse) exhibit the same power consumption rate per accepted traffic unit in low traffic conditions. When the traffic increases, the HPC-Mesh usually performs better than the other topologies. It is worth mentioning that this does not mean all the topologies deliver the same amount of traffic (as seen in the previous figures). However, at all rates the HPC-Mesh usually helps to obtain good results in power consumption.

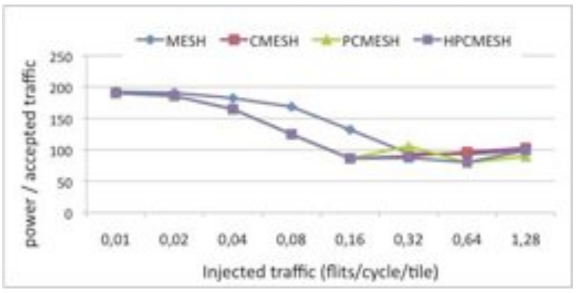
Figures 5.8 show the same results but now for 32-node systems (4×8 topologies). Both HPC-Mesh and PC-Mesh throughput is increased for the different traffic patterns when compared with the 2D-Mesh and C-Mesh topology. Also, as in the 16-node case, the power per flit unit delivered in the HPC-Mesh topology remains low when compared to the 2D-Mesh and similar with the other topologies.

5.3.2 Real Application Results

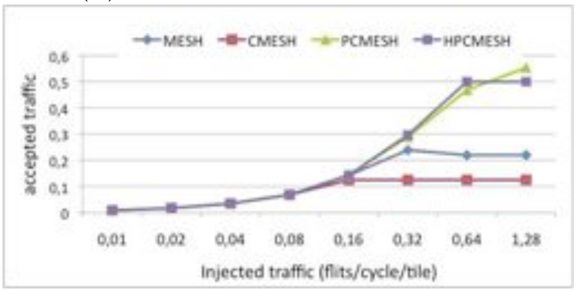
Now we evaluate application's execution time, and energy savings, when real applications are run. Figures 5.9 and 5.10 show normalized execution time and network energy consumption for different applications when using different topologies for 16- and 32-Node system configurations, respectively. Results are normalized to the 2D-Mesh case. We can see in either case how the execution time is reduced when using the C-Mesh, PC-Mesh and HPC-Mesh. This is mainly due to the reduction in the path length. Another benefit comes when looking into the network energy consumption values. Here we can also notice a significant reduction again when using C-Mesh, PC-Mesh and HPC-Mesh. These results demonstrate the low traffic conditions in the SPLASH-2 applications leading the PC-Mesh and HPC-Mesh topologies to perform as the



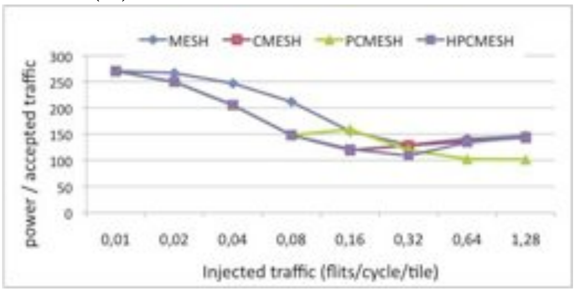
(a) accepted traffic, uniform



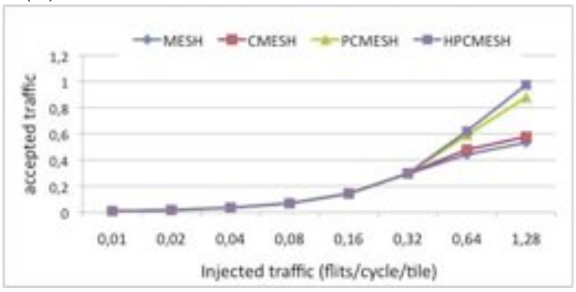
(b) power per flit, uniform



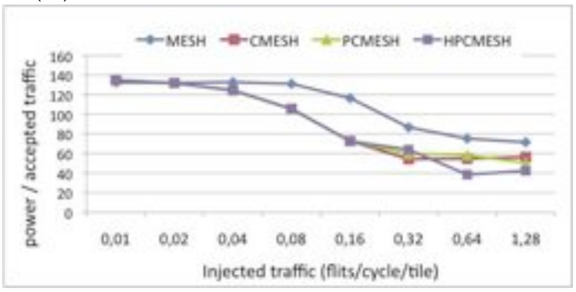
(c) accepted traffic, bit-complement



(d) power per flit, bit-complement

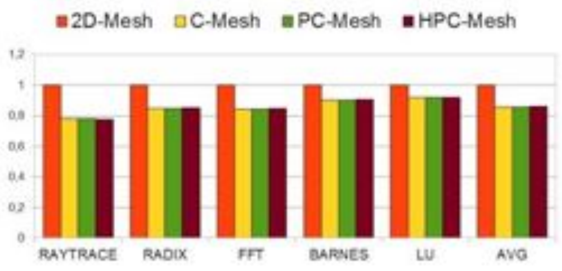


(e) accepted traffic, bit-reversal

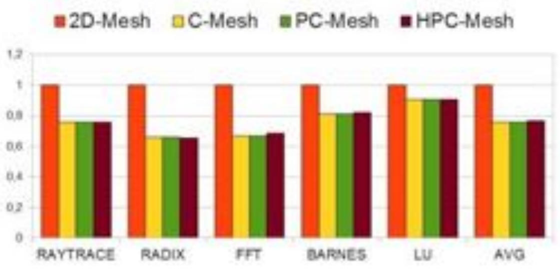


(f) power per flit, bit-reversal

Figure 5.8: 32-node synthetic traffic comparison (accepted traffic measured as flits/cycle/tile).



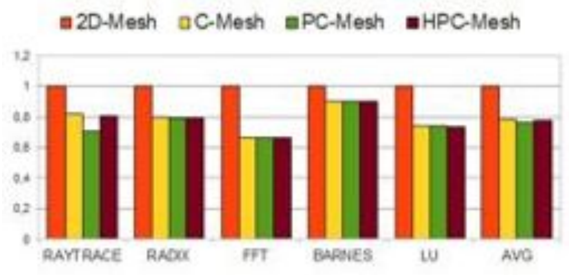
(a) 16-cores, execution time



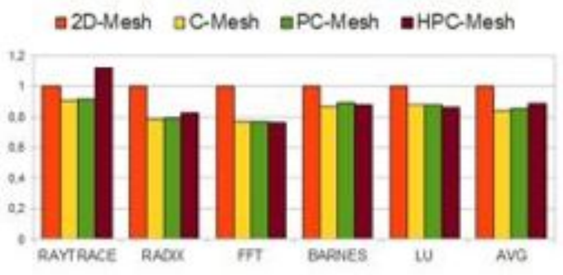
(b) 16-cores, energy consumption

Figure 5.9: 16-node execution time and network energy consumption comparison.

C-Mesh topology almost all the time, on average.



(a) 32-cores, execution time



(b) 32-cores, energy consumption

Figure 5.10: 32-node execution time and network energy consumption comparison.

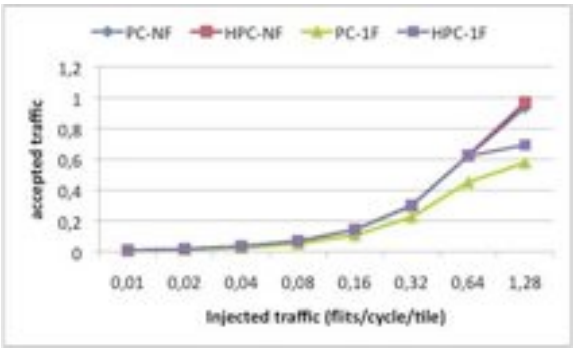
Figure 5.10 shows results for a 32-Node system, where the same trends are achieved, thus exhibiting good scalability values.

Thus, the main conclusion from this evaluation, and taking as a reference the same evaluation performed in the previous chapter, is that the HPC-Mesh is able to behave as the PC-Mesh, at least, in low traffic conditions. Therefore, from performance and power point of view, there are no significant differences and gains. Notice that the aim of the HPC-Mesh topology is to enhance fault tolerance of the PC-Mesh topology and not enhancing its performance.

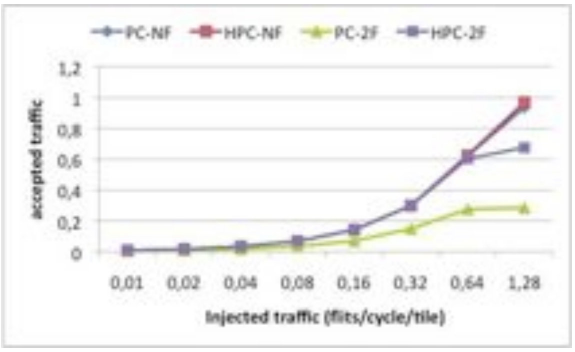
5.3.3 Performance Under Faulty Networks

As fault tolerance is the driving factor of HPC-Mesh, in this section we evaluate its performance in the presence of network failures. We evaluate both PC-Mesh and HPC-Mesh (PC and HPC in the figures). NF in the figures means when there is no failure at all, and xF means when x subnetworks have failed. A subnetwork fails when one of its components fails, e.g., a switch or a link. We only compare the PC-Mesh and HPC-Mesh because of 2D-Mesh and C-Mesh do not support any component failure combination.

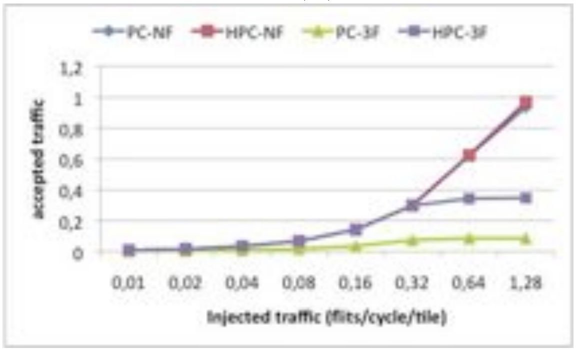
To perform a fair comparison between both topologies, we take into account all the possible failures. Notice that in the PC-Mesh, the first subnetwork can fail and the accepted traffic in this case is 0, because the node 0 is dependent to this network and it only has one injection link going to this subnetwork. For either 1 faulty network, 2 faulty networks and 3 faulty networks we obtain the average accepted traffic taking into account all possible failures, that is,



(a) accepted traffic, 1 network failure



(b) accepted traffic, 2 networks failure



(c) accepted traffic, 3 networks failure

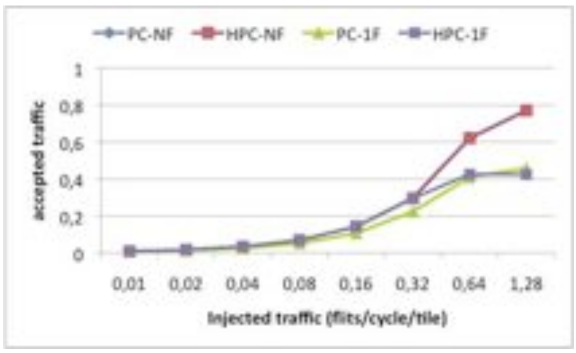
Figure 5.11: 16-node fault tolerance support. HPC-Mesh versus PC-Mesh (accepted traffic measured as flits/cycle/tile).

we average the accepted traffic when one subnetwork fails (0, 1, 2 or 3), 2 subnetworks fail (0-1, 0-2, 0-3, 1-2, 1-3 or 2-3) and 3 subnetworks fail (0-1-2, 1-2-3 or 1-2-4). Obviously, there is nothing to do when the 4 subnetworks fail.

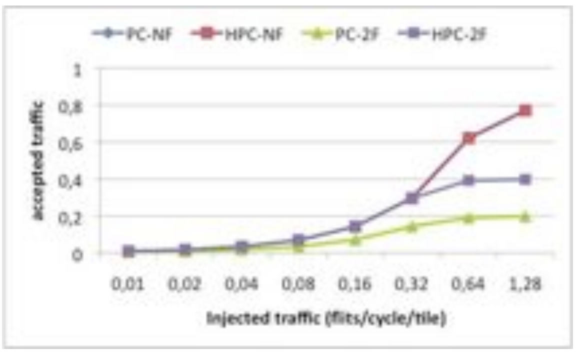
Figure 5.11 shows performance results when different number of subnetworks have failed. The first thing to notice is the higher performance of HPC-Mesh when compared to PC-Mesh. This is noticeable when several subnetworks have failed. For the 0F case no differences are present.

Also, we can notice that 1F and 2F cases do not affect performance of HPC-Mesh until the injected traffic is high. This does not happen with PC-Mesh as one failure in the first subnetwork makes the topology unpractical.

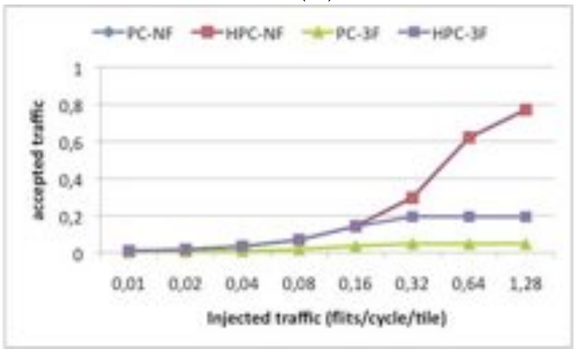
For the 3F case, although the HPC-Mesh performance gets more degraded, it still delivers acceptable throughput numbers, much higher than the ones achieved by PC-Mesh topology. Notice that the HPC-Mesh topology supports a failure in the first network (see table 5.1) in contrast with the PC-Mesh one. Figure 5.12 shows the results with faulty components when simulating 32-node



(a) accepted traffic, 1 network failure



(b) accepted traffic, 2 networks failure



(c) accepted traffic, 3 networks failure

Figure 5.12: 32-node fault tolerance support. HPC-Mesh versus PC-Mesh (accepted traffic measured as flits/cycle/tile).

systems. Although the accepted traffic is lower for both topologies, the HPC-Mesh topology still achieves very good results compared with the PC-Mesh topology.

To conclude this section, it is worthy to say that very good results have been obtained in terms of fault tolerance with the HPC-Mesh in contrast with other topologies, and all of this avoiding using complex routing algorithms.

5.4 Towards a 3D mesh structure

As a final effort in this chapter, and due to the homogeneity provided by the HPC-Mesh topology, now we propose and discuss how the HPC-Mesh network can be extended to a 3D stacking scenario. By providing a new dimension, end-to-end latency and power consumption can be further reduced. The trend is to implement the additional links in the vertical dimension by using through silicon vias (TSVs), although there are several problems to consider (manufacturing costs and thermal effects).

Figure 5.13 shows the structure for a 3D HPC-Mesh implementation (HPC-3DMesh). As can be seen the vertical links, implemented as TSVs, are a quarter of the switches used in the 2D mesh plane (see also [20] and [41]). TSVs are only used for the first subnetwork. In contrast of other solutions, a complex routing algorithm is not needed. We just need to know whether the destination is in another layer, then the dimension order routing algorithm (Z-X-Y) is used through subnetwork 0. Using this implementation we simplify the network design and we reduce its manufacturing costs.

The routing algorithm can be implemented at the network interface. By simply comparing the Z coordinate of the source node and the destination node, the network interface can know whether the first subnetwork or any other one can be used. In case both nodes are placed in different planes (Z components differ) then subnetwork 0 is used. If not, the usual power-aware injection algorithm of HPC-Mesh is used. Notice that subnetwork 0 is never switched off, thus packets crossing the Z dimension will always find the network available and ready for transmission.

We evaluate the performance using the HPC-Mesh as a 3D network on-chip. Although TSVs are faster than horizontal links, we take the conservative approach where vertical links still need two cycles (as the horizontal links). We leave the power consumption estimation for a future contribution. Notice that this section intends to show a preliminary implementation of a 3D HPC-Mesh version, which could help to understand the potential of the HPC-Mesh design in this new environment.

Figure 5.14 shows the performance comparison between 2D and 3D HPC-Mesh structures. Figure 5.14.(a) shows results for different Splash-2 real applications normalized to the 2D case (4x8 HPC-Mesh). By using a new dimension, execution time is reduced by 9% on average in the HPC-3DMesh topology. Although the improvement seems not to be as expected, this is due to the low traffic injected by the applications (we have observed that the overall link utilization in Splash-2 applications is usually around 10% on average in most cases). Looking at Figure 5.14.(b), we can see the same comparison when using synthetic traffic with a uniform distribution of message destinations. This time we compare the average latency for a packet through the network. Here,

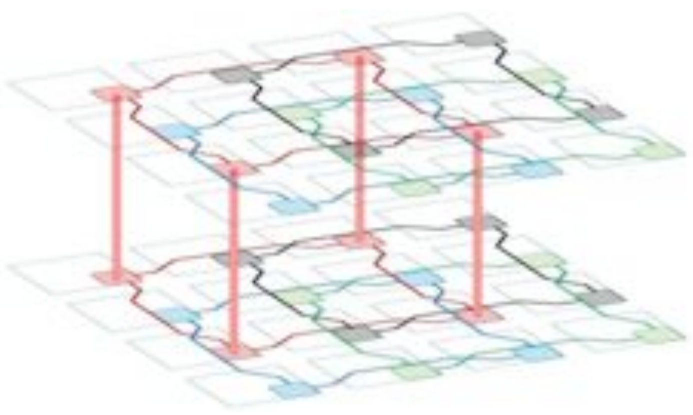


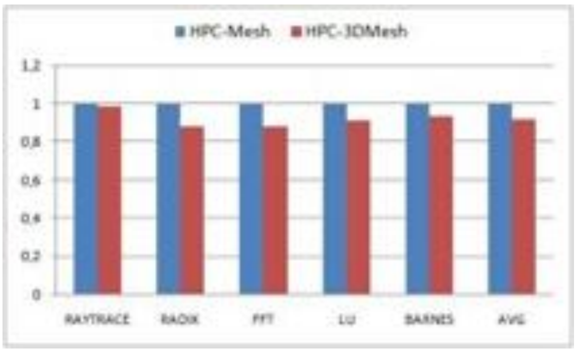
Figure 5.13: HPC-Mesh 3D structure.

we can see that at low traffic rates the latency does not vary much. However, when a moderate traffic arises the latency reduction in the HPC-3DMesh is very significant because the HPC-Mesh 2D has roughly double latency for moderate and high traffic rates. In Figure 5.14.(c) we run the same applications, but now we add synthetic traffic, specifically 0.06 flits/tile/cycle as a background component. Provided results are very promising for the HPC-3DMesh, achieving on average up to 57% reduction in execution time and, therefore, achieving large power consumption savings.

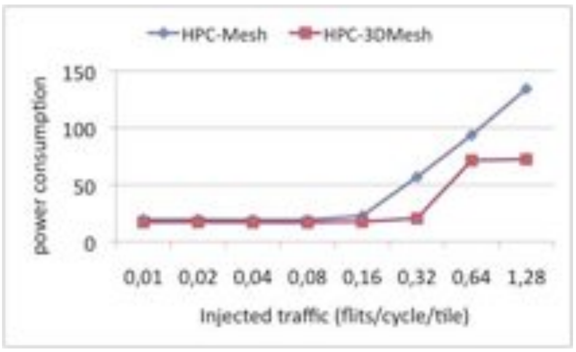
Then, as a conclusion we can state the HPC-3DMesh exhibits better performance than the HPC-Mesh using a limited number of vertical links (TSVs) and the most important, using a simple algorithm benefiting from the HPC-Mesh implementation just adding a quarter of possible TSVs.

5.5 Conclusions

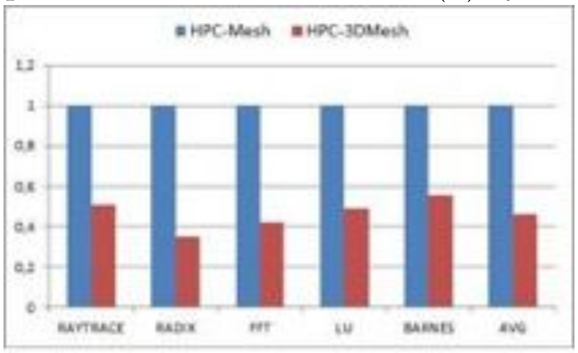
We have proposed in this chapter an homogeneous parallel concentrated network (HPC-Mesh) which uses a tiny injection algorithm to inject packets through four alternative concentrated networks. We study the impact of the HPC-Mesh when compared to other topologies. This topology exhibits a good trade-off between performance, power consumption and fault tolerance support, using an intelligent injection algorithm capable of managing every concentrated subnetwork dynamically. The HPC-Mesh supports a high degree of fault tolerance in contrast with the other studied topologies, and using a



(a) real applications



(b) synthetic traffic



(c) real applications + synthetic traffic

Figure 5.14: HPC-Mesh versus HPC-3DMesh.

simple routing algorithm (Dimension Order Routing).

Also, the HPC-Mesh can perform as a 3D structure with a reasonable cost by using a quarter of vertical TSV links and, therefore, reducing thermal effects.

Although we provide notable improvements compared with the PC-Mesh topology, we can expect an increase in latency when high traffic arises because of the connection pattern. As an example, the central nodes do not have a direct connection between them (See Figure 5.3). To overcome this problem, in the next chapter we design an hybrid topology between PC-Mesh and HPC-Mesh, the HNPC-Mesh topology, using the best properties from each topology at every time.

Besides, in the last chapter with also compare our contributions with the 2D-Mesh with express channels (a *virtual express channels* approach is provided in [19]).

Chapter 6

HNPC-Mesh Topology

In this final chapter we propose the final topology, HNPC-Mesh, which includes the best properties from the previous two topology proposals, PC-Mesh and HPC-Mesh. Also, we compare the proposed variants of the PC-Mesh and C-Mesh topologies against the 2D-Mesh with express channels, taking the 2D-Mesh network on-chip topology as a reference. First of all, we compare the execution time and energy consumption when using real applications. Later, we compare the same parameters but adding background synthetic traffic.

6.1 HNPC-Mesh

The HNPC-Mesh topology (Homogeneous/Non homogeneous Parallel Concentrated Mesh) is a hybrid design between the PC-Mesh and HPC-Mesh topology. The purpose of this implementation is clear: provide fault tolerance and the maximum possible performance at the same time.

One drawback in the PC-Mesh topology is the lack of support for fault-tolerance. However, the PC-Mesh achieves the best performance among all the topologies studied in this thesis. On the other hand, the HPC-Mesh topology is able to provide full fault tolerance support without adding extra resources. Although this last topology overcomes the PC-Mesh, when high traffic arises, the HPC-Mesh can provide worse results in some cases. As we will see later, the HPC-Mesh provides higher execution time than the PC-Mesh when high traffic arises. This is mainly due to the connection pattern, because, as an example, the central nodes need 3 hops to connect between them instead of one as in the PC-Mesh topology.

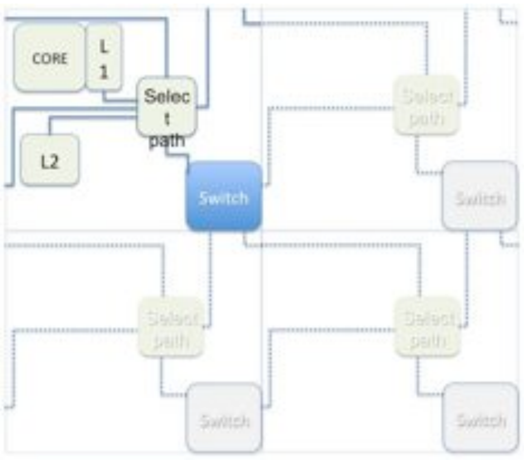
To prevent the HPC-Mesh from providing lower perfomance when compared to the PC-Mesh, we extend the HPC-Mesh topology and add the internal links used in the PC-Mesh topology. The injection algorithm includes a multiplexor which will decide whether the PC-Mesh or HPC-Mesh topology is used, choosing the injection algorithm described for each one in the last chapters. Therefore, the new topology, HNPC-Mesh should be seen as an overlapping of both previous topologies, where the injection links of both topologies can be used and are selected at the injection. The final topology will behave as the PC-Mesh or as the HPC-Mesh depending on the traffic and on the presence of failures.

6.2 Tile Based Design

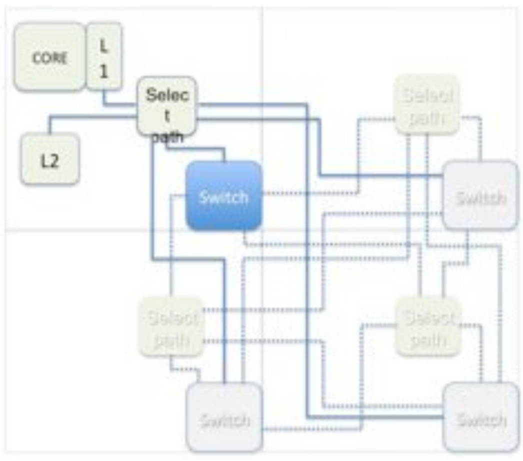
Figure 6.1(a)(b)(c) shows the tile based design for internal links (those connecting the end nodes to the switches) for the PC-Mesh, HPC-Mesh, and the new topology, the HNPC-Mesh topology. Figure 6.1(d) shows the connection pattern in a tile base design for the external links, which is the same for all the topologies.

Basically, the HNPC-Mesh merges the internal links of both PC-Mesh and HPC-Mesh topologies. Because of the negligible impact of the selection function, the idea is to use the PC-Mesh injection algorithm and, in case a switch fails, changing to the HPC-Mesh selection function. As only one of the selection functions is working, the internal links not used are powered off.

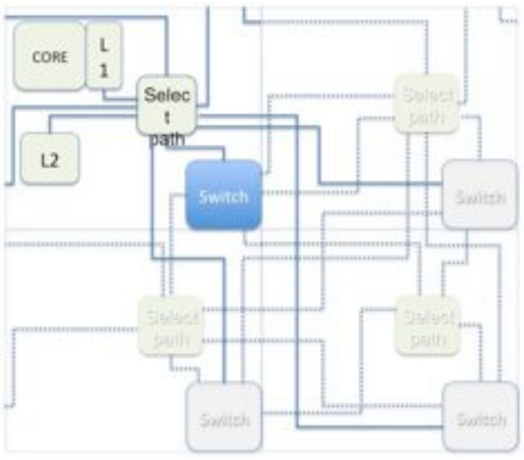
In this way we provide lower message latencies and fault tolerance at the same time. Notice that we add only seven internal links per tile, one to connect to the local switch in the tile, three to connect to the switches following the PC-Mesh pattern, and three to connect to the switches following the HPC-Mesh pattern. Notice that three of such links will always be powered off, one set or the other depending if there is a permanent failure or not and depending on the traffic load.



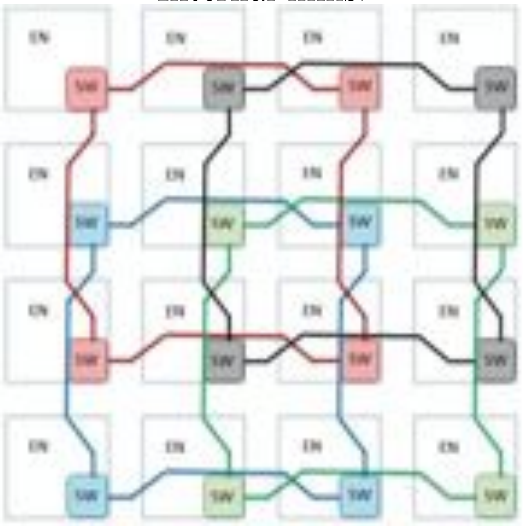
(a) PC-Mesh tile based design. Internal links.



(b) HPC-Mesh tile based design. Internal links.



(c) HNPC-Mesh tile based design. Internal links.



(d) PC/HPC/HNPC-Mesh. External links.

Figure 6.1: PC-Mesh, HPC-Mesh and HNPC-Mesh tile based design.

6.3 Injection algorithm

As mentioned before, the logic is in charge of choosing between the PC- or HPC-Mesh only enabling the proper internal links. By default, the PC-Mesh is enabled because of its performance properties. However, when a component fails, the logic enables the internal links for the HPC-Mesh topology and disables the links for the PC-Mesh network. See Figure 6.2.

In the next section we perform a further evaluation of the different topologies. In that evaluation we will identify the shortcoming of the HPC-Mesh

```
function HNPC_injection-algorithm()
   if (!fail) then
      switch off HPC-Mesh internal links
      switch on PC-Mesh internal links
      PC-Mesh_injection-algorithm()
    else
      switch off PC-Mesh internal links
      switch on HPC-Mesh internal links
      HPC-Mesh_injection-algorithm()
    end if
end function
```

Figure 6.2: Injection algorithm for the HNPC-Mesh topology.

network in terms of performance and, thus, the need of the HNPC-Mesh design. Notice that in the evaluation performed, the HNPC-Mesh network would work as the PC-Mesh, thus achieving the maximum performance but keeping the fault-tolerance properties of HPC-Mesh.

6.4 Performance Analysis

In addition to further compare PC-Mesh and HPC-Mesh, in this section we also compare with the 2D-Mesh network with express channels (we term this topology EC-Mesh). The express channels are added to the 2D-Mesh as shown in Figure 6.3 (blue color), every two switches. In particular, every switch is now connected to 2-hop neighbors along each direction and dimension. The algorithm used is DOR (as the rest of studied topologies in this chapter), prioritizing the use of long channels when possible. That is, when a message is headed to a destination that is more than one hop away trough a direction, then the express channel is selected as candidate for routing. On the contrary, if the message destination is one hop away, then the normal channel is selected as candidate. Thus, one-hop channels are used for local traffic and express channels are used for non-local traffic.

In this topology we can also use the power management logic described

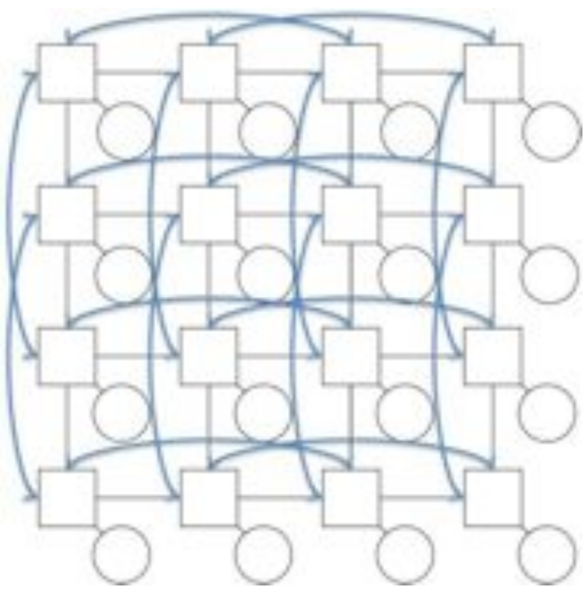


Figure 6.3: 2D-Mesh topology with express channels.

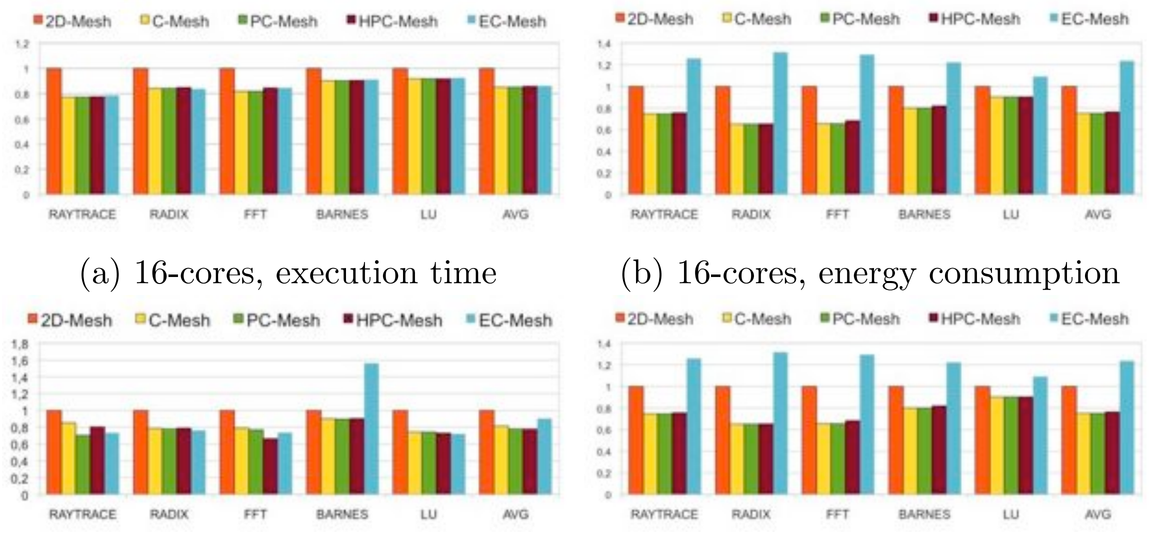
in previous chapters. Indeed, we do the evaluation by assuming that logic on every switch.

6.4.1 Real Applications

Parameters of the simulations are the same as of previous chapters. Indeed, we evaluate 16- and 32-core systems, for the different topologies analyzed so far: 2D-Mesh (as a baseline), C-Mesh, PC-Mesh, HPC-Mesh, and the 2D-Mesh with express channels, which we refer to EC-Mesh.

Looking at the performance numbers in Figure 6.4, we can see how the use of the express channels is sufficient to reduce the execution time of applications to the levels achieved by the proposals in this thesis. Indeed, for 16-core systems, the execution time is almost identical. However, for 32-core systems performance is impacted and the use of express channels does not solve the execution time problem of the 2D-Mesh completely, mainly due to some applications, such as BARNES in this case.

However, when looking at the energy consumption plots, we can clearly see totally different numbers. The express channel approach consumes far more energy than the 2D-Mesh and, of course, than the proposals in this thesis. 40% more energy is wasted when using express channels. This extra energy



(a) 16-cores, execution time (b) 16-cores, energy consumption

(c) 32-cores, execution time (d) 32-cores, energy consumption

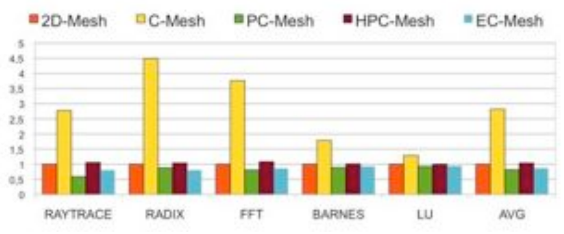
Figure 6.4: Application execution time and network energy consumption comparison. Results normalized to the 2D mesh case.

is mainly due to the extra buffers and extra links required at each port of the 2D-mesh with express channels. As high-radix switches are used, more buffers are needed.

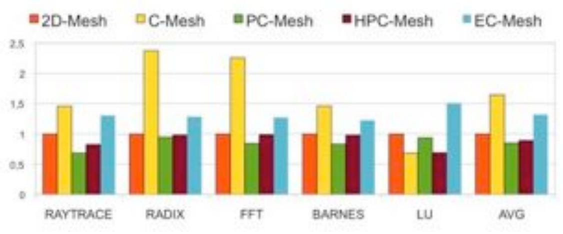
6.4.2 Real Applications with Background Traffic

Now (Figure 6.5) the network is loaded with additional synthetic background traffic, made of a uniform distribution of message destinations, and with an injection rate of 6% and 12% more messages for 16- and 32-node systems, respectively. As in previous chapters the C-Mesh topology is the one achieving the worst results when high traffic arises. The 2D-Mesh with express channels performs similar with background traffic with respect to the other topologies, except to the HPC-Mesh, when we can see how the execution time is larger than the one achieved even for the 2D-Mesh in both analyzed system sizes, 16-core and 32-core. Therefore, from a performance point of view, the HPC-Mesh network, in a overloaded configuration, presents higher execution times. This is mainly due to the higher latencies of this topology for certain pair of end nodes. For instance, the end nodes located in the center of the topology do not have a direct link connection between them, thus, hurting performance.

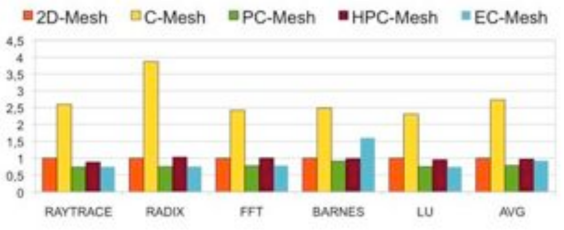
From the energy point of view, we see clearly that this effect does not



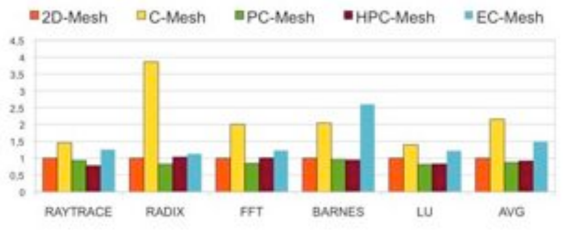
(a) 16-cores, execution time



(b) 16-cores, energy consumption



(c) 32-cores, execution time



(d) 32-cores, energy consumption

Figure 6.5: Application execution time and network energy consumption comparison with background traffic.

translate to higher energy consumption levels for the HPC-Mesh network. Indeed, we see comparable results as without background synthetic traffic. Therefore, in the previous examples, the HNPC-Mesh can be mapped into the PC-Mesh, and only it would map into the HPC-Mesh in presence of failures.

This is the motivational example that triggered us to design the HNPC-Mesh topology, the need to develop a high performance and power efficient on-chip network topology, but having a good degree of fault tolerance.

6.5 Conclusions

It is clear that the best solution here is to combine the PC-Mesh and the HPC-Mesh topologies to create the HNPC-Mesh network on-chip topology. As said before, the hybrid design allows to perform very fast without wasting a large amount of power consumption and, besides, is able to tolerate up to three failures in different subnetworks.

To summarize this chapter, we can see how the EC-Mesh topology performs better than the rest of the topologies, however, the power consumption in this topology is prohibitive and higher than the one dissipated by the HNPC-Mesh topology. Therefore, the best trade-off between performace, power consumption and fault tolerance is for the HNPC-Mesh topology. All these properties convert the Homogeneous/Non-homogeneous Parallel Concentrated Mesh in the best topology we have designed in this thesis.

Chapter 7

Conclusions

In this final chapter, we present the conclusions of this dissertation, the contributions to the research domain and a brief list of research directions that will be addressed in the future. Finally, we also expose the results in terms of industry internships and publications and other contributions derived from the work presented in this dissertation.

7.1 Conclusions

The following is a list of conclusions extracted from the current dissertation. These conclusions helped to obtain the contributions and scientific publications that are at the core of the document.

- Current high-performance multicore solutions pledge for tile-based designs. Tile-based design is gaining momentum for newer Chips Multiprocessor (CMPs) solutions. As the expected trend is to include more and more cores inside a chip, these solutions rely on networks-on-chip (NoCs) to handle all the communication traffic between cores.

- 2-Dimensional mesh and the proposed topologies are appealing for CMPs. Manufacturers prefer planar mesh topologies due to their simplicity for routing purposes and because they fit very well the chip layout. Although other topologies are also interesting, from the point of view of performance, design tools are not suitable (e.g. fat-tree topologies) or the tile-based design enforce an homogeneous and regular structure of the network.

- It is imperative to find NoC solutions that offer at the same time flexibility and high perfomace, lower power consumption, and high redundancy leading to fault tolerance support. These are critical key aspects in the NoC domain. 2-Dimensional meshes offer an excellent flexibility, but they suffer from poor scalability, affecting the performance and power consumption on the network on-chip. Our proposals are able to scale better than the 2-Dimensional meshes achieving much better results in terms of performance, power consumption and fault tolerance support.
- Future challenges in CMPs will demand dynamic mechanisms in the chip so to adapt to such challenges. Challenges identified in the document are (1) to improve the performance obtaining power consumption savings using mechanisms which are able to power on/off unnused components in the network, (2) the use of several injection links in each end node to be able to achieve the previous goal, (3) to decouple the network on-chip topologies in several smaller subnetworks to obtain more power saving opportunities when low traffic arises and improving the fault tolerance support, and (4) the implementation of a 3-Dimensional structure with a relative minimal effort.

The previous conclusions from the dissertation put the research performed in perspective so to obtain the intended goals. In the next section the specific contributions of the current dissertation are highlighted, and conclusions derived from the proposals are provided.

7.2 Contributions

The overall contribution of the dissertation is to design a network on-chip topology able to provide a good trade-off between performance and power consumption providing a high degree of fault tolerance at the same time.

This overall achievement has been obtained with a step by step procedure described next, improving every proposed topology at the same pace the dissertation has been progressing. Constant to all the proposals, every node has up to four injection ports through different switches and has up to four reception ports also through different switches. The contributions and derived conclusions are:

- NR-Mesh (Nearest neighboR Mesh) network on-chip topology. This topology improves the performance and power consumption by saving several hops, and by, powering off unnused components (using a power management logic), thus, avoiding congested situations. Besides, provides a good degree of fault tolerance.

- PC-Mesh (Parallel Concentrated Mesh) network on-chip topology. This topology provides four completely decoupled parallel concentrated networks. It is able to power on/off entire subnetworks depending on the traffic conditions with a new power management logic. The performance and power savings are improved respect to the NR-Mesh, and the PC-Mesh also improves the fault tolerance support.

- HPC-Mesh (Homogeneous Parallel Concentrated Mesh) network on-chip topology. This topology mainly improves the fault tolerance property with respect to the previous one. This is achieved at the same time it significantly reduces the number of switches than the FTPC-Mesh (Full tolerant PC-Mesh version) requires.

- HPC-Mesh provides a 3-Dimensional structure with a relative minimal effort, reducing the vertical links by 75% and, therefore, reducing the thermal effects due to the homogeneous positioning of the links.

- HNPC-Mesh (Homogeneous/Non-homogeneous Parallel Concentrated Mesh) network on-chip topology. This topology is a hybrid design between both PC-Mesh and HPC-Mesh. It is able to provide a good trade-off between performance and power consumption in high traffic conditions as the PC-Mesh network, while providing the same fault tolerance degree as the HPC-Mesh with no extra resources added.

- All the topologies have been evaluated and compared against a baseline design as the 2D-Mesh and C-Mesh topologies. Also, the best performant network has been compared against other variations like the 2D

Mesh with extra express links. In all the cases, results provided positive margins for the proposed topologies in this thesis.

- All the topologies use simple routing algorithms, mainly deterministic XY algorithm, and in some cases basic adaptive routing algorithms, using on/off link status as an input for routing decisions. All these algorithms present no design challenges nor overheads.

- All the topologies use very simple injection algorithms based exclusively on queue occupancy thresholds. In all the cases, delay and area overhead are negligible.

7.3 Future Work

There are several research directions that can be taken out of this dissertation. This is a brief list of possible efforts for the future:

- To improve the 3D study for the HPC-Mesh topology providing results such as the power consumption. A similar direction is the analysis of thermal effects and manufacturing costs when dealing with other routing algorithms in addition of DOR routing.

- To study adaptive routing algorithms for the PC/HPC/HNPC-Mesh topologies and compare them with the deterministic DOR algorithm.

- To use different subnetworks of the topologies for different purposes. For instance, coherence protocols require different virtual channels to avoid protocol-level deadlocks. We could conceive a parallel network where different traffic classes are mapped into different subnetworks.

- Analyze the effect of higher/lower number of subnetworks in the different proposed topologies. Different performance/power numbers would arise.

7.4 Industry Internships and Related Publication

During the research period of this dissertation, two internships in Sun Microsystems (later became Oracle) have been achieved. Both were used to propose new topologies and new evaluations of topologies. In particular, crossbars, rings and meshes were evaluated in the presence of real applications and different design parameters. Also, the first proposal of this dissertation (NR-Mesh) was co-developed during the first internship.

The following list enumerates the papers related with this dissertation that have been published, or are under review process, in specialized conferences or journals. We outline for each contribution the novelties that are part of this dissertation.

- Camacho, J., Flich, J., Duato, J., Eberle, H., Gura N. and Olesinski, W. A performance evaluation of 2D-mesh, ring, and crossbar interconnects for chip multi-processors. In 2nd International Workshop on Network on Chip Architectures (NoCArc 2009), pages 51 -56.

The previous paper represent a study between Crossbars, Rings and 2D-Mesh topologies. This work allowed the setting of the simulation infrastructure for the analysis of the different topologies when running applications on top of SIMICS simulator.

Next papers are all related with the NR-Mesh topology, first using small crossbars within the end node (not included in the thesis), and finally using a simple injection algorithm (implemented using different tools with Verilog) decreasing the complexity and, then, obtaining negligible impact in the final node.

- Camacho, J., Flich, J. and Duato, J. Multiples Puertos de Inyección en una Red en Chip. In Actas XXII Jornadas de Paralelismo (JP2011), pages 273-278.

- Camacho, J., Flich, J., Duato, J., Eberle, H and Olesinski, W. A power-efficient network on-chip topology. In Proceedings of the Fifth International Workshop on Interconnection Network Architecture: On-Chip, Multi-Chip (INA-OCMC 2011), pages 23-26.

- Camacho, J., Flich, J., Duato, J., Eberle and H. Olesinski, W. Towards an Efficient NoC Topology through Multiple Injection Ports. In Proceedings of the 14th EUROMICRO Conference on Digital System Design (DSD 2011), pages 165-172.

Next papers are related with the PC-Mesh topology and its improvements. They mainly study the performance, power consumption using a power management logic, fault tolerant degree, and simulations with synthetic traffic and Splash-2 real applications with/without background traffic.

The second paper improves the first one mainly in terms of fault tolerance support, achieving the same degree using much less resources.

- Camacho, J., Flich, J., Roca A., Duato, J. PC-Mesh: A Dynamic Parallel Concentrated Mesh. In Proceedings of the International Conference of Parallel Processing (ICPP 2011), pages 642-651.

- Camacho, J. and Flich, J. HPC-Mesh: A Homogeneous Parallel Concentrated Mesh for Fault-Tolerance and Energy Savings. In Proceedings of the Seventh ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS 2011), pages 69-80.

These papers are summarize of the work done in the dissertation and belong to national and international conferences.

Bibliography

[1] Alaa R. Alameldeen, Carl J. Mauer, Min Xu, Pacia J. Harper, Milo M.K. Martin, Daniel J. Sorin, Mark D. Hill and David A. Wood, "Evaluating Non-deterministic Multi-threaded Commercial Workloads," in *Workshop on Computer Architecture Evaluation Using Commercial Workloads.*

[2] J.D. Balfour and W. J. Dally, "Design Tradeoffs for Tiled CMP On-Chip Networks," in *International Conference on Supercomputing*, June 2006.

[3] G. E. Blelloch, C. E. Leiserson, B. M. Maggs, C. G. Plaxton, S. J. Smith, and M. Zagha, "A Comparison of Sorting Algorithms for the Connection Machine CM-2," in *Proceedings of the Symposium on Parallel Algorithms and Architectures*, pp. 3-16, July 1991.

[4] E. Carara, F. Moraes, And N. Calazans, "Router architecture for high-performance NoCs," in *Proceedings of the 20th annual conference on Integrated circuits and systems design (SBCCI)*, 2007.

[5] C.-H. Chao, K.-Y. Jheng, H.-Y. Wang, J.-C. Wu, and A.-Y. Wu, "Traffic- and thermal-aware run-time thermal management scheme for 3D NoC systems," in *ACM/IEEE Int. Symp. Networks-on-Chip (NoCS)*, pp. 223-230, May 2010.

[6] X. Chen and L.-S. Peh, "Leakage Power Modeling and Optimization in Interconnection Networks," in *International Symposium on Low Power Electronics and Design*, pages 90-95, August 2003.

[7] W. J. Dally, "Express Cubes: Improving the Performance of k-ary n-cube Interconnection Networks," in *IEEE Transactions on Computers*, 40(9):10161023, September 1991.

[8] W. J. Dally and B. Towles. "Principles and Practices of Interconnection Networks," in Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[9] W. J. Dally, "Virtual-channel flow control," in IEEE Transactions on Parallel and Distributed Systems, 3(3):194205, March 1992.

[10] J. Duato, "A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks," in *IEEE Transactions on Parallel and Distributed Systems*, 1993.

[11] J. Duato, S. Yalamanchili, and Ni. L. M, "Interconnection Networks: An Engineering Approach," in Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[12] J. Flich, D. Bertozzi, "Designing Network On-Chip Architectures in the Nanoscale Era," in Chapman & Hall/CRC computational science series, 528 pages, 2011.

[13] B. Grot, J. Hestness, S. Keckler, O. Mutly, "Express Cube Topologies for On-Chip Interconnects," in *International Symposium on High-Performance Computer Architecture*, 2009.

[14] K. C. Hale, B. Grot, S. W. Keckler, "Segment Gating for Static Energy Reduction in Networks-on-Chip," in *International Workshop on Network-on-Chip Architectures*, December 2009.

[15] W.-H. Hu, S. E. Lee, and N. Bagherzadeh, "DMesh: a Diagonally-Linked Mesh Network-on-Chip Architecture", in *First International Workshop on Network on Chip Architectures Workshop*, 2008.

[16] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose, "Microarchitectural Techniques for Power Gating of Execution Units," in *International Symposium on Low Power Electronics and Design*, pages 32-37, August 2004.

[17] A. Kahng, B. Li, L.-S. Peh and K. Samadi, "ORION 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Explo-

ration," in *Design Automation and Test in Europe (DATE)*, Nice, France, April 2009.

[18] J. Kim, J. Balfour, W. Dally, "Flattened Butterfly Topology for On-chip networks," in *International Symposium on Microarchitecture*, December 2007.

[19] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha, "Express Virtual Channels: Towards the Ideal Interconnection Fabric," in *International Symposium on Computer Architecture*, pages 150-161, May 2007.

[20] C. Liu, L. Zhang, Y. Han, X. Li, "Vertical interconnects squeezing in symmetric 3D mesh Network-on-Chip," in *ASP-DAC 2011*: 357-362, 2011.

[21] I. Loi, F. Angiolini, and L. Benini, "Supporting vertical links for 3d networks-on-chip: Toward an automated design and analysis flow," in *Proceedings of the Nano-Net Conference*, pp. 23-27, 2007.

[22] P. S. Magnusson et al., "Simics: A full system simulation platform," in Computer, 35(2):50-58, 2002. IEEE Computer Society Press.

[23] M. Martin, D. Sorin, B. Beckmann, M. Marty, M. Xu, A. Almadeen, K. Moore, M. Hill and D. Wood, "Multifacet, a general execution-driven multiprocessor simulator (GEMS) toolset," in *Computer Architecture News*, September 2005.

[24] H. Matsutani, M. Koibuchi, D. Wang, and H. Amano, "Adding Slow-Silent Virtual Channels for Low-Power On-Chip Networks," in *International Symposium on Networks-on-Chip*, pages 23-32, April 2008.

[25] H. Matsutani, M. Koibuchi, H. Amano, and D. Wang, "Run-time Power Gating of On-Chip Routers Using Look-Ahead Routing.," in *Asia and South Pacific Design Automation Conference*, pages 55-60, January 2008.

[26] A. Mello, L. Copello, O. Gehm, N. Laert, V. Calazans, Evaluation of Routing Algorithms on Mesh Based NoCs, in Technical Report Series, No. 040. Faculty of Informatics, Pontifícia Universidade Católica do Rio Grande do Sul, Brasil, May 2004.

[27] G. D. Micheli , L. Benini, "Networks on Chips: Technology and Tools (Systems on Silicon)," in Morgan Kaufmann Publishers Inc., San Francisco, CA, 2006.

[28] S. Noh, V.-D. Ngo, H. Jao and H.-W. Choi, "Multiplane Virtual Channel Router for Network-on-Chip Design", in *First International Conference on Communications and Electronics (ICCE)*, 2006.

[29] V. F. Pavlidis and E. G. Friedman, "3-D Topologies for Networks-on-Chip," in *IEEE TVLSI*, October 2007.

[30] D. Pham et al., "Overview of the Architecture, Circuit Design, and Physical Implementation of a First-Generation Cell Processor," in *IEEE Journal of Solid-State Circuits*, 41(1):179196, January 2006.

[31] M. Powell, S.-H. Yang, B. Falsafi, K. Roy, and T. N. Vijaykumar, "Gated-Vdd: a Circuit Technique to Reduce Leakage in Deep-Submicron Cache Memories," in *International Symposium on Low Power Electronics and Design*, pages 90-95, July 2000.

[32] S. Pasricha, "Exploring serial vertical interconnects for 3D ICs," in *Proceedings of the 46th Annual Design Automation Conference*, pp. 581-586, ACM, 2009.

[33] A. Pullini et al, "Bringing NoCs to 65nm," in *IEEE Micro Magazine*, Vol. 12, Nr. 5, pp. 75-85, IEEE Press, September 2007.

[34] V. Soteriou and L.-S. Peh, "Dynamic Power Management for Power Optimization of Interconnection Networks Using On/Off Links," in *International Symposium on High Performance Interconnects*, pages 15-20, August 2003.

[35] S. Vangal et al., "An 80-Tile 1.28 TFLOPS Network-on-Chip in 65nm CMOS," in *International Solid-State Circuits Conference*, pages 9899, February 2007.

[36] E. Waingold et al., "Baring It All to Software: RAWMachines," in *IEEE Computer*, 30(9):86-93, September 1997.

[37] R. Weerasekera, L. Zheng, D. Pamunuwa, and H. Tenhunen, "Extending systems-on-chip to the third dimension: performance, cost and technological tradeoffs," in *Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD).* IEEE, pp. 212-219, 2007.

[38] D. Wentzlaff et al., "On-Chip Interconnection Architecture of the Tile Processor," in *IEEE Micro*, 27(5):1531, September/October 2007.

[39] P. T. Wolkotte, G. J. M. Smit, G. K. Rauwerda, and L. T. Smit. "An energy-efficient reconfigurable circuit-switched network-on-chip," in *IPDPS05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2005) - Workshop 3*, page 155.1, Washington, DC, USA, 2005. IEEE Computer Society.

[40] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, A. Gupta, A., "The SPLASH-2 programs: characterization and methodological considerations," in *22nd Annual Int. Symposium on Computer Architecture*, Italy, June 22 - 24, pp. 24-36, 1995.

[41] T. C. Xu, P. Liljeberg, and H. Tenhunen, "A study of through silicon via impact to 3d network-on-chip design," in *Proceedings of the 2010 International Conference on Electronics and Information Engineering (ICEIE 2010)*, August 2010.

[42] C. Zhu, Z. Gu, L. Shang, R. Dick, and R. Joseph, "Three-dimensional chip-multiprocessor run-time thermal management," in *IEEE Transactions on Computer-Aided Design*, vol. 27, no.3, 2008.

[43] A.E. Zonouz et al. "A Fault Tolerant NoC Architecture for Reliability Improvement and Latency Reduction," in *2009 Euromicro Conference on Digital System Design*, 1999.

[44] 45nm Nangate opensource library available at http://www.si2.org/openeda.si2.org/projects/nangatelib.

[45] Broadband Engine Processor available at http://en.wikipedia.org/wiki/Cell_(microprocessor).

[46] Encounter RTL Compiler available at http://www.cadence.com/products/ld/rtl_compiler/pages/default.aspx

[47] Intel Teraflops Research Chip available at http://www.intel.com/pressroom/kits/teraflops.

[48] Place and Route from cadence available at http://www.cadence.com/products/pages/default.aspx.

[49] Power Compiler from Synopsys available at http://www.synopsys.com/tools/implementation/rtlsynthesis/pages/powercompiler.aspx.

[50] Single-chip Cloud Computer available at http://techresearch.intel.com/articles/Tera-Scale/1826.htm.

[51] Source Routing (Linktionary term) available at http://www.linktionary.com/s/source_routing.html.

[52] Tile-Gx Processors Family available at http://www.tilera.com/products/TILE-Gx.php.

[53] TOP500 Supercomputing Sites available at http://www.top500.org.