Exploration of multicore systems based on silicon integrated communication networks

Charles Emmanuel Effiong

THÈSE POUR OBTENIR LE GRADE DE DOCTEUR DE L'UNIVERSITÉ DE MONTPELLIER

En Université Montpellier

École doctorale Information, Structures, Systèmes (I2S)

Unité de recherche Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM)

Exploration of Manycore Systems based on Silicon Integrated Communication Networks

Présentée par Charles Emmanuel EFFIONG

Le 16 Novembre 2017

Sous la direction de Abdoulaye GAMATIE et Gilles SASSATELLI

Devant le jury composé de

Pierre BOULET, Professeur, Université de Lille, Sciences et Technologies, FRANCE	Président du jury
Abdoulaye GAMATIE, Directeur de Recherche au CNRS, LIRMM - UM, FRANCE	Co-directeur de thèse
Gilles SASSATELLI, Directeur de Recherche au CNRS, LIRMM - UM, FRANCE	Co-directeur de thèse
Guy GOGNIAT, Professeur, Université Bretagne Sud – UBL, FRANCE	Rapporteur
Alberto GARCIA-ORTIZ, Professeur, University of Bremen, GERMANY	Rapporteur
Lionel TORRES, Professeur, LIRMM – UM, FRANCE	Examinateur

UNIVERSITÉ DE MONTPELLIER

EXPLORATION OF MANYCORE SYSTEMS BASED ON SILICON INTEGRATED COMMUNICATION NETWORKS

Presented by
Charles EFFIONG

Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier
Université de Montpellier

This dissertation is submitted for the degree of Doctor of Philosophy
November 2017

Dedicated to my Beloved Parents

Acknowledgements

This research work was carried out at the Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM), France in fulfillment of a Ph.D. degree in Microelectronics. During the course of my research at the lab, I received tremendous support from people who contributed to my research work.

First and foremost, my sincere gratitude goes to my thesis supervisor, Prof. Abdoulaye GAMATIE, who provided me with much guidance throughout my research work. I am thankful for all his ideas, encouragement and support. He kept me motivated and inspired during the course of my research activities. I am also thankful for his help in reviewing my slides, conference/journal manuscripts, and my Ph.D. thesis. I have learned to be thorough, to stay motivated and pay attention to detail from him.

On a similar note, my appreciation goes to my thesis co-supervisor Prof. Gilles SASSATELLI for his professional advice during my Ph.D. He literally made a researcher scientist out of me. I have learned to broaden my horizon and better carry out my research activities from him. His suggestions and support helped me bring my Ph.D. research ideas to life. He believed in me even when the going was tough and kept looking at the bright side of things. His optimism stood out to me and kept me going through my research.

Special thanks to my thesis reviewers, Prof. Guy GOGNIAT and Prof. Alberto GARCIA-ORTIZ, for reviewing my Ph.D. manuscripts and offering interesting feedbacks for improving the manuscript. Their feedback helped me better prepare for the defense. I am also thankful to all the members of the jury for making my defense a memorable one.

Special thanks to all my colleague at the lab who contributed in one way or the other to my thesis.

I would like to especially thank my parents Mr. and Mrs. Effiong for their consistent support and prayers for me. They have always been there for me at all times. Their labors of love will not be in vain. Thanks to all my families, friends and loved ones for their moral support.

Above all, I am grateful to God the giver of life and good things. I am thankful for His unfailing love and compassion towards me. He heard and answered my prayers and showered His blessings upon me.

Charles EFFIONG
Montpellier
November 2017

Résumé

De nos jours, manycores de calcul sont intégrés sur une seule puce pour satisfaire les demandes toujours croissantes des applications des systèmes haute performance et basse consommation. Cela a conduit au passage des systèmes mono-cœur (composés d'un cœur de calcul sur une puce qui peut exécuter seulement un processus à la fois) aux systèmes multi-cœurs composés de deux à huit cœurs et récemment aux systèmes manycores composés de dizaines voire de centaines de cœurs intégrés sur une seule puce.

Comme le prévoit l'ITRS, le nombre de cœurs de calcul sur une puce continuera d'augmenter, ouvrant la voie à des systèmes encore plus avancés. Cette transition a conduit au développement d'architectures de calcul plus efficaces offrant de meilleures performances et une consommation d'énergie plus faible pour répondre aux différentes demandes d'applications, car les tâches peuvent être exécutées en parallèle contrairement à un système mono-cœur. On a observé qu'un processeur de 2 cœurs exécutant plusieurs applications est 1,5 fois plus rapide qu'un processeur mono-cœur. Une manière classique d'améliorer la performance des systèmes mono-cœur est d'augmenter la fréquence d'horloge, ce qui entraîne une exécution plus rapide du programme. Cependant, la fréquence d'horloge ne peut être augmentée au-delà d'un certain seuil (4GHz pour la plupart des processeurs disponibles dans le commerce) à cause du taux de dissipation de puissance insoutenable. Par conséquent, l'augmentation des performances des processeurs mono-cœur est limitée par la dissipation de puissance. Contrairement à un processeur mono-cœur, chaque cœur d'un processeur manycores peut fonctionner à une fréquence d'horloge inférieure, répartissant ainsi la puissance consommée parmi les cœurs, ce qui entraîne une consommation d'énergie plus faible et des performances supérieures. Des études ont montré que l'utilisation d'un processeur double cœurs à une fréquence d'horloge réduite de 20

De nombreux cœurs de calcul sont maintenant intégrés sur une puce afin d'obtenir un degré encore plus élevé de traitement en parallèle. Un exemple est le processeur Intel Xeon Phi avec 64 cœurs de calcul, une fréquence de base de 1,30 Ghz et gravé en technologie 14 nm. Selon les exigences de l'application, les systèmes multi-cœurs peuvent être composés de cœurs de calcul homogènes ou hétérogènes. Un exemple est un système sur puce, constitué de cœurs de calcul spécifiques à des applications (tels que le cœur de traitement de signal numérique (DSP) pour l'exécution

d'applications multimédia avec des calculs mathématiques intensifs). Une telle architecture permet d'exécuter différentes applications sur le système.

L'augmentation croissante du nombre de cœurs de calcul est proportionnelle à la demande de réseaux d'interconnexions évolutifs sur puce pouvant fournir une communication à grande vitesse entre les différents cœurs. Des études ont montré que, au fur et à mesure que le nombre de cœurs augmente, l'interconnexion entre ces cœurs devient un facteur dominant, ce qui impose des contraintes de performance et de puissance importantes sur l'ensemble des performances du système. Ainsi, la nécessité d'une interconnexion évolutive sur puce pouvant fournir un transfert de données à large bande passante entre les cœurs de calcul est essentielle.

Le bus d'interconnexion est un moyen de communication partagé par les cœurs de calcul d'un système pour le transfert des données. Un bus peut être conçu de manière simpliste (car il est composé principalement de fils et d'un arbitre pour contrôler l'accès au bus) de sorte à avoir un moyen de communication peu coûteux en terme de conception. Cependant, on observe une dégradation des performances et une forte consommation électrique du bus d'interconnexion avec l'augmentation du nombre de cœurs de calcul. L'une des raisons est qu'un seul bus partagé ne peut pas assurer de transfert de données simultané. Ainsi, les accès au bus sont bloqués jusqu'à la fin d'une transaction en cours, ce qui entraîne une dégradation des performances. En outre, un bus plus grand signifie une longueur de fil plus longue et des retards de fil associés qui limitent la bande passante du bus. Par conséquent, il n'est pas évolutif et est inadapté pour les SoCs à multi-cœurs.

Afin de surmonter les limites du bus d'interconnexion, l'interconnexion crossbar a été adoptée. On le retrouve dans le processeur 8-cœurs Sun Niagara. L'interconnexion crossbar assure un transfert de données point à point à grande vitesse entre les cœurs de calcul. Contrairement au bus d'interconnexion, le transfert de données simultané est possible à travers l'interconnexion crossbar puisque chaque nœud est connecté à tous les autres nœuds du réseau. Ainsi, le transfert de données en parallèle peut s'effectuer entre manycores de calcul, ce qui entraîne une faible latence. L'amélioration de la performance à travers l'interconnexion crossbar est cependant avec coût significatif en raison d'un grand nombre de fils. Aussi, l'arbitrage devient plus difficile avec l'augmentation croissante du nombre de cœurs de calcul. L'interconnexion crossbar convient à un petit nombre de nœuds, mais elle n'est pas évolutive, car le coût du fil devient encore plus coûteux avec un grand nombre de nœuds.

Contrairement à l'interconnexion traditionnelle sur puce, les Network-on-chip (NoCs) sont apparu comme une interconnexion alternative mature pour manycore

architectures car il offre une évolutivité améliorée et une efficacité énergétique accrue. Le Réseau-sur-puce (NoC) assure une communication parallèle sans une augmentation de coût significative comme dans l'interconnexion crossbar. Le NoC est utilisé dans le processeur 64-coeurs Tilera TILE64. Contrairement aux bus et crossbar interconnexion, le Réseau-sur-puce offre une diversité de chemins car plusieurs chemins existent entre les cœurs de calcul source et destination. La diversité de chemins peut être exploitée pour atténuer les pertes de performance causées par des conflits de réseau élevés, car des chemins de réseau alternatifs peuvent être utilisés. Par conséquent, cette thèse porte sur le NoC. Le NoC est composé de routeurs, interconnectés par des liens de données point à point. Les routeurs jouent le rôle essentiel du routage des données des nœuds source vers les nœuds de destination dans le réseau.

Le routeur standard bufférisé d'un NoC est composé de buffers d'entrée et / ou de sortie qui sont utilisés pour stocker temporairement des paquets qui ne peuvent être acheminé à leur sortie souhaitée en raison de conflits de réseau. Le stockage du paquet dans les buffers est bénéfique car il garantit que les lignes du réseau sont disponibles pour les autres paquets. D'autre part, des études ont montré que les buffers sont la composante la plus chère en termes de surface de silicium et de consommation d'énergie dans un routeur NoC. Bien que les buffers soient très coûteux, des études ont montré qu'ils sont souvent inutilisés (c'est-à-dire inactifs ou sous-utilisés), en particulier lorsqu'ils exécutent des applications avec un comportement de trafic non uniforme et / ou des comportements en rafale. La raison en est que les routeurs typiques bufférisée allouent un ensemble de buffers à leurs ports d'entrée et / ou de sortie et ces buffers ne peuvent être exploités que par des flux de données traversant ces ports. Cela entraîne une dégradation des performances pour le trafic non uniforme car seule une fraction du nombre total de tampons est disponible pour l'exploitation par les paquets.

Afin de maximiser l'utilisation des ressources buffer, cette thèse propose une nouvelle architecture de routeur NoC inspirée des ronds-points de circulation à plusieurs voies de la vie réelle. Ce routeur s'appelle R-NoC et sert d'architecture de routeur initial ou de base. Le concept R-NoC fournit une architecture hautement adaptable, qui permet au routeur d'être configuré pour répondre à de nombreuses topologies de réseau et demandes d'applications. Le concept R-NoC permet de partager les ressources disponibles par plusieurs ports. Le partage des ressources disponibles parmi plusieurs ports permet de prendre en charge les applications présentant des caractéristiques de trafic différentes. L'architecture du routeur R-NoC peut être facilement adaptée pour prendre en charge manycores de traitement sans entraîner d'importantes modifications du routeur. La connexion de manycores à un routeur réduit le nombre de routeurs nécessaires dans le réseau et participe à

une diminution de la latence puisque le nombre de paquets de données traversant le réseau est réduit.

Les architectures de routeur inspirées des ronds-points sont particulièrement attrayantes car elles offrent un partage de ressources intrinsèque et efficace pour améliorer les performances du réseau. Cependant, ils sont enclins aux blocages dus à leurs architectures en anneau / cyclique. Afin d'éviter de tels blocages, un algorithme capable de générer des configurations de routeur R-NoC sans blocage est proposé. La principale contribution ici réside dans la réalisation d'une nouvelle architecture de routeur inspirée des ronds-points offrant des améliorations de performance par rapport aux routeurs standards à l'entrée bufférisée sans compromettre le coût de la surface de silicium et la consommation d'énergie. Ce travail diffère de Rotary, qui est aussi un routeur inspiré des ronds-points et qui offre une amélioration de la performance du réseau mais qui introduit également des coûts supplémentaires en termes de surface de silicium et consommation d'énergie associés au contrôle de flux utilisé pour éviter les pannes.

Cette thèse a ensuite exploré d'autres façons d'améliorer les performances du routeur R-NoC de base sans compromettre du surface de silicium et puissance. Pour ce faire, l'impact de l'introduction de buffers supplémentaires et l'utilisation de voies supplémentaires sur les performances et la surface est d'abord étudié. Cependant, d'autres recherches révèlent que les améliorations de performance sont accompagnées d'un coût de surface supplémentaire associé aux buffers. La contribution principale ici est d'exploiter la topologie hautement adaptable du routeur R-NoC pour produire des configurations de routeurs offrant des compromis topologiques variés pour améliorer considérablement les performances du réseau sans compromettre du surface de silicium et puissance.

Étant donné que la topologie du réseau affecte de manière significative la performance globale du réseau, la puissance, la fiabilité et le coût, cette thèse étend le routeur R-NoC à d'autres topologies de réseau pour d'autres améliorations de performance sans introduire de coût importants en terme de surface/ puissance par rapport aux routeurs standards bufférisés en entrées et / ou sorties. À cette fin, des ports supplémentaires sont ajoutés à la configuration initiale du routeur R-NoC afin de réaliser différentes configurations de routeurs pour différentes topologies. Bien que plusieurs topologies de réseau puissent être prises en charge, cette thèse étudie R-NoC pour la topologie du réseau maillé diagonalement, nommé R-NoC-D et la topologie de maille. Par rapport au routeur standard bufférisé en entrée, le R-NoC-D routeur réalisent des améliorations significatives de la performance sans impact important en termes de surface de silicium et consommation d'énergie. Cela est possible en exploitant la topologie hautement adaptable de R-NoC.

En résumé, l'évaluation montre que R-NoC peut fournir des performances améliorées

sur un routeur tamponné typique. L'amélioration des performances ne se fait pas au détriment de la surface et de l'alimentation, ce qui rend le R-NoC adapté au NoC avec des contraintes d'alimentation étroites. De plus, R-NoC peut être étendu à d'autres topologies de réseau pour améliorer les performances et réduire la consommation d'énergie.

Mots-clés : réseau sur puce, partage de ressources, routeur, topologie circulaire de NoC, performance, bande passante, surface, consommation énergétique.

Abstract

More computing cores are now being integrated on a single chip in order to meet the ever-growing application demands for high performance and low power computing systems. As the number of cores continues to grow, so is the demand for scalable on-chip communication networks that can deliver high-speed communication among the cores. Contrary to traditional on-chip networks, Networks-on-Chip (NoCs) have emerged as a mature alternative interconnect for manycore architectures since it provides enhanced scalability and power efficiency.

Typical NoC routers consist of buffers which serve as temporary data storage. However, studies have shown that buffers are often unutilized (i.e. idle or underutilized) especially when executing applications with non-uniform traffic patterns or bursty behaviours. This is because most typical routers dedicate a set of buffers to their input and/or output ports and these buffers can only be exploited by dataflows using them, which leads to significant performance degradation. Therefore, router architectures capable of maximizing buffer utilization for performance gains are indispensable.

In order to maximize buffer resource utilization, this thesis proposes a novel NoC router concept called Roundabout NoC (*R-NoC*) that is inspired by real-life multi-lanes traffic roundabout. Contrary to existing approaches, *R-NoC* provides intrinsic and effective resource utilization. However, roundabout-inspired routers are susceptible to deadlocks due to their ring-like architecture. Contrary to existing solutions, *R-NoC* achieves deadlock-freeness and enhanced network performance over typical NoCs without compromising network area/power. This thesis further exploits *R-NoC* highly parametric architecture in order to produce different router configurations with varying topological trade-offs for performance gains without sacrificing area.

Keywords: Network-on-Chip, resource sharing, router, deadlock-freeness, circular NoC topologies, performance, throughput, area, power consumption.

Contents

List of Figures

List of Tables

List of Abbreviations

2D	**T**wo **D**imensional
3D	**T**hree **D**imensional
AFC	**A**daptive **F**low-**C**ontrol
BE	**B**est **E**ffort
BFC	**B**ubble **F**low-**C**ontrol
CBS	**C**ritical **B**ubble **S**cheme
CDG	**C**hannel **D**ependency **G**raph
CMOS	**C**omplementary **M**etal-**O**xide **S**emiconductor
DI	**D**elay-**I**nsensitive
DL	**D**ate**L**ine
DMESH	**D**iagonally-**L**inked **M**esh
DOR	**D**imension-**Or**dered **R**outing
DSP	**D**igital **S**ignal **P**rocessing
DVFS	**D**ynamic **V**oltage and **F**requency **S**caling
EB	**E**lastic **B**uffers
FBFC-C	**F**lit **B**ubble **F**low-**C**ontrol **C**ritical
FBFC-L	**F**lit **B**ubble **F**low-**C**ontrol **L**ocalized
FIFO	**F**irst-**I**n-**F**irst-**O**ut
FSM	**F**inite **S**tate **M**achine
GALS	**G**lobally **A**synchronous **L**ocally **S**ynchronous
GS	**G**uarantee **S**ervice
HOL	**H**ead-**O**n-**L**ine Blocking
IBR	**I**nput **B**uffered **R**outer
IP	**I**ntellectual **P**roperty
LBS	**L**ocal **B**ubble **S**cheme
LEDR	**L**evel-**E**ncoded **D**ual-**R**ail
MANGO	**M**essage-passing **A**synchronous **N**oC providing **G**uaranteed **S**ervices
MMS	**M**ulti**M**edia **S**ystem
MWD	**M**ulti-**W**indow **D**isplay
NMAP	**N**ear **M**apping
NoC	**N**etwork-on-**C**hip
QoS	**Q**uality-of-**S**ervice

R-NoC	**R**oundabout **N**etwork-on-**C**hip
R-NoC-A	**R**oundabout **N**etwork-on-**C**hip **A**synchronous
R-NoC-DM	**R**oundabout **N**etwork-on-**C**hip **D**iagonally-linked **M**esh
R-NoC-SE	**R**oundabout **N**etwork-on-**C**hip **S**ynchronous **E**lastic
RC	**R**oute **C**omputation
RMAP	**R**andom **M**apping
RR	**R**ound **R**obin
RTL	**R**egister **T**ransfer **L**evel
SA	**S**witch **A**llocation
ST	**S**witch **T**raversal
SoC	**S**ystem-on-**C**hip
VA	**V**irtual-channel **A**llocation
VC	**V**irtual **C**hannel
VCT	**V**irtual-**C**ut-**T**hrough
VHDL	**V**ery **H**igh **S**peed Integrated Circuit **H**ardware **D**escription **L**anguage
VLSI	**V**e**r**y **L**arge **S**cale **I**ntegrated **C**ircuit
VOPD	**V**ideo **O**bject **P**lan **E**ncoder
WBFC	**W**orm **B**ubble **F**low-**C**ontrol

Chapter 1

Introduction

«"Writing is an exploration. You start from nothing and learn as you go."»

E. L. Doctorow

Contents

1.1 Scalable on-chip communication fabric

The advancement in Very Large Scale Integrated circuit (VLSI) process technology, coupled with the ever-increasing demand for more efficient computing systems, has led to the transition from single core systems (consisting of one core on a chip that can run only one process at a time) to multi-core with two to eight cores and recently to many-cores consisting of tens, hundreds or thousands of cores integrated on a single chip. As predicted by International Technology Roadmap for Semiconductors [52], the number of cores on a chip will continue to increase, paving the way for even more advanced systems. This transition has led to the development of more efficient computing architectures that deliver better performance and power characteristics for meeting varying applications demands since tasks can be executed in parallel contrary to a single core system.

A classical way of improving the performance of single core systems is by increasing the clock-frequency, which leads to faster program execution. However, the clock frequency cannot be increased beyond a certain threshold (4GHz for most commercially available processors) due to unsustainable rates of power dissipation [83]. Hence, increasing performance of single core processor is limited by power dissipation. Contrary to a single core processor, each core in multi/manycore processors can operate at a lower clock-frequency, thereby distributing power consumption among the cores, which leads to lower energy consumption and higher performance. Studies have shown that operating a dual-core system at 20% reduced clock frequency leads to almost 2 times performance improvement and consumed approximately similar power to that of a single core processor operating at maximum frequency [104]. Many processing cores are now being integrated on a chip in order to achieve an even higher degree of parallel processing. An example is the Intel® Xeon Phi™ Processor with 64 processing cores, a processor based frequency

of 1.30 *GHz* and based on 14 *nm* process technology [51]. Depending on application requirements, manycore systems can be composed of homogeneous or heterogeneous cores. Fig. 1.1 shows the architecture of a System-on-Chip, which consists of dedicated application specific cores (such as digital signal processing cores (DSP) for executing multimedia applications with intensive mathematical computations). Such an architecture allows varying applications to be executed on the system [83].

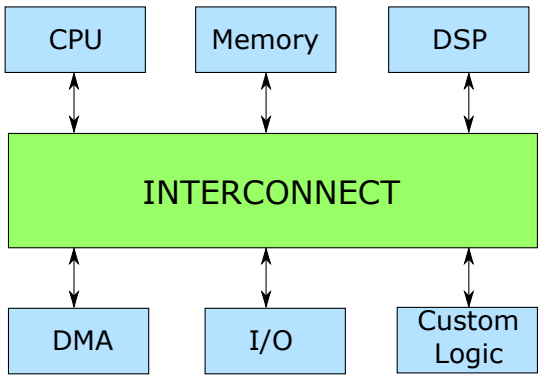


FIGURE 1.1: Generic Systems-on-Chip (SoC) architecture

A key component of the manycore system is the on-chip interconnect, which connects all the cores together. The interconnect is considered the backbone of manycore systems, ensuring inter-core communication. Studies have shown that as the number of cores continues to grow, the interconnect becomes a dominating factor, imposing significant performance and power constraints on the overall system performance [15]. Thus, the need for scalable on-chip interconnect that can deliver high bandwidth and low latency inter-core data transfer is critical.

Bus interconnect

The bus interconnect provides a shared communication medium for inter-core data transfer. A bus can be simply designed (since it is made up of mostly wires and an arbiter to controls access to the bus) and provides a low-cost communication fabric. However, the bus interconnect suffers from poorer performance and higher power consumption with increasing number of cores. One reason is that a single shared bus does not provide concurrent data transfer. Thus, accesses to the bus are blocked until an ongoing transaction completes, which leads to performance degradation in manycore systems. In addition, larger bus means longer wire length and associated wire delays which limit the bus bandwidth. Hence, it is not scalable and unsuitable for manycores SoCs [47]. Techniques such as bridging shared buses have been introduced to improve performance. An example is the *AMBA bus* from ARM [65]. However, they still suffer poor performance with increasing number of cores [47].

Crossbar interconnect

In order to overcome the limitations of the bus interconnect, the crossbar interconnect was adopted. The crossbar is used in 8-core Sun Niagara [59]. The crossbar interconnect provides high-speed point to point data transfer between cores. Unlike the shared bus interconnect, crossbar provides parallel data transfer since each node is connected to every other node in the network. Thus, parallel data transfer can occur for varying destination cores, which leads to lower latency. However, performance improvement of the crossbar over the bus comes at a significant cost due to a large number of wires. In addition, the crossbar becomes more difficult to arbitrate with increasing number of cores. The crossbar is suitable for a small number of nodes but not scalable as wire cost becomes even more expensive with manycores [1].

Network-on-Chip (NoC) interconnect

The Network-on-Chip (NoC) provides parallel communication but without significant cost overhead as in the crossbar interconnect. The NoC is used in 64-core Tilera TILE64 [8]. NoC uses shorter wires, which helps to reduce the overall total-wire length in the chip, and supports the distributed nature of modern chip architecture [9]. Contrary to bus and crossbar interconnects, NoC provides path diversity as several paths exist between a source and destination cores. The path diversity can be exploited to mitigate performance loss caused by high network contentions since alternative network paths can be utilized. Thus network-on-chip has emerged as alternative on-chip interconnect for manycores systems.

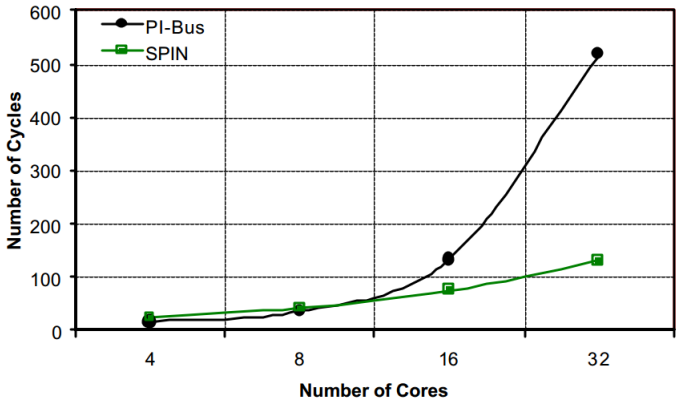


FIGURE 1.2: Salability of bus versus Network-on-Chip [4]

Fig 1.2 shows the scalability of SPIN NoC and the bus interconnects. It is observed that the bus suffers poor performance as the number of cores increases. Conversely, NoC provides better scalability as the number of cores grow [4]. Thus, NoC is more suitable for manycores system.

1.2 NoC design challenges

Although NoC is a viable alternative on-chip interconnect for systems with multiple cores, there exist some challenges with its design. This section gives an overview of some of the key NoC design challenges, while detailed presentation is given in Chapter 2, section 2.2. Some of the key challenges facing NoC design are described next.

- **Power consumption** is one of the biggest challenges facing NoC design. It has been observed that current NoCs still consume significant power in manycores systems [15]. Previous work [15] showed that manycore communication network can consume up to $500mW$ power at each node. Existing power management techniques such as *sleep-states* incur latency overhead, which degrades system performance [15]. Therefore, techniques to reduce as much as possible the power consumed by NoCs without impairing system performance are necessary [15, 75].

- Another challenge facing NoC design is **quality-of-service (QoS)**, which refers to predictable NoC services (e.g. bandwidth, latency). Most existing techniques for providing QoS are expensive in terms of design complexities [79]. A classical way of providing QoS is by creating a connection between source and destination nodes (i.e. *circuit switching*) before actual data transfer. This method is used in Æthereal [90] and aSoC [64]. However, it has been observed that this method leads to poor scalability since router area growth is proportional to the number of required connections [69]. Also, managing circuits introduces additional latency.

- **Latency** is another challenge facing NoC design. This is because NoCs are required to provide low latency under stringent power, area and delay constraints. *Adaptive* routing techniques have been proposed to provide faster data routing in NoC. However, this often requires additional complexity in terms of area and resolving issues such as *deadlocks, livelocks and starvation* [24, 89] (discussed in Chapter 2) and power consumption. Thus, techniques to provide low latency without compromising NoC area and power is an important goal in NoC design [79].

1.3 Buffer resource sharing in NoC

Most typical NoC routers dedicate a set of buffers to their input and/or output ports (such routers are normally called input or output buffered routers). The buffers are used to temporarily store packets that cannot proceed to their desired output ports

due to network contentions. Storing packets in buffers ensures that the router link bandwidth is not unnecessarily consumed, thus available for use by other packets. On the other hand, studies have shown that buffers are often unutilized (i.e. idle or underutilized) especially when executing applications with non-uniform traffic patterns. The reason being that most typical routers dedicate a set of buffers to their input and/or output ports and these buffers can only be exploited by data-flows using them. This leads to performance degradation since only a fraction of the total buffering resource are available for use.

In order to improve network performance, router architectures capable of maximizing the use of buffering resource are necessary. Router architectures capable of sharing the buffering resources in the router for performance gains are particularly of interest. Such routers decouple the buffers from a specific port, thus allowing several ports to share the router buffers.

Unlike most typical shared buffer routers, *roundabout or ring-like* inspired router architectures are attractive because they provide intrinsic resource sharing, which improves network performance and reduces design complexity. An example is the Rotary router [2] which consists of dual-rings shared by multiple input ports. However, such routers are susceptible to deadlocks due to their *ring-like or cyclic* architecture. In order to avoid deadlocks, the Rotary router used a combined *virtual-cut-through and bubble* flow-control known as *(local-bubble-scheme)*, where at least an empty packet-sized buffer is maintained in the ring for achieving deadlock-freeness. This introduces additional area/power overhead (associated with large buffers), and also incur higher packet injection latency. Therefore, roundabout-inspired routers capable of providing performance improvement without compromising area and power consumption are attractive.

1.4 Thesis objectives and contributions

The main goal of this thesis is to design a novel scalable, high performance and power efficient on-chip interconnect network, named *Roundabout NoC (R-NoC)*, for systems with many intellectual property cores.

Objective 1 The first objective of this thesis is to answer the following questions:

> How to maximize NoC resource utilization for improving network performance, without introducing significant area/power overheads? How to resolve issues such as deadlocks associated with resource sharing without compromising area and power?

In order to meet Objective 1, a NoC router architecture inspired by real-life multi-lane traffic roundabouts is considered. This router is called *R-NoC* router and serves

as the initial or base router architecture. As stated earlier, roundabout-inspired router architectures are particularly attractive as they provide intrinsic and effective resource sharing for enhancing network performance. However, such architectures are deadlock-prone. In order to avoid such deadlocks, an algorithm capable of generating deadlock-free *R-NoC* router configurations is proposed. The main contribution here lies in realizing a novel roundabout-inspired router architecture offering performance improvements over typical input buffered routers and without compromising area and power consumption. This work differs from Rotary [2], which is a similar roundabout-inspired router that provides network performance improvement but also introduces significant area and power overheads associated with the flow-control utilized for deadlock-avoidance. This work is supported by the following publications:

- [36]: Charles Effiong, Gilles Sassatelli, Abdoulaye Gamatie. Scalable and Power-Efficient Implementation of an Asynchronous Router with Buffer Sharing. In *Euromicro Conference on Digital System Design (DSD 2017), Vienna, Austria*, September 2017.

- [35]: Charles Effiong, Gilles Sassatelli, Abdoulaye Gamatie. Roundabout: a Network-on-Chip Router with Adaptive Buffer Sharing. In *IEEE International NEW Circuits And Systems (NEWCAS 2017)*, Strasbourg, France, June 2017.

Objective 2 The second objective of this thesis is to explore alternative ways of improving the performance of base *R-NoC* without area/power degradation.

In order to achieve Objective 2, the impact of introducing additional buffers and using additional lanes on performance and area is first investigated. The evaluation shows that network performance can be significantly improved through the use of additional buffers in the router. However, further investigations reveal that the performance improvements come with an additional area cost associated with buffers. The main contribution here is in exploiting the highly-adaptive *R-NoC* topology in order to produce router configurations offering varying topological trade-offs for significantly improving network performance without sacrificing area. This work is supported by the following publication:

- [34]: Charles Effiong, Gilles Sassatelli, Abdoulaye Gamatie. Distributed and Dynamic Shared-Buffer Router for High-Performance Interconnect. In *The International Symposium on Networks-on-Chip (NOCS)*, Seoul, South Korea, October 2017.

Objective 3 The third objective of this thesis is to answer the following question:

Can the *R-NoC* router architecture be extended to other network topologies for further performance improvements without introducing significant area/power overheads compared to typical input buffered routers?

For this purpose, additional ports are added to the initial *R-NoC* router configuration in order to realize varying router configurations for various topologies. Although several network topologies can be supported, this thesis considers *R-NoC* for diagonally-linked mesh network topology, named *R-NoC-D*. Compared to typical input buffer router, *R-NoC-D* routers achieve significant performance improvements without significantly impacting area and power overhead. This is made possible by exploiting *R-NoC* highly-adaptive topology.

1.5 Thesis organization

The remainder of this thesis is organized as follows:

- **Chapter 2** gives an overview of Networks-on-Chip (NoCs) concepts in terms of topology, flow-control, switching and routing. It then discusses the problem of resource underutilization in NoCs, which motivates the need for resource sharing. Challenges such as deadlocks and design complexity to realizing deadlock-free NoC router architectures are also covered. The motivation for router architectures with inherent resource sharing and desirable mechanisms are presented in this chapter.

- **Chapter 3** provides an exhaustive review of existing NoC such as buffered, bufferless, minimum buffers (i.e. min-buffered routers), shared buffers and *roundabout-inspired* architectures. The analysis of state-of-the-art deadlock-free flow-control techniques such as dateline, local bubble scheme, flit-bubble flow-control is covered in this chapter.

- **Chapter 4** presents the general principles of the proposed *R-NoC* router. It begins with presenting the benefits of the approach, after which the operation of the initial/base topology is presented. The deadlock problem in the base topology is identified. Then, an algorithm to generate deadlock-free *R-NoC* router configurations is presented. Finally, the algorithm is then applied to realizing deadlock free *R-NoC* router configuration for different network approaches.

- **Chapter 5** presents the synchronous elastic and asynchronous based implementations of *R-NoC*. The chapter ends with a discussion on both implementation styles.

- **Chapter 6** presents the synchronous and asynchronous evaluation of the baseline *R-NoC*. The chapter also evaluates further synchronous and asynchronous

R-NoC topologies. It ends with a quantitative comparison of both versions (i.e. synchronous and asynchronous).

- **Chapter 7** presents the evaluation of *R-NoC* for the Diagonally-linked mesh network topology (i.e. *R-NoC-D*). A quantitative comparison of the considered topologies is carried out in the chapter.
- **Chapter 8** provides a global discussion on all presented results and gives future research directions.

Chapter 2

Networks-on-Chip (NoCs) design

«"Don't be afraid to give up the good to go for the great."»

John D. Rockefeller

Contents

2.1 General introduction to NoCs

Compared to traditional bus and crossbar interconnects, Network-on-Chip (NoC) is the most suitable interconnect for manycores system as discussed in Chapter 1. NoC provides enhanced scalability, uses shorter wires, which help to reduce the overall total-wire length in the chip (thereby reducing wire delays), and supports the distributed nature of modern chip architecture. This section discusses some of the factors that influence NoC performance.

NoC consists of routers interconnected by data-links. Routers play a key role of routing packets from source to destination nodes in the network, while the links are sets of wires that connect the routers together. The manner in which the routers are arranged in the network is governed by the topology. Popular topologies include mesh, torus, and ring. NoC topologies are often represented by a graph $G(N, C)$,

where N is the set of routers and C specifies the sets of communication channels. The network topology significantly affects the overall network performance (since it determines the number of intermediate routers i.e. hops from source to destination nodes. Longer route means more latency and energy consumption), reliability (since it specifies the number of alternative paths for avoiding faults or network contention) and cost (number of routers affect area/power cost) [24, 29].

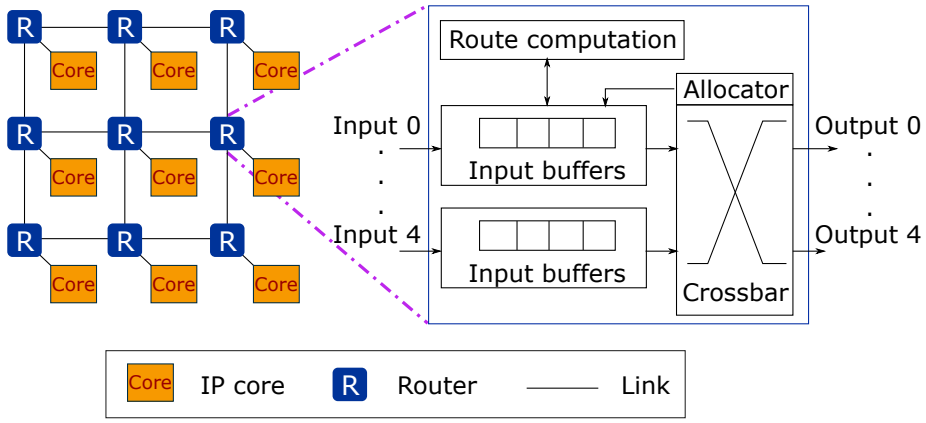


FIGURE 2.1: Mesh topology with input buffered router architecture

Fig. 2.1 shows mesh network work topology and typical input buffered router architecture such as the Hermes router [74]. The router consists of 5 input/output ports i.e. west, south, east, north and local. The local port connects the router to the local core via a network interface, while other ports connect to neighbouring routers. The input buffers are used to store data that cannot proceed to their desired output port due to network contentions. The router employs several strategies (such as routing, switching) in order to route data. These strategies are discussed next.

2.1.1 Routing

When *data* enters the router, the router needs to decide which of the output ports to forward the data. This decision is based on the routing algorithm implemented in the router. The goal of the routing algorithm is to ensure even distribution of the network traffic so as to avoid hotspot regions in the network and also to reduce network contentions, thus enhancing network performance [29]. Routing algorithm can be classified into two i.e. dimension-ordered (DOR) and adaptive routing. In DOR, the paths taken by data is fixed, so data always take the same paths regardless of the network conditions (i.e. it is deterministic in behaviour). Example of DOR is the *XY routing*, which is commonly used in Mesh network topology. In XY routing, data is first routed in the *X-dimension* (if the destination node *X-dimension* is not equal to that of the current node. Otherwise, its takes the *Y-dimension* to the destination), then the data is routed along the *Y-dimension* to the destination node. [12, 24, 29]. XY routing is illustrated in Fig. 2.2a. Deterministic routings are very simple to

implement and use less routing area resource [89]. However, they suffer from performance loss when network contention is high since data cannot take alternative paths to their desired destination.

(A) Deterministic routing (B) Adaptive routing

FIGURE 2.2: Deterministic vs. Adaptive routing

Conversely, data are allowed to take alternative paths to their destinations in adaptive routing. This provides reliability since alternative paths can be utilized to avoid faults or network congestions. Fig. 2.2b shows an example of adaptive routing. This often requires some monitoring of channels and/or buffers to determine when to re-route data. Adaptive routing is more complex to implement compared to deterministic routing. Also, additional care needs to be taken to avoid *deadlock*, *livelock*, and *starvation* of data. Deadlock occurs when data cannot progress because they are occupying network resources and requesting for resources occupied by other data. Thus, forming a cycle of dependencies as shown in Fig. 2.3, where the packet cannot advance to their desired destination.

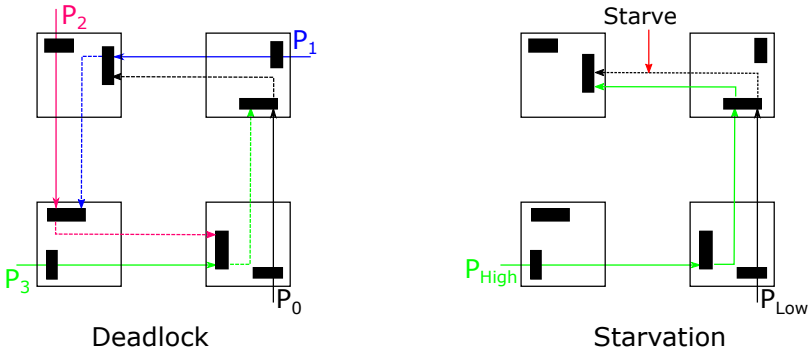


FIGURE 2.3: Deadlock and starvation scenario

Mechanisms such as dropping and re-transmitting data can help avoid deadlocks. However, this introduce additional design complexity and incurs latency overhead. Livelock is a special kind of deadlock where a data keeps traversing the network without ever reaching its destination. In order to avoid livelock, data can be removed from the network after a period of time. An alternative approach would be to revert to deterministic routing after a certain period of time, which will ensure that data will get to their destination. Starvation occurs when data with low priority

are denied of network resources because higher priority data are always utilizing the resources [89]. This is illustrated in Fig. 2.3, a where low priority packet is denied access to resources because of a high priority packet. Implementing mechanisms to avoid these adverse effects introduce additional complexity and area [89]. Routing algorithms can also be classified as source routing and destination tag routing [89]. In source routing, the sender determines all the paths the packet will follow to its destination. This information is typically stored in the packet header. Intermediate nodes read the information and forwards the packet accordingly. Conversely, in destination-tag routing, the packet only carries the routing address. The intermediate nodes take routing decisions, which allows for alternative paths to be used as opposed to source routing.

2.1.2 Flow-control

Flow-control determines the manner in which network resources are allocated. Before discussing the various flow-control it is important to introduce the concept of *message, packets and flits*. A message is the actual data to be transmitted from a source to a destination core. A message is usually broken-down into packets. Each packet has header information used for reconstructing the initial message at the receiving end. Packets belonging to a message can be routed differently to the destination. A packet may be further broken into smaller units known as flit (i.e. flow-control units). Flits of a packet typically follow each other in a *chain-like* fashion.

Bufferless flow-control

In bufferless flow-control (also known as bufferless routing), flits that cannot immediately advance to their desired output ports are either dropped and re-transmitted or deflected to an available output port (as shown in Fig. 2.4) since there are no buffers in the router to temporarily store packets [75]. This leads to significant area and power savings since buffers are eliminated. However, it results to additional energy consumption and higher packet latency at high traffic injection rates due to increase in packet deflections that unnecessarily consumes link bandwidth [75, 101].

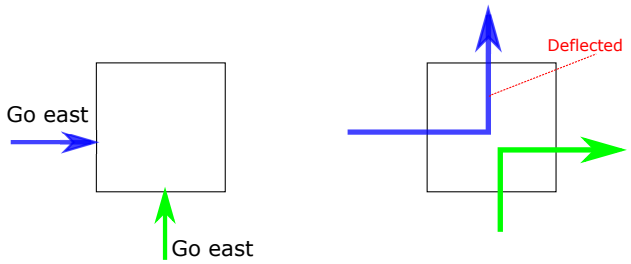


FIGURE 2.4: Bufferless flow-control

Circuit switching

Circuit switching is a *connection-oriented* flow-control, meaning a connection (i.e. circuit) is first established between a source and destination nodes before actual data transfer takes place. Circuit switching is used for providing guaranteed services where system predictability is required. As an example, circuit switching is used in the Æthereal NoC [90] to provided guaranteed packet throughput. Circuit switching provides contention free data-transfer since a network path is completely isolated. However, it leads to reduced link utilization since the same link cannot be used by multiple flows. In addition, it incurs the overhead of setting-up and tearing-down circuits.

Store-and-forward

In store-and-forward flow-control, the router waits until an entire packet is received before transmitting the packet to the next node. Hence, resource (buffers and channels) are allocated to an entire packet. This leads to higher per-packet latency, which makes it unsuitable for delay-critical on-chip networks [12]. Also, store-and-forward requires large buffers (capable of storing an entire packet), which leads to additional area/power overheads associated with large buffers especially with long packets [75].

Virtual-cut-through (VCT)

In order to reduce the delay associated with store-and-forward, VCT flow-control was proposed. Unlike store-and-forward, VCT transmits packets without waiting for the arrival of the entire packet. Thus, per-packet latency is drastically reduced. However, a packet is still transmitted if there exist enough storage to hold the entire packet (i.e. buffers and channels are allocated to an entire packet), which also leads to area overhead and power consumption associated with buffers [24].

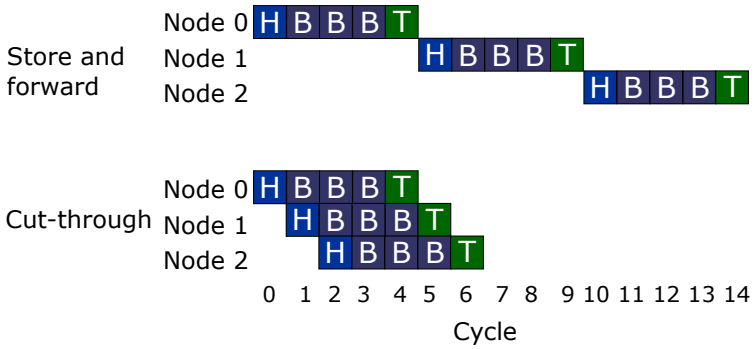


FIGURE 2.5: Store-and-forward vs Cut-through flow-control

Fig. 2.5 illustrate packet transmission for store-and-forward and cut-through flow-controls for 14 clock cycles. Store-and-forward incurs significant delay since an entire packet must be received before the packet is transmitted. Contrary, to store-and-forward, cut-through incurs much lower latency since packet transmission occurs before the entire packet is received.

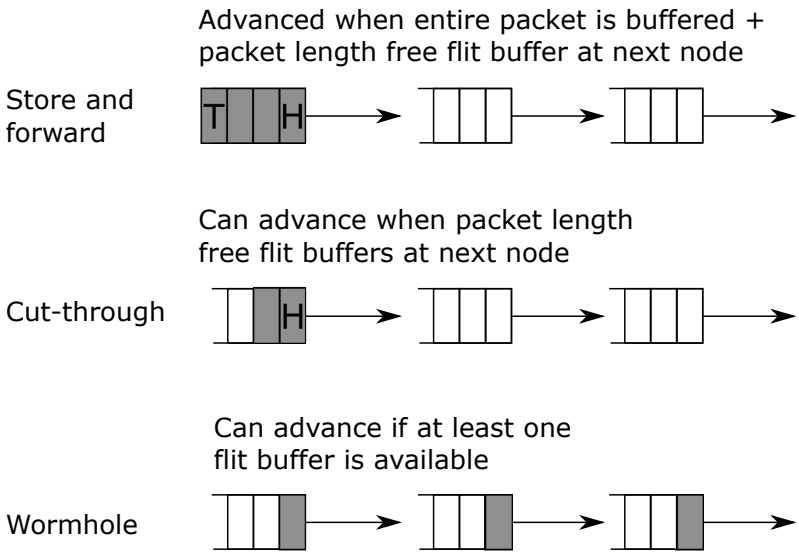
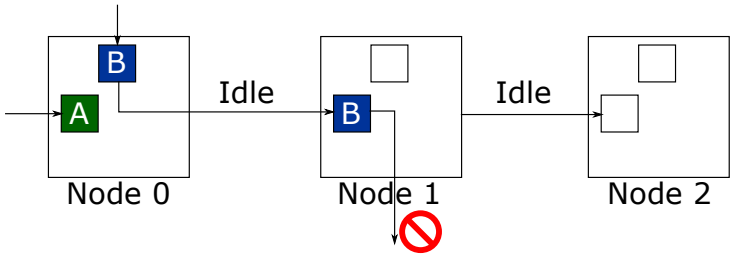


FIGURE 2.6: Store-and-forward, cut-through and wormhole flow-control

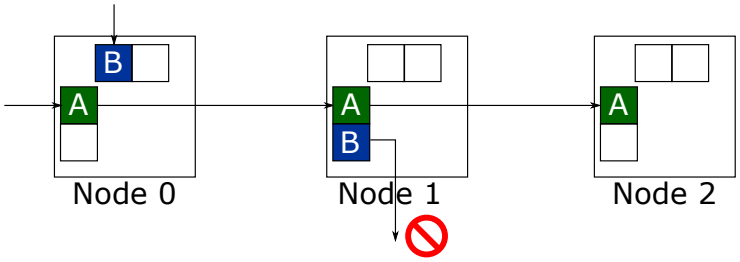
Wormhole

In order to reduce the area/power overheads associated with VCT flow-control, *wormhole* flow-control was proposed. In wormhole flow-control, packets are broken down into flits. Thus, buffers are allocated not to an entire packet, but to individual flits as illustrated in Fig. 2.6. Thus, the number of buffers required in the network is drastically reduce compared to VCT. The head-flit carries the routing information, while other flits follow the head-flit in sequence. If the head-flit is blocked, the other flits stop. Wormhole provides lower latency and allows for better buffer utilization. However, it suffers from *head-on-line (HOL) blocking*, which leads to poor link bandwidth utilization and reduced packet throughput.

Fig. 2.7a shows an example of HOL blocking in wormhole network. A packet *B* using north output of *node 1* cannot advance because no free buffer is available at the receiving end. Packet *A* destined for *node 2* is blocked even though the links between *nodes 1, 2* and *3* are free. Thus, packet *A* needs to wait until packet *B* is completely transmitted.



(A) Head-on-line blocking in wormhole



(B) Avoiding head-on-line block using virtual-channels

FIGURE 2.7: Wormhole versus virtual-channel flow-control

Virtual channel (VC)

In order to overcome HOL blocking in wormhole flow-control, the physical links (or channels) can be multiplex to 2 or more *virtual-channels* [23]. As shown in Fig. 2.7b, the input buffers in the router consist of multiple buffers sharing a single physical channel. Thus, packet *A* can still utilize the network links even though packet *B* is blocked. This leads to improved link usage. VC can also be used for deadlock avoidance. This is achieved by switching to different sets of virtual channels on some turns in order to break possible cyclic dependencies. As an example, VC is used to avoid cyclic dependency in networks using *Dateline*. In Dateline, each node has two virtual channels i.e. *low and high*. Packet switches from *low* to *high* VC when they pass certain point (i.e. the dateline) in order to avoid cyclic-dependencies in the network [24]. VC can also be used to prioritize traffic. As an example, a set of VC is dedicated to guaranteed throughput traffic while others are dedicated to best effort traffic in the MANGO NoC router [13] (see Fig. 3.6). Table 2.1 provides a summary of the flow-control discussed.

TABLE 2.1: Flow-control summary

Flow-control	Properties	Buffer requirement
Bufferless	No buffer in router. Packets are deflected if desired output port is not available or dropped and retransmitted. Leads to poorer performance and higher power at high traffic injection	Lowest
Circuit switching.	Connection between source and destination node is first set up before actual data-transfer. Leads to poor scalability	High
Store-and-forward	Receive entire packet before forwarding. Poor per-packet throughput	Highest
Virtual-cut-through	Can forward part of a packet before receiving the entire packet if receiving buffer has enough storage for the entire packet	High
Wormhole	Packets are broken down into smaller units (i.e. flits). A flit can advance if there is a flit size storage. Suffers from head-on-line blocking	Low
Virtual-channels (VC)	Multiplex physical channels to multiple virtual channels. Can provide QoS, avoid deadlock and Removes HOL blocking	Low

2.1.3 Buffer management

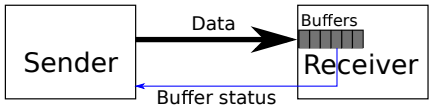


FIGURE 2.8: Buffer management between communicating nodes

Fig. 2.8 shows two communicating nodes i.e. the upstream node (i.e. the sender) and downstream node (the receiver). The upstream node needs to know if the downstream node has enough storage position to store data before attempting to send data. The buffer information (i.e. buffer status in Fig. 2.8) of the downstream node needs to be communicated to the upstream node. This is achieved using some buffer management mechanisms which are discussed next.

In *credit-based* buffer management technique, each upstream node keeps a record of the available buffer positions in the downstream node, typically by means of a

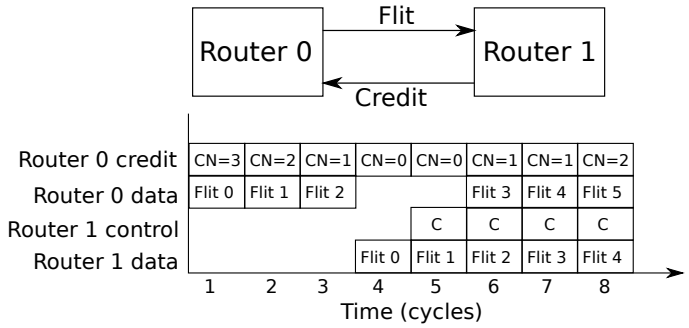


FIGURE 2.9: Credit-based buffer management. CN: credit number

counter (i.e. the credit). The downstream node signals the upstream node to increment the counter when a buffer is freed. The upstream node decrements the counter each time its communicates with the downstream node. This leads to significant signaling of the nodes [24]. Fig. 2.9 shows an example of credit-based buffer management. In this example, Router 0 (i.e. the upstream node) can keep sending data as long as Router 1 (i.e. the downstream node) has free buffer slots to receive data (i.e. credit is not 0).

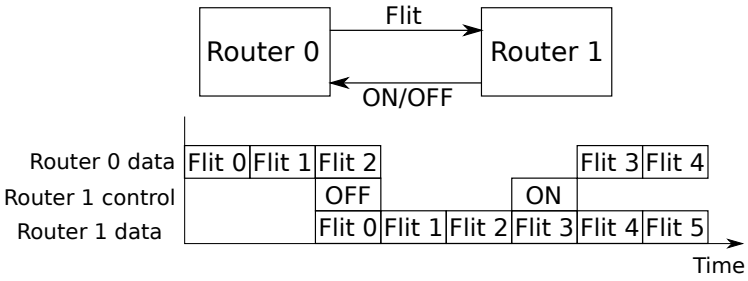


FIGURE 2.10: On/Off buffer management

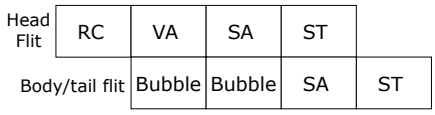
In order to reduce the overhead associated with credit-based buffer management, the *on/off* technique was proposed [24]. In this mode, the downstream node keeps track of its buffering positions. When the number of available buffering positions reaches a certain threshold, it signals *off* to the upstream node to stop sending packets. Similarly, the downstream signals *on* when it can start receiving packets as shown in Fig. 2.10. The *on/off* technique reduces the amount of signaling which saves power since the router switching activity is reduced. However, it requires more buffers and can lead to poor buffer utilization [24].

Router architectures that use *elastic-buffers (EB)* [17, 73] do not have implicit input buffers. Such routers utilize EBs, which are basically buffers (latches or flip-flop) that implement elastic FIFOS. Such routers use handshaking protocol to control dataflows in the router. An example of such protocol is the *Ready/Valid* handshake which is similar to asynchronous *Req/Ack* handshaking protocol [96]. In the Ready/Valid protocol, the upstream node must have *valid* data and the downstream node must

be *ready* to accept the data before a data-transfer transaction can be successful. This is extensively covered in Chapter 5.

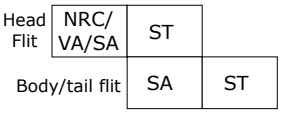
2.1.4 Router pipeline

An input buffered router typically has four pipeline stages i.e. *Route computation (RC), Virtual-channel allocation (VA), Switch allocation (SA) and Switch traversal (ST)*, assuming a router with input virtual channels (VCs) discussed in Subsection 2.1.2. Fig. 2.11a shows the pipeline stages of a typical input buffered router. The head-flit goes through all the stages, while the body/tail flits go through only the SA and ST stages. When the head-flit enters the router, the RC stage is executed to compute the packet output channel in the current router. The head-flit then competes for VCs corresponding to the assigned output channel in the VA stage. Since flits from other VCs shares similar physical channels, a flit must compete for access to the allocated output physical channel. This is done in the SA stage. Finally, a flit is transmitted to the next router in the ST stage. Notice that two bubbles are introduced by the header flit because it performs two more stages as shown in Fig. 2.11a.

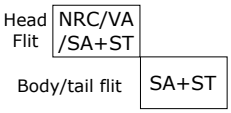


(A) Typical input virtual-channel router pipeline

(B) Router pipeline with look-ahead routing



(C) Speculative router pipeline (look-ahead and speculation)



(D) Speculative router pipeline (single stage/cycle)

FIGURE 2.11: Different types of router pipelines

In order to reduce the number of bubbles introduced by the head-flit and the router pipeline, *look-ahead* routing was proposed. In look-ahead routing, the RC stage of the next router is performed by the current router i.e. one hop in advance. Thus, reducing the total pipeline to three stages with only a bubble as shown in Fig. 2.11b [42]. In order to further decrease the router pipeline, *speculative routers* performs the VA and SA stages in parallel. This is shown in Fig. 2.11c, where look-ahead and speculation are combined to reduce the pipeline stages. Speculative routers partially remove the bubbles introduced by the VA stage. The speculation might be successful when network contention is low. However, if the VA fails, the allocated output channels becomes un-utilized. Recent works [46, 21, 91] propose to follow an *aggressive speculation* approach shown in Fig. 2.11d, where VA, SA and ST

are performed in parallel. Thus, realizing a single-cycle router. However, it can lead to un-utilized resource and retries in cases of failures.

2.1.5 Network traffic and Performance metric

For the purpose of benchmarking the NoC performance and power consumption against the literature, this thesis considers both well-known *experimental* traffics and also *traffic* from real-world applications.

Network traffic

NoC traffic can be typically classified as either synthetic or application traffic. Application traffics are traces from real-life application workloads. Examples include telecom, networking and consumer applications from the E3S benchmark suite [27]. On the other hand, synthetic traffics are experimental traffics used for benchmarking the communication architecture and they attempt to mimic real-world application traffics. Synthetic traffic can be uniform (also known as regular) or non-uniform (also known as irregular).

An example of a regular traffic is the *random (or also called uniform)* traffic pattern, where each node communicates with every other node with equal sending probability. An example of irregular traffic pattern is *transpose*, where each node communicates with its transpose. As an example, consider a 2x2 mesh network topology with 4 nodes with *(x,y)* coordinates [(0, 0), (1, 0), (0, 1), (1, 1)]. Node (0,0) communicates with its transpose i.e. node(1, 1) and vice versa. *Hotspot* is another example of an irregular traffic pattern, where all the nodes communicate with one node *i.e. the hotspot node* in the network. Irregular traffic pattern creates more network contention compared to a regular traffic pattern, which creates communication bottleneck in the network. Techniques such as adaptive routing can help mitigate performance loss caused by irregular traffic pattern. However, as discussed earlier in this chapter, adaptive routing is generally more complex to implement [89].

Network performance metrics

Two important performance metrics considered in this thesis are *latency* and *throughput*. Latency is defined as the time lapse between when a packet is to be transmitted from the source node and received at the destination node. The *clock cycles* is considered as the unit of latency for evaluating synchronous networks, while *nanoseconds* is considered for evaluating clockless networks. Instead of a single latency, the average latency of the entire network is normally considered [24]. The latency depends on network contention and the distance between a source and a destination node. Hence, high network contention and/or longer distance will produce more latency.

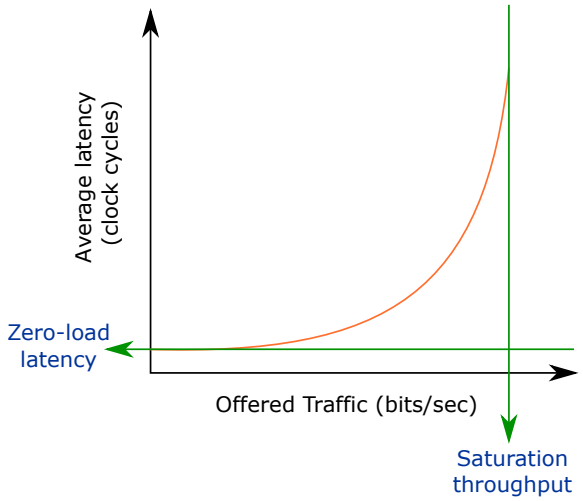


FIGURE 2.12: Latency vs. offered traffic in networks

Fig. 2.12 shows the latency vs. offered load (also called *injection rates*) of an interconnect network. The zero-load latency is the optimal delay of the network, where the source nodes communicate with the destination nodes without any contention. The *offered throughput* (also refers to as *offered traffic* or *offered load*) refers to the amount of traffic sent to the network per unit time. As the nodes keep injecting traffic into the network, it reaches a *saturation throughput*, which defines the maximum amount of traffic that can be sustained by the network [24].

2.2 Networks-on-Chip (NoCs) design challenges

As stated in Chapter 1, although NoC has become the pervasive on-chip interconnect for manycores systems, there exist numerous challenges with its design. This section provides a more detail description of some of the challenges facing NoC designs.

2.2.1 Power consumption

Power consumption is one of the biggest challenges facing today's NoC interconnect. Today's NoC consumes a significant portion of the total system power. The NoC consumes 28% of the total chip power in the Intel Terascale 80-chip chip [48], 36% for MIT RAW [98] and 10% in the Intel 48-core SCC chip [14]. Current power management techniques such as clock-gating or sleep-states incur wake-up latency, which leads to performance loss [15]. Since a large percentage of the total power consumption is associated with NoC router buffers, techniques to remove router buffers in order to save power exist. However, they often lead to higher energy consumption and performance degradation under high traffic. Thus, techniques to reduce power consumed by NoC without compromising performance are desirable [75].

2.2.2 Quality-of-Service

Quality-of-Service (QoS) is an important aspect of NoC design. QoS generally refers to proving system predictability or service guarantees. Generally, NoC provides two main service types: guaranteed services (GS) and best-effort (BE). Thus the traffic in NoCs can either be one of the two types. Providing GS in NoC typically requires connection establishment (core-to-core resource reservation) before actual data transfer i.e. connection oriented. As stated earlier, circuit switching is a common way of providing QoS in NoC, where a connection is first set up between source and destination nodes before actual data-transfer takes place. As discussed in Chapter 1, circuit switching leads to poor scalability since router area growth is proportional to the number of required connections [69], and also additional latency in managing circuits. Virtual-channels (VCs) have been use to provide QoS support in NoCs. This is normally achieved through allocating priority to the physical channel bandwidth. As an example, a set of VC is dedicated to guaranteed throughput traffic while others are dedicated to best effort traffic in the MANGO NoC router [13] (see Fig. 3.6). Most NoCs provide only BE service as they only ensure communication without any performance guarantees. Also, BE NoCs are generally simpler to implement compared to GS NoCs. However, GS service is vital for application with strict constraints. Examples includes hard real-time in automotive and space control systems where deadlines need to be met. Most NoC combines both GS and BE traffic which in many cases requires separate logic to handle both traffic types. This leads to increase in NoC design complexity. Thus, techniques for providing QoS without latency and design complexity is an important goal in NoC design [79].

2.2.3 Latency

NoC is required to provide low latency under-stringent power, area and delay constraints. Therefore, minimizing delay is a crucial aspect of NoC design, especially for cache coherence traffic where the NoC is expected to provide faster remote data read in cases of cache miss. A typical NoC router has a 4 or 5 stages pipeline which incurs higher latency. Techniques such as speculation have been proposed to reduce the pipeline to 1 or 2 stages. However, more study is still required to improve the accuracy and efficiency of such design techniques [79]. As stated earlier, NoC provides alternative paths between a source and destination IP. Thus, data can be routed using alternative paths when congestion increases, which help reduce data latency. However, data may be delivered in a different order in which they are sent (i.e. *out-of-data delivery*. Out-of-order delivery often requires large buffers at the receiving end for buffering and re-arranging data. This, in turn, leads to performance loss and

introduces additional area/power overhead associated with buffers. This limits the use of adaptive routing in NoCs [79].

2.2.4 Synchronization

Globally Asynchronous Locally Synchronous (GALS) paradigm allows synchronous cores to communicate asynchronously. GALS has become the default design style due to difficulty in distributing a single synchronized high frequency clocks chip-wide in purely synchronous systems [107]. GALS is suited for power optimization technique such as *DVFS* [107]. It allows heterogeneous cores to operate at their maximum potential, instead of constraining all the cores to operate at a specific frequency. GALS paradigm prevents cores with low frequency from slowing down cores capable of operating at a higher frequency. Synchronous NoCs, with each router operating at a different clock frequency, has been proposed for GALS. In order to ensure the correct operation of the system, data crossing different clock domains need to be synchronized. Hence, the need for synchronization interfaces between the routers themselves and the IP cores. Synchronization interfaces incur additional area, power and latency overhead. In order to reduce the overhead, asynchronous NoCs has been proposed for GALS. Asynchronous NoCs reduce the number of synchronization interfaces to only the interface between each router and its local IP core. However, asynchronous NoCs incur large area-overhead and design complexity compared to synchronously designed NoCs. Although, several proposals for synchronization interface such as asynchronous FIFOs [56], bi-synchronous FIFOs [82] exist, further research is still needed to provide more power efficient and low latency interfaces.

2.2.5 Traffic Variability and Network topology

Different types of traffics can be delivered from different applications or from different phases of the same application. The traffic types can include messages that differ in length, types (data, synchronization), patterns (uniform, non-uniform) and injection rates (steady or bursty) [79]. Designing NoC to cope with the different types of traffic is certainly an import consideration in NoC design. The challenging aspect is in devising NoC router architectures that provide supports for some or all of the traffic types without compromising efficiency in terms of power/performance [79].

Network topology determines how the nodes are connected in the network. Although several NoC topologies exist in the literature, only very few have been implemented in manufactured chips. For example, Intel Xeon Phi series [88] uses ring topology. NoC topology affects performance, hence careful consideration needs to be given to select a suitable topology depending on the application. Authors in [37]

studied the impacts of NoC topologies on MPEG4 video application. Further investigation is needed to determine suitable NoC topology for heterogeneous systems capable of running different applications.

2.3 Resource sharing in NoCs

This section identifies the problem of buffer underutilization, which leads to performance degradation in NoCs. The benefits and challenges of resource sharing is presented. The section ends with a discussion on inherent resource sharing in NoCs and proposes some desirable mechanisms for inherent resource sharing.

2.3.1 Underutilization of NoC resources

As stated earlier, an input buffered router is composed of input and/or output buffers which are used to temporarily store packets that cannot advance to their desired output ports due to network contentions. Storing packet in buffers is beneficial as it ensures that the network links bandwidth are available for use by other packets. On the other hand, studies have shown that buffers are the most expensive router resource in terms of area/power overheads [102, 38, 101].

Although buffers are very expensive, studies have shown that they are often unutilized (i.e. idle or underutilized) especially when executing applications with non-uniform traffic pattern and/or bursty behaviours. The reason being that typical input buffered routers allocate a set of buffers to their input and/or output ports and these buffers can only be exploited by data-flows using the ports. Unutilized buffers lead to significant network performance degradation [102, 101].

The buffer activities were recorded when simulating an 8x8 mesh network topology (consisting of typical input buffered routers), for *uniform, and non-uniform (transpose and bit-compliment)* traffic patterns for 30,000 clock cycles [102]. It was observed that for uniform traffic pattern, only about 10% of the total buffers in the network were always empty. Hence, most buffers are utilized for uniform traffic pattern. On the other hand, it was observed that 47.5% and 45% of the total buffers in the network were always idle for transpose and bit-compliment traffic patterns respectively [102]. Thus, leading to poor performance especially for applications with non-uniform patterns or bursty behaviours.

(A) Conventional 2D mesh (B) Concentrated mesh with C = 8

FIGURE 2.13: 64-node on-chip network using (a) 2D mesh and (b) concentrated mesh. C: concentration

2.3.2 Benefits of resource sharing in NoCs

Area and power reduction

Network resources can be shared in order to exploit their utilization for improving overall network performance and/or to reduce area overhead and power consumption. Resource sharing can be applied at different granularity level. One possibility is to share the routers and channels among several processing cores, in order to reduce the cost of an on-chip network. An example is the *so-called concentration networks* [58, 93], which has been used to drastically reduce the cost of an on-chip network. As an example of area reduction using a concentration network, consider an *N-node* mesh network topology. This network requires N routers since each router is attached to a core. The network can be concentrated to reduce as much as possible the area overhead associated with routers in the network. A concentration of 8 will drastically reduce the number of routers to *N/8* [60] as shown in Fig. 2.13. Similarly, the X-Mesh network, where each router is connected to four cores for area reduction and performance gain, has been proposed [93, 105].

Performance improvement

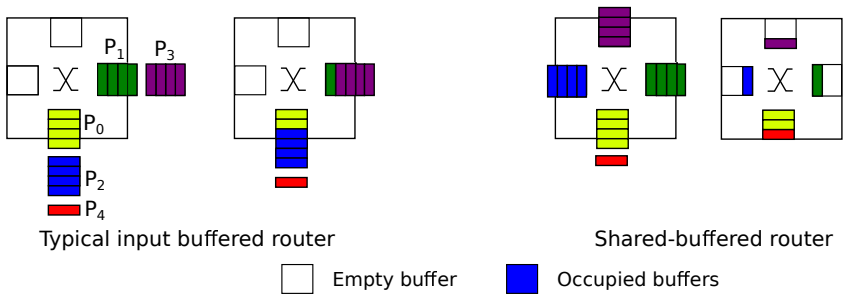


FIGURE 2.14: Typical input buffered vs shared-buffered routers

Contrary to concentration networks which mainly target area/power reductions, the buffers in the routers can be shared for improving network performance in terms of network saturation throughput. This is the case in *shared-buffer* routers, where the

router buffers are shared by multiple input ports. Fig. 2.14 shows cases of data-flow in typical input buffered (without resource sharing) and shared-buffered routers. As illustrated in Fig. 2.14, 3 packets (2 at the south input port and 1 at the east input port) are waiting to be injected in the router. The packets cannot be injected because the receiving buffer storage is full although other buffers are idle. The packets wait until there is room at the receiving buffer before they can be injected into the router. However, for the case of shared-buffered router, the idle buffers can be utilized for performance gain.

Architect.	Typical wormhole	DLABS_1+1	DLABS_2+2	DLABS_2+2 _duallink
random	10.0%	1.0%	0.9%	0.9%
transpose	47.5%	16.2%	16.9%	16.9%
bit-comp.	45.0%	8.3%	9.8%	9.8%

FIGURE 2.15: % of idle buffers for different traffic patterns [102]

Fig. 2.15 compares the percentage of idle buffers for different traffic patterns in typical input buffered router (i.e. wormhole in Fig. 2.15) and different versions of a similar shared buffered routers (i.e. DLABS [102]). It is observed that shared-buffered routers drastically reduce the number of idle buffers in the router for all the considered traffic patterns. In terms of performance DLABS [102] reported network performance improvement of 4%, 26% and 64% over typical input buffered router for uniform, transpose and bit-comp traffic patterns respectively. As discussed in Subsection 2.1.1, typical input buffered router shares the router link bandwidth among several routers. However, head-on-line blocking result in wormhole networks, which leads to link underutilization and poor performance. In order to maximize link usage, virtual channels (VC) flow-control was proposed. VC ensures idle network-links are utilized, which brings improved performance over wormhole.

2.3.3 Some resource sharing challenges

Although sharing network resources brings a number of benefits, implementing such networks is often challenging compared to network without resource sharing. This section discusses two key challenges with resource sharing, which motivates some of the contributions in this thesis.

Deadlock freeness/avoidance

One of the main challenges with designing communication architecture with shared resources is deadlock-freeness. This is because resource sharing often introduces

cyclic-dependencies, which can lead to deadlock [31, 24]. Although, *roundabout-inspired* routers such as Rotary [2] provides inherent resource utilization for improving network performance, the deadlock problem is exacerbated in such routers due to their *ring-like* topology. Deadlocks can typically be avoided using either VCT-based or wormhole based flow-control techniques. Existing VCT based techniques [24, 18, 20] introduce additional area/power overhead and higher packet injection latency. On the other hand, existing wormhole based techniques [19, 66] reduces the router area overhead but also lead to higher packet injection latency, places a restriction on the number of input buffers and often introduce additional design complexity compared to *true wormhole flow-control*. The different deadlock-avoidance techniques are covered in greater depth in Chapter 3

Design complexity

Designing NoCs with resource sharing often introduce additional design complexities particularly in the NoC router architecture. These complexities can, in turn, introduce additional area/power and/or performance overheads.

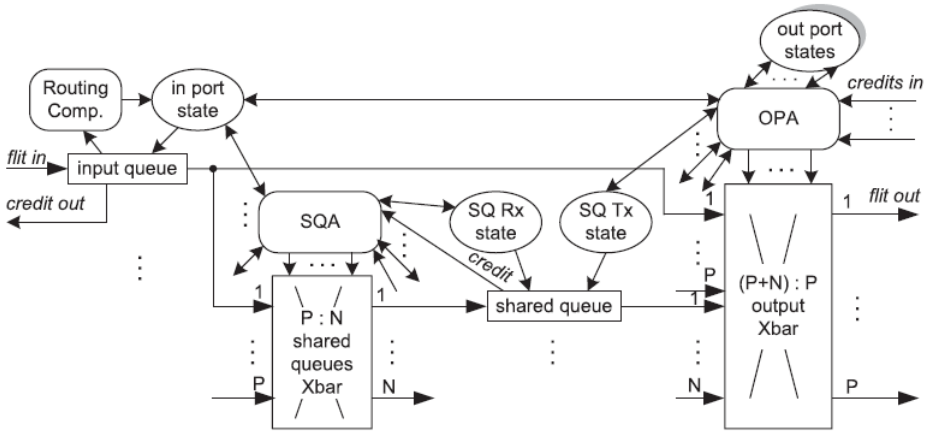


FIGURE 2.16: RoShaQ [101] router microarchitecture

The RoShaQ [101] is a router architecture with *shared-queues* located between the input and output router ports as shown in 2.16. The shared-queues can be exploited by packets from several router input ports. Roshaq provides performance improvements over typical input buffered routers, especially for non-uniform traffic pattern. However, contrary to typical input buffer architecture, Roshaq is more complex. It requires one additional crossbar that is used to control access to the output port from the shared queues. The additional crossbar introduces additional area/power overhead compared to typical input buffer router. In addition, router pipeline incurs three additional clock-cycles in cases of heavy load. Authors [95] proposed a distributed router with shared output queue, which offers improved network saturation threshold compared to typical input buffer router. However, similar to the

Roshaq, this router also needs two crossbars, which leads to increase area and power overheads.

Router architectures capable of sharing the input port buffers among several buffers have been proposed [44, 61]. These routers avoid the area and power overhead associated with an additional crossbar and arbitration complexity by using adaptive routing [44] or by sharing buffers between two neighbouring input ports [61]. In any case, these router requires additional control logic to implement the routing and/or for allocating idle buffers to incoming packets, which incurs additional area and power overheads.

As stated earlier, concentration networks drastically reduce the network area and power overhead since the number of routers in the network are reduced. However, they can lead to additional complexity (bigger crossbars, allocators) due to additional ports required to implement concentration. In addition, increasing number of ports can lead to higher contention which can reduce network performance [60, 93].

2.3.4 Motivation for roundabout-inspired router in NoCs

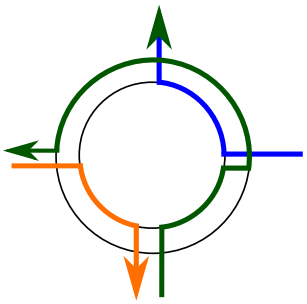


FIGURE 2.17: Data-flows in a generic roundabout-inspired router

Among the possible router architectures supporting resource sharing, roundabout-inspired or ring architectures are regarded as promising. The reason being that they do not rely on costly crossbars (that increases with additional ports) and use decentralized arbitration which allows for higher data-rates. Such routers are based on the principles of real-life traffic roundabout, where arbitration/control is done by cars (e.g. drivers or packets in NoC terms) and not a centralized unit (traffic light or crossbar control in NoC terms) as illustrated in Fig. 2.17. Such router architectures provides inherent resource sharing, which allows the available buffering resources to be better exploited for performance gains. They consist of lanes or rings that are shared by multiple input ports for improved router resource utilization as shown conceptually in Fig. 2.17.

(A) Typical shared buffer router (B) Roundabout-inspired router

FIGURE 2.18: Data-flows in typical shared buffer and Roundabout-inspired router architectures. Shaded box: buffers

Improved network performance

Typical shared-buffer routers allow data-flows to utilize alternative idle buffers (of other input ports) if their desired input port buffer is full. This brings performance improvement in cases of non-uniform traffic, where network contention is high. However, it can lead to buffer underutilization especially in cases of light network load (i.e. little or no network contentions) and/or few input ports receiving data-flows. Fig. 2.18 shows a scenario where the network load is light and only two input ports are receiving data-flows. It is observed that idle buffers exist in the router. Utilizing those idle buffers can help improve packet throughput. Contrary to typical shared-buffer routers, the network resources are always shared, leading to improved packet throughput in roundabout-inspired routers. This is illustrated in Fig. 2.18b, where the router buffering resources are well utilized for improving network performance. Thus, roundabout-inspired routers provide better buffer resource usage for improving network performance under varying traffic characteristics.

Enhanced scalability

In terms of scalability (i.e. introducing additional ports in the router), roundabout-inspired router provides enhanced scalability due to its distributed architecture. The distributed nature allows for additional ports to be added without an increase in arbitration complexity and significant area and power overheads since it does not use implicit crossbar(s) and input buffers. Most typical shared buffer routers use a centralized architecture with one [61, 91] or two [95, 101] crossbars, which introduce arbitration complexity with increasing number of ports. Additional ports also introduce additional area/power overhead since typical crossbar cost grows as a square of input and output ports count [71]. Roundabout-inspired router architecture allows for higher data-rates since output port arbitration is localized contrary to centralized arbitration used in most typical shared-buffer routers.

2.3.5 Desirable design mechanisms

An example of a roundabout-inspired router is the Rotary [2], which consists of two rings that are shared by all the input ports, thereby providing enhanced resource utilization compared to typical shared-buffered routers. As stated earlier, roundabout-inspired routers are deadlock-prone due to their ring-like architectures which introduce cyclic dependencies among shared resource. In order to avoid deadlock, the Rotary [2] router utilizes a combined VCT and bubble flow-control. Although this allows for deadlock-free data-flow, it introduces significant area and power overhead associated with VCT since buffer allocation is carried out at packet-level. Also, a packet can only advance if there's enough buffer storage for the entire packet, which leads to higher *in-transit* packet latency. The bubble flow-control requires two empty packet-sized buffers before injecting packet. This can lead to higher per-packet injection latency since a packet injection into the ring is delayed until the minimum free-buffer requirement is met.

As stated earlier, the Rotary router [2] uses combined VCT and bubble flow-control (i.e. local bubble scheme), which leads to poor buffer utilization especially for cache-coherent traffics. The reason being that the majority of such traffics are short packets (typically single flit packet). The implication is that short packets must also be regarded as long packets (for coping with variable sized packets). This, in turn, leads to poor buffer utilization [3]. The Rotary router [2] allows packets to make multiple turns in the ring (if the desired output is not available) before forcing the packet to use any available output port. This can lead to significant dynamic power consumption especially under high traffic injection rates.

In order to reduce the area/power overheads associated with VCT/bubble, router architectures using flit-based flow-controls such as wormhole and virtual-channels are desired. The reason being that such flow-controls use fewer buffers since buffers are allocated to flits instead of packets. This, in turn, leads to low injection and in-transit packet latency. In addition, such flow-controls lead to better buffer utilization for traffics, where the majority of packets are short. In order to further reduce dynamic power consumption, roundabout-inspired architecture should reduce the number of turns made by packets in the ring.

2.4 Summary

It is widely admitted that NoC provides better scalability and faster data-transfer compared to traditional bus/crossbar interconnects. Therefore, it is more suitable for multi/manycore computing systems. Sharing network resources leads to improved resource utilization, which brings about performance, area/power benefits.

However, networks with resource sharing are more prone to deadlocks and are more complex to design.

NoC routers can be shared among several IP cores, which leads to a significant area and power reduction since the total number of routers required in the network is drastically reduced. However, it can lead to performance loss due to increase in network contentions. In addition, sharing routers among several cores introduces additional design complexity in terms of crossbar cost.

In order to reduce performance loss, routers buffers can be shared among several input ports. This leads to performance improvement but can also introduce additional arbitration complexity and crossbar cost. Roundabout inspired routers are particularly attractive as they provide inherent resource sharing. They are however prone to deadlocks due to their *ring-like* architectures. An existing work [2] uses combined VCT/bubble flow-control to avoid deadlock. This, in turn, introduces additional power/area overheads since VCT requires large buffers. In addition, it can lead to higher packet injection latency.

In order to reduce area/power cost and reduce packet injection latency, wormhole flow-control is suitable for a roundabout-inspired router. However, it is even more challenging to realize roundabout-inspired router architecture using wormhole since wormhole creates additional resource dependencies. One of the major goals of this thesis is to research novel router architecture that removes the limitations of existing solutions.

Chapter 3

State of the art in NoC routers

«"Success is when opportunity meets preparation."»

Anon

Contents

3.1 Introduction

Network-on-Chip can implement buffered or bufferless flow-control. The choice of flow-control, in turn, affects the overall network performance, on-chip area overhead and power consumption. Buffered flow-control includes store-and-forward, virtual-cut-through (VCT), wormhole and virtual channel (VC) as discussed in Chapter 2. However, only VCT, wormhole and VC based routers are discussed here since they are generally preferable in terms of performance and power consumption.

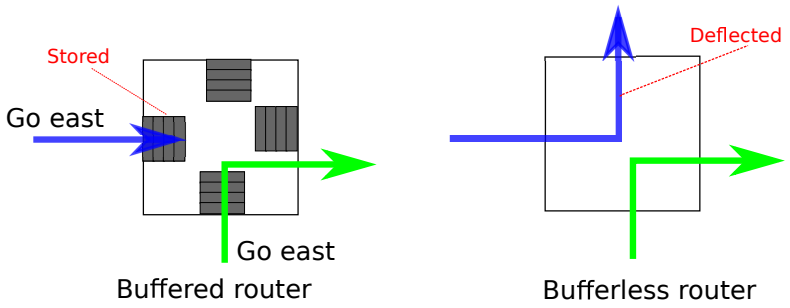


FIGURE 3.1: Conceptual buffered vs bufferless router

In buffered flow-control, the routers contain buffers that are temporarily used to store packets that cannot advance to their output ports due to network contention. This is shown in Fig. 3.1. However, these buffers consume a significant amount of

the router total area and power consumption and are often unutilized for certain traffic characteristics [102, 101].

On the other hand, bufferless flow-control/routing advocate completely removing buffers from the router in order to save power/area. Example of data-flow in a bufferless router is shown in Fig. 3.1. However, these routers suffer performance loss and higher dynamic power consumption under high loads due to increase in packet deflections, which consume the router link bandwidth. In order to reduce performance loss and power consumption caused by packet deflections, several techniques to reduce packet deflections have been proposed. Other works suggest a hybrid solution of adding small buffers (i.e. min buffer) in the router to reduce packet deflections.

The remainder of this chapter is as follows: VCT, wormhole and VC based routers are discussed in section 3.2. Bufferless routers are discussed in Section 3.3, while section 3.4 discusses minimum buffered routers. Section 3.5 discussed shared buffer routers.

TABLE 3.1: State of the art in Network-on-Chip. T: topology

NoC	Topology	Routing	Flow-control	QoS
Kim [57]	2D-Mesh	Adaptive	VCT	No
Hermes [74]	2D-Mesh	DOR - XY	Wormhole	No
Æthereal [28]	N/A	Contention free	Circuit switching	Yes
DyAD [49]	2D-Mesh	Combined	Wormhole	No
Hermes [70]	2D-Mesh	Adaptive/DOR	Wormhole	No
MANGO [13]	2D-Mesh	Deterministic	VC	Yes
BLESS [75]	2D-Mesh	Deflection	Bufferless	No
DeC [106]	2D-Mesh	Deflection	Bufferless	No
SCARAB [46]	2D-Mesh	Drop/re-transmit	Bufferless	No
CHIPPER [39]	2D-Mesh	Drop/re-transmit	Bufferless	No
Oxman [80]	2D-Mesh	Deflection	Bufferless	No
MinBD [38]	2D-Mesh	Deflection	Pseudo bufferless	No
DeBAR [55]	2D-Mesh	Deflection	Pseudo bufferless	No
HiPAD [94]	2D-Mesh	Deflection	Pseudo bufferless	No
AFC [54]	2D-Mesh	Adaptive	Bufferless/Buffered	No
RoShaQ15 [101]	2D-Mesh	DOR	VC	No
Soteriou [95]	2D-Mesh	DOR - XY	VC	No
DPSB [44]	2D-Mesh	Partial adaptive	VC	No
DLABS router [102]	2D-Mesh	DOR	wormhole	No
Khalid [61]	2D-Mesh	DOR	VC	No
FlexNoC [91]	2D-Mesh	DOR	VC	No
Rotary [2]	T. Agnostic	Adaptive	LBS	No
This work	**Various**	**Adaptive/DOR**	**Wormhole**	No

3.2 Input buffered NoC routers

This section discusses three different flow-controls used in typical input buffered routers. They include virtual-cut-through, wormhole and virtual channel. Router architectures employing these flow-controls are also presented in this section.

3.2.1 Virtual-cut-through based routers

Although Virtual-cut-through (VCT) flow-control allows a packet to transmission to begin before receiving the entire packet, the allocation of network resources (e.g. buffers and channel bandwidth) is carried out per packet basis as discussed in Chapter 2. The implication of this is that packets must reserve enough resource to be able to make forward progress. VCT flow-control places huge demands on silicon area and power consumption due to large buffers. Hence, it's not suitable for on-chip interconnects with tight area and power budgets [84].

An adaptive routing algorithm has been proposed for VCT based router [57] in order to improve network performance without significantly compromising area overhead. In the proposed router, a packet requests for output channel is granted if the channel is free. Hence, if the desired output channel is currently being used by a different packet, a *second choice* output channel is requested. The process continues until an output channel is granted or all permitted output channels are exhausted. In terms of performance, the proposed router achieves higher network saturation throughput compared to wormhole routing. However, this performance improvement comes with area overhead associated with large buffers. Although authors did not present result comparing power consumed in the proposed router and wormhole, the proposed router is expected to consume more power associated with buffers.

3.2.2 Wormhole router

The wormhole flow-control is considered the most popular switching technique due to its lower hardware requirement and performance. Wormhole divides packets into smaller units known as flits (flow-control digits). The first flit (known as the head-flit) of a packet usually contains the routing information (destination address) and responsible for reserving network resources as discussed in Chapter 2. Unlike, VCT which requires buffers to be large enough to accommodate an entire packet, wormhole flow-control can use a single flit-size buffer. Hence, wormhole flow-control is more suitable for an on-chip network with stringent power and area budget compared to VCT. Some state-of-the-art wormhole based routers are presented next.

Fig. 3.2 shows the architecture of the Hermes router [74]. It consists of communication ports, buffers and control logic. The router consists of five input/output

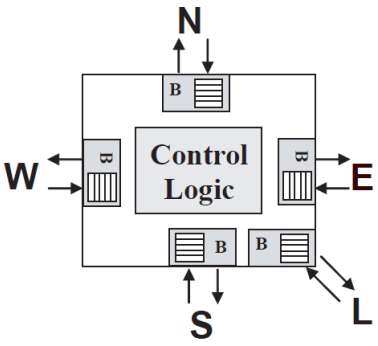


FIGURE 3.2: Hermes router architecture [74]

ports (i.e. west, south, east, north and local). The local port connects the router to its local core, while the other ports are connected to neighbouring routers. The router consists of buffers that are used to store packets that cannot advance to their desired output ports, while the routing and arbitration unit is handled in the control logic block. When multiple flits (head-flits) enter the router, one of the flits can be processed in a given cycle. The arbitration unit is responsible for selecting the flit to be processed. After selecting a flit, the routing unit executes the *XY-routing function* to connect the input port data to the correct output port data. The connection remains active until all the flits of the packet are completely transmitted.

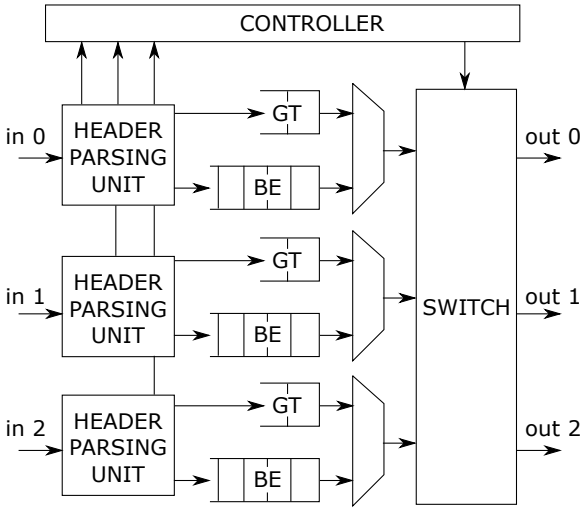


FIGURE 3.3: Æthereal router architecture [28]

The Æthereal [28] router architecture is shown in Fig. 3.3. It provides guaranteed services (GS) in terms of throughput and latency. Thereby, offering predictability of the system. In order to provide guaranteed services, uncertainties in data transfer must be eliminated. The Æthereal router achieves this using a contention-free routing, which is based on *time-division multiplexed circuit-approach*. Basically, circuits (logical connections between source and destination routers i.e. slots) are first setup in the network before the actual data transfer takes place. This ensures that data

transfer occurs without any contention, thereby offering guaranteed packet throughput and latency (this in Æthereal term is known as guaranteed throughput (GT)). In order to avoid output ports contentions, the slots are reserved in such a way that only one GT data transfer is scheduled on an output port of the router or network interface. Since Æthereal combines BE and GS, the BE data is transferred on slots not reserved by the GT traffic or on slots which are reserved but not used. In Æthereal router, the head-flit carries information about the traffic type (i.e. GT or BE as shown in Fig. 3.3. When a flit arrives at the router, it is received by the *header parsing unit*. The unit moves to flit to either the GT buffer or the BE buffer depending on the traffic type. The controller is then notified of the presence of flits. The flits are scheduled for transfer in the next cycle by the controller.

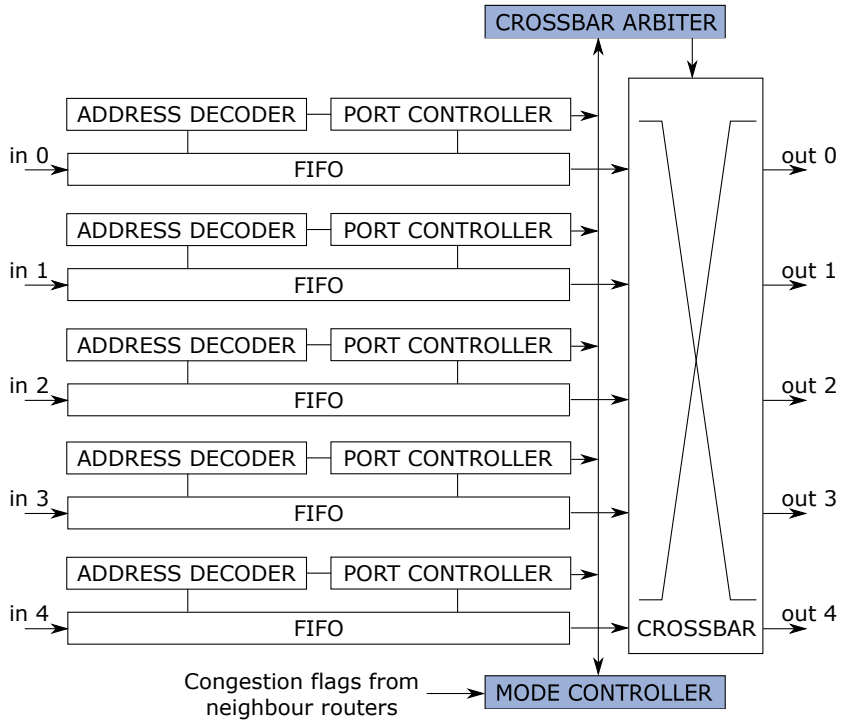


FIGURE 3.4: DyAD router architecture

The DyAD [49] router uses both deterministic and adaptive routing algorithms to route packet in the network. The dimension-ordered (XY routing) is used when network contention is low, while the router switches to the adaptive mode when network contention increases. In order to avoid deadlocks which may be caused by packets waiting for each other in a cycle, the DyAD router employs *old-even* minimal adaptive routing. Generally, packets can take up to 8 turns in a mesh network topology. Four turns are supported in the clockwise direction, while the other four turns are for the counter clockwise direction. Each of the four turns combined to form a cycle with leads to a deadlock situation. Deadlock can be avoided if at least one turn is disallowed in both clockwise and counter clockwise direction [24]. This

is covered extensively in Chapter 4. Basically, the old-even routing forbids certain turns in the old and even column of the network for avoiding deadlock. Fig. 3.4 shows the DyAD router architecture. Each input port has its own FIFO used to temporarily store flits. When a head flit is received, the *address decoder* computes the packet output port and informs the port controller. The port controller decides which of the output ports (more than one output port may be available since the router uses adaptive routing) to forward the packet. The choice is based on the number of available storage positions in the downstream router input FIFO. The port controller then informs the crossbar arbiter to create a connection between the input and the output port for data transfer. DyAD provides improved performance compared to deterministic purely adaptive routers and with a small area overhead. The router is expected to consume more power compared to wormhole routing based on deterministic routing algorithm. This may be due to dynamic energy incurred when switching between deterministic and adaptive modes.

3.2.3 Virtual-channel router

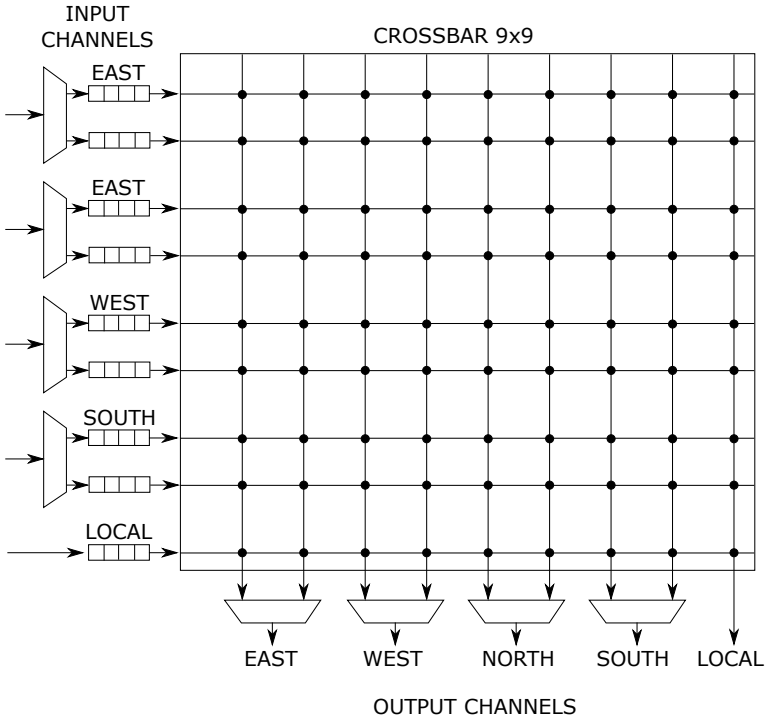


FIGURE 3.5: Hermes router with two VCs

Although wormhole flow-control leads to less area overhead compared to VCT, it suffers from lower network saturation throughput caused by blocked flits in the network which spans across several routers. As discussed in Chapter 2, head-online (HOL) blocking reduces router link utilization which leads to poor link resource utilization. In order to alleviate the performance loss and provide QoS support,

the physical channels can be multiplexed into multiple virtual channels (VC) [23] as discussed in Chapter 2. On the other hand, VC introduces additional arbitration complexity due to VC allocation and area overhead. Here, three virtual-channels with distinct properties are reviewed.

In order to improve network performance, the default Hermes router [74] described in Subsection 3.2.2 was extended to support virtual channels (VCs) [70]. Fig. 3.5 shows the Hermes VC router architecture which is similar to the default Hermes [74] except that the physical channels have been multiplex into two VCs except for the local port. One can readily observe that VC introduce additional area overhead associated with larger crossbar size. As an example, the Hermes router without VC requires 5x5 crossbar, while Hermes VC router, shown in Fig. 3.5 requires 9x9 crossbar. Also, it is important to note that the crossbar size grows quadratically with the number of VCs in the Hermes VC architecture. This is reflected in the results presented by the authors [74].

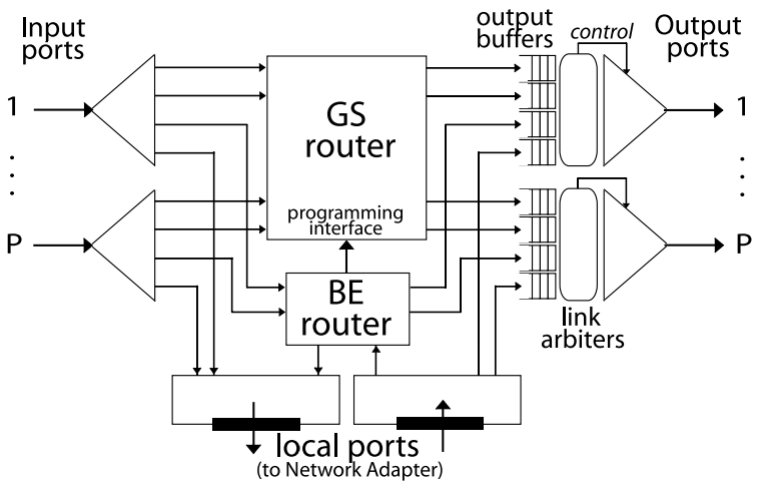


FIGURE 3.6: MANGO router architecture

The Message-passing Asynchronous NoC providing Guaranteed services through OCP interfaces (MANGO) router [13] is a clockless router architecture that provides both best-effort (BE) and guaranteed service (GS). Fig. 3.6 shows the router architecture, which is composed of two sub-router components (i.e. BE and GS routers). One for handling BE traffics and the other for handling GS traffics. In the MANGO router, a subset of the total VCs is dedicated to routing BE traffic, while the remaining routes only GS traffics. In order to provide GS, a connection must first be established between source as destination nodes, before the actual data transfer. For this reason, a sequence of VCs through the network is first reserved before data transfer. This guarantees that BE traffics do not interfere with GS traffics. This connection is programmed by the BE router and fed into the GS router as shown in Fig. 3.6.

3.3 Bufferless router

As stated earlier, bufferless routing completely removes buffers from the router in order to eliminate area/power overhead associated with buffers. Bufferless routers consist of pipeline latches that are capable of holding only one flit. Therefore, flits are sent to some output port at the end of each clock cycle. In BLESS [75], packets that cannot advance to their desired output ports are deflected or miss-routed since there are no buffers in the router to store packets. Therefore, packets are continually deflected from one hop to another until they finally reach their desired destination hop.

It has been observed that BLESS [75] consumes far more power and have worse average packet latency compared to input buffered router (IBR) under high network load [101, 38]. This is because more packets get deflected further away from their destination node at high traffic injection rates and these deflected packets unnecessarily consume the router link bandwidth. Whereas, link bandwidth is saved since packets can be stored in buffers when network contention increases in typical IBR routers. Bufferless routing generally leads to simple flow-control since flow-control information can be determined locally without the need for inter router communication. However, large reorder buffers are required at the receiving nodes since flits of a packet are routed independently from each other due to increase in deflections [75, 106].

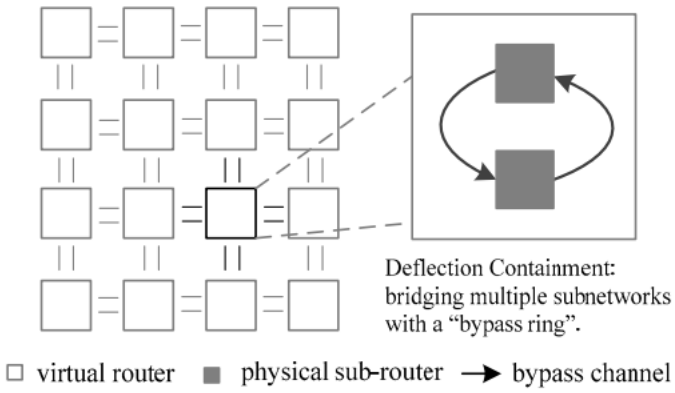


FIGURE 3.7: Overall structure of DeC [106]

Several techniques have been proposed in order to mitigate the latency and power overhead caused by deflected packets at high traffic injection in BLESS. The Deflection Containment (DeC) router [106] is composed of two sub-routers connected by a *bypass channel* as shown in Fig. 3.7. One of the routers receives input port request from neighbouring routers, while deflected packets are forwarded to the second router via the bypass channel. Both routers use independent output links which increase path diversity. The forwarded packet can compete for output port only after a clock cycle. When two packets contend for the same output port, the packet

with the highest priority is granted the output port, while the second packet is forwarded to the second router instead of deflecting it further away from its desired output port. Compared to BLESS, DeC achieves reduced power consumption due to reduced deflections. However, the DeC router leads to poor network utilization under low traffic.

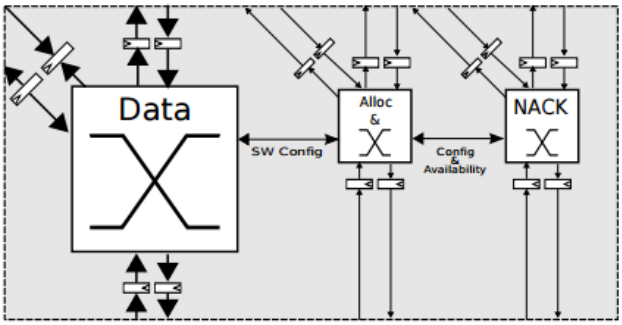


FIGURE 3.8: SCARAB router showing the signaling between allocation, data and NACK networks

Unlike BLESS routing where packets are miss-routed in the case of contention, SCARAB [46] proposed to drop and re-transmit packets that cannot be immediately transmitted. SCARAB [46] uses a negative acknowledgement retransmission network to achieve this goal. SCARAB architecture is shown in Fig. 3.8. However, the retransmission network introduces additional design complexity. In addition, the sending back acknowledgement and retransmitting dropped packets increase overall network loads and limits packet throughput.

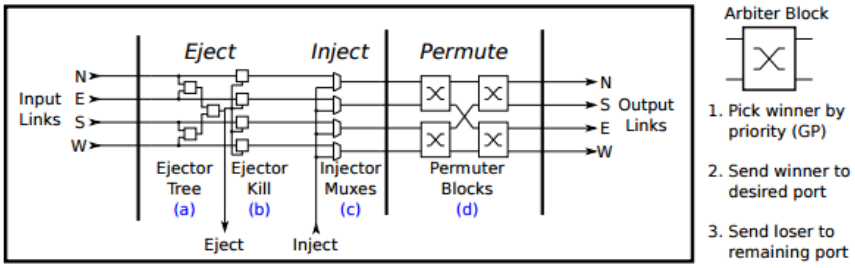


FIGURE 3.9: CHIPPER architecture: a permutation network replaces the traditional arbitration logic and crossbar [39].

A low-cost bufferless router called CHIPPER was proposed in [39]. CHIPPER aims to reduce router complexity introduced by crossbar switch and allocators such as in BLESS [75]. CHIPPER uses a two-stage permutation network requiring smaller crossbars and arbitration unit compared to BLESS router [75]. Chipper architecture is shown in Fig. 3.9. Unlike the BLESS router where packet priority is sequenced, CHIPPER employs random flit priorities. The CHIPPER router design choice leads to much simpler hardware and reduces the number of miss-routed flits compared to

BLESS since packets are simply dropped and retransmitted instead of miss-routed. However, it achieves poor performance compared to BLESS [75].

The performance loss incurred in BLESS router is due to increase in network contentions caused by increased packet deflections. In order to reduce network contention, Authors [80] proposed a method that uses "time average of link utilization" as a metric to determine heavily utilized links in the network. In this approach, flit are deflected to only the least congested network links. Intuitively, avoiding heavily utilized links helps to decrease network contention per time in the network. Compared to BLESS, this approach brings improved network saturation throughput since network contention is reduced.

3.4 Minimal buffered router

In order to reduce growing packet deflections especially under high traffic injection, several existing works proposed hybrid approaches of combining both bufferless and buffered routings. These techniques are discussed in this subsection.

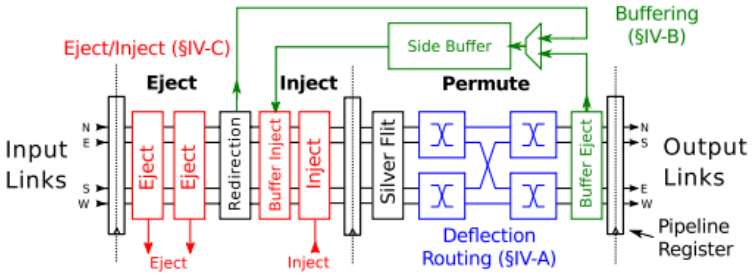


FIGURE 3.10: MinBD router pipeline [38]

A minimally-buffered deflection (MinBD) [38] router proposed to add small buffers (called *side buffers* shown in Fig. 3.10) in the router to temporarily store flits that would otherwise be deflected. However, only a fraction of the flits is stored in the buffer, while others are still deflected. The stored flit can re-enter the network (if there's a free input slot) and participates in output port contention in the next cycle. In order to prevent starvation of flits stored in the side buffers a *buffer redirection* scheme is used. In this scheme, a flit from the router input is randomly chosen and forced into the side buffer in one cycle, while a flit at the end of the buffer is forced to inject into the port in the same cycle when starvation occurs. The temporary flit-storage is beneficial as it reduces the number of miss-routed flits which leads to energy efficiency and performance gains. However, it still provides poorer performance compared to input buffered router since increasing packet defections still occurs at high loads.

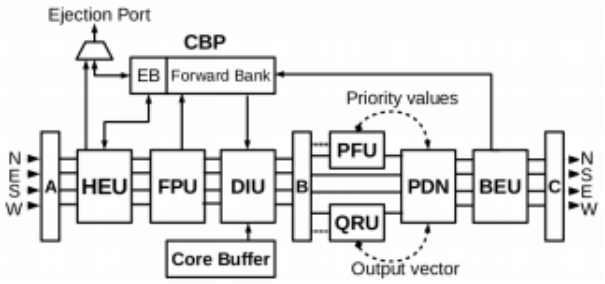


FIGURE 3.11: Router pipeline for DeBAR. HEU-Hybrid Ejection Unit, FPU-Flit Preemption Unit, DIU-Dual Injection Unit, PFU-Priority Fixer Unit, QRU-Quadrant Routing Unit, PDN-Permutation Deflection Network, BEU-Buffer Ejection Unit, CBP-Central Buffer Pool. A, B, and C are pipeline registers. [55]

The DeBAR router [55] uses a set of centralized buffers (known as *central buffer pool*) to store flits that would have otherwise be deflected when flits contend for similar output ports. The DeBAR router pipeline is shown in Fig. 3.11. In order to speed up packets ejection in the router, the router uses a *hybrid ejection-unit* that can process two flits ejections in the same cycle. For this reason, the central buffer pool contains a special flit slot known as *ejection bank (EB)*. If multiple flits from the input port are requesting the router local output port, one of them is granted while the other is stored in the EB. In the next cycle, the flit stored in the EB advances to the local output block without any further delay. In order to decrease packet deflections, flits that have been previously deflected are given priority to use output ports. The DeBAR router [38] leads to improve average network throughput compared to MinBD [38].

A single cycle router architecture capable of controlling the manner in which flits are stored in the *side-buffers* for performance and energy gains, named HiPAD, have been proposed [94]. Unlike MinBD, where flits cannot always progress to their desired output ports since all input ports are not connected to output ports, HiPAD uses a router network that connects all input ports to the output port. This design choice ensures that flits that would otherwise progress to their desired output ports are not unnecessarily buffered in the *side buffer* due to lack of input/output ports structural connection [94]. Also, ensuring that fewer flits are buffered leads to lower packet latencies since buffered flits have to be re-injected into the router input port incurring an extra cycle. Thus, HiPAD leads to reduced average packet latency and lower packets defection rate.

Adaptive Flow-Control (AFC) [54] proposed to combine bufferless and input buffered flow control. Basically, AFC works like a typical bufferless router at low loads but switches to input buffered mode at high traffic injection rates. When the routing is operating in the input buffered mode, packets that would otherwise have been deflected are stored in an input buffered similar to a typical input buffer router. Storing packets leads to performance improvement since router link bandwidths are

not unnecessarily consumed due to packet deflections. However, AFC incurs higher energy associated with storing every flit in the router. AFC leads to complex design since control logic for both forms of routing is present in each router. In addition, power gating is required to switch off the input buffer mode at low load and switch it on at high load [38].

3.5 Shared-buffer routers

As stated earlier, resource sharing is beneficial as it ensures that the buffering resources in the router are exploited for improving the overall network performance. On the other hand, resource sharing can introduce deadlocks caused by cyclic dependencies among shared network resources. This section presents approaches that are used to avoid deadlocks. Typical shared-buffered routers and *roundabout-inspired* routers are also covered in this section.

3.5.1 Deadlock-freeness challenge in resource sharing

Several techniques have been proposed in the literature to tackle the deadlock problem which in most works arises from the use of adaptive routing techniques or folded network topologies such as the Torus. These techniques relate to router flow-control (the manner in which the buffering resources are allocated in the network). Table 3.2 shows the properties of state-of-the-art deadlock-free flow-control techniques. The presented techniques are mostly divided into two categories i.e. virtual-cut-through based and wormhole-based. An ideal flow-control requires one virtual channel per input ports and wormhole flow-control for reducing overheads in terms of area/power introduced by buffers [19].

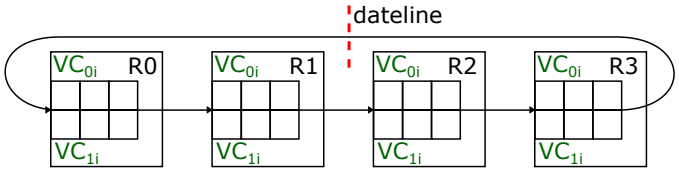


FIGURE 3.12: Dateline deadlock-avoidance technique [66]

The Dateline [24] is a classic flow-control technique used to avoid deadlocks in NoCs. It requires two virtual channels (VCs) (i.e. *low* i.e. VC_{0i} and *high* VC_{1i} in Fig. 3.12) per input port. The "dateline" determines which of the two VCs packets are allowed to use in order to avoid deadlocks. Basically, packets use the low VCs (i.e. VC_{0i}) before crossing the "dateline" whereas, only the high VCs (i.e. VC_{1i}) are used after crossing the dateline as displayed in 3.12. As displayed in Table 3.2, the Dateline approach is available for both VCT and wormhole. The Dateline technique

TABLE 3.2: Deadlock-free flow-control techniques. DL: Dateline [24], LBS: Local bubble scheme [18], CBS: Critical bubble scheme [20], WBFC: Worm-bubble flow-control [19], FBFC-L: Flit-bubble flow-control localized [66], FBFC-C: Flit-bubble flow-control critical [66]

Criteria	Virtual-cut-through			Wormhole			
	DL	LBS	CBS	DL	WBFC	FBFC-L	FBFC-C
No. of VCs	2	1		2	1		
Min VC slot for pck injection	1	2	1		Depends on packet length		
Suitable for long pcks	Yes			Yes			
Suitable variable pcks	Yes	No		Yes	Yes		
Injection pcks latency	High			Low	High		
In-transit pcks latency	High			Low			
Buffer capacity	Large			Small			
Buffer utilization	Low			High			
Design Complexity	High	Higher		High	Higher		
True wormhole	No			Yes	No		

incurs significant area and power overhead associated with VC and also introduces additional implementation complexity for managing VC allocation.

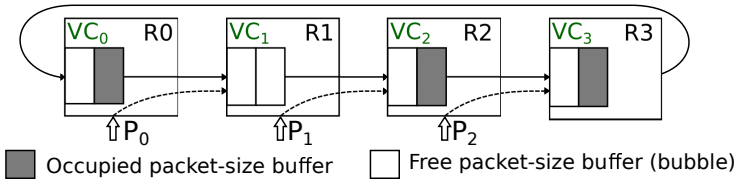


FIGURE 3.13: Local bubble scheme deadlock-avoidance [66]

In order to reduce the overheads associated with using two VCs per input port in Dateline, the bubble flow-control was proposed [18]. Here, a bubble is an empty packet-sized buffer. It is commonly used in tori networks and requires a bubble to be maintained in the *ring* for avoiding deadlock. Bubble flow-control concept is simple but challenging to implement since each node does not have information about the other node sharing a similar ring. Therefore, coordinating resource allocation for all nodes is challenging. In order to avoid the difficulty of gathering global information, the Local-bubble scheme (LBS) was proposed [18]. A Packet can be injected in LBS if the receiving VC has enough space for at least two bubbles so as to ensure that at least one bubble will exist in the VC after injecting a packet. This is illustrated

in Fig. 3.13, where 3 packets (i.e. P_0, P_1, P_2) are waiting to enter the ring. In this example, only P_0 can be injected since the receiving VC has two empty packet-sized buffers. This requirement (of having at least two packets sized buffers per input port) introduces higher VC cost and leads to lower buffer utilization.

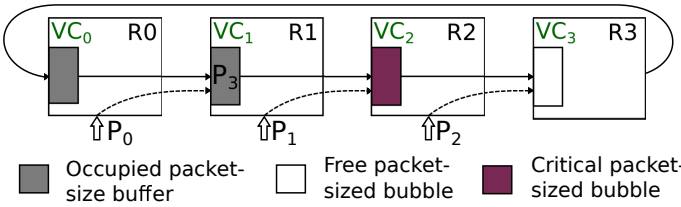


FIGURE 3.14: Critical bubble scheme technique [66]

The Critical bubble scheme (CBS) [20] removes the buffer requirement associated with LBS. It allows each receiving VC to have space for a single packet-sized buffer for packet injection. In CBS, an empty packet sized buffer is marked as *critical* the critical bubble. Packets can be injected as long as the critical bubble is not used. CBS is illustrated in Fig. 3.14, where 3 packets (i.e. P_0, P_1, P_2) are waiting to enter the ring. In this example, P_1 cannot be injected since its injection will occupy the critical bubble. However, P_2 can be injected although the receiving VC (i.e. VC_3) has only one free empty packet-sized VC. In Fig. 3.14, when P_3 advances to VC_2, the critical bubble is displaced backwards i.e. to VC_1.

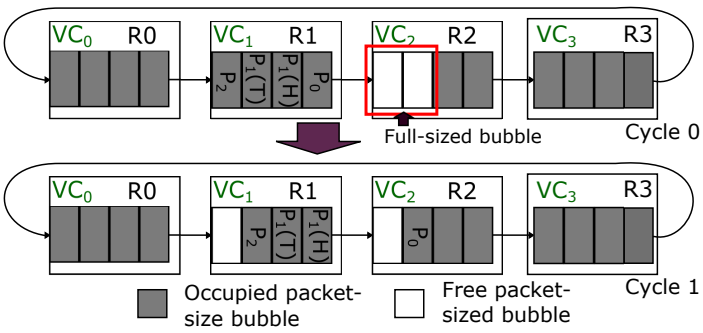


FIGURE 3.15: Deadlock with variable-sized packets in LBS [66]

Compared to Dateline [24], both LBS and CBS require only one VC per input port which leads to buffer reduction. However, both LBS and CBS are not suitable for variable packet size. This is because bubble fragmentation occurs for variable packets, which in turn leads to deadlock [3, 66]. This is illustrated in Fig. 3.15 for LBS, where 3 in-transit packets are flowing in the ring. Note that P_0 and P_2 have length of 1, while P_1 has length of 2. In cycle 0, there are two empty packet-sized slots in VC_2. Next P_0 can advance and occupy one of the empty slot in VC_2 in cycle 1 as illustrated in Fig. 3.15. However, bubble-fragmentation occurs which leads to deadlock in the ring. Packet P_1 cannot advance since VC_2 does not have enough free-slot to accommodate the packet. In terms of implementation complexity, LBS

and CBS eliminates the need to manage VC allocation as they use only one VC per input port. However, additional logic is required for handling packet injection in both schemes.

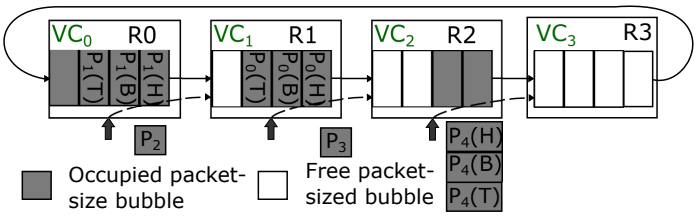


FIGURE 3.16: Flit-bubble flow-control localized (FBFC-L) [66]

Although the LBS and CBS solve the problem of gathering global information and they use only one VC, they still require large buffers which introduce power and area overheads. This is mainly due to the VCT flow-control where buffers are allocated to packets. In order to reduce buffer cost, the bubble flow-control was extended to support wormhole. It has been observed that, for wormhole flow-control, deadlock will not occur in a ring as long as an empty flit-sized buffer slot (i.e. flit bubble) is maintained. In flit bubble flow-control (FBFC) [66], a packet can be injected only if the receiving VC has enough slot to receive the packet. FBFC includes the localized scheme (i.e. FBFC-L) and the critical scheme (i.e. FBFC-C). Both schemes are similar to the LBS and CBS respectively. In order to inject a packet, the FBFC-L scheme requires the number of free buffer slots in the receiving VC to be greater than the packet length. This is to ensure that at least a flit-sized bubble will exist after packet injection. On the other hand, in-transit packets can be forwarded if the receiving VC has one free buffer slot. This is a typical wormhole requirement for injecting or forwarding a packet. FBFC-L is illustrated in Fig. 3.16. In this example, P_3 and P_4 can be injected since the receiving VCs have more free slots than the packet length. On the other hand, P_2 cannot be injected into the ring. In Fig. 3.16, $P_1(H)$ can advance to VC_1 since the buffer requirement for in-transit packet is met.

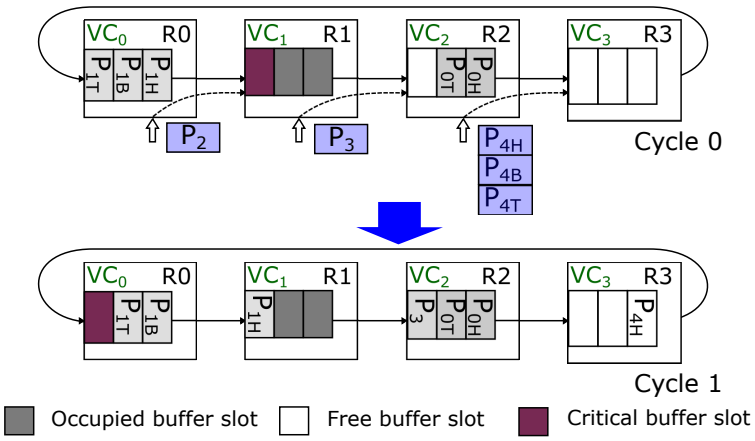


FIGURE 3.17: Flit-bubble flow-control critical (FBFC-C) [66]

In order to reduce the minimum buffer requirement for injecting a packet in the FBFC-L scheme, the FBFC-C was proposed. Similar to CBS, an empty flit-size buffer slot is marked critical. The critical buffer slot can only be occupied by in-transit packets. Contrary to FBFC-L, packets can be injected if the number of free slots in the receiving VC is at least equal to the injecting packet length. However, a packet cannot be injected if its injection will consume the critical flit-sized bubble as shown in Fig. 3.17, where P_2 cannot be injected into the network. This requirement ensures that at least a bubble is maintained in the ring for avoiding deadlocks.

Both FBFC-L and FBFC-C are not suitable for long packets. Implementations of both schemes to support long packets will incur prohibitive buffer cost for wormhole. Both schemes claim to support variable packets length. However, the packet length that can be supported is limited by the number of input VC slots. Conversely, the number of input VC slots can be less than packet length in a true wormhole. Additionally, both FBFC-L and FBFC-L suffers from lower buffer utilization. This is because of the minimum buffer requirement in FBFC-L and the fact that injecting packet cannot use the critical bubble in FBFC-C.

Contrary to FBFC, input buffer capacity can be smaller than packet length in Worm-bubble flow-control (WBFC) [19]. However, packets must reserve additional buffers in the *ring* before they can be injected. This leads to lower buffer utilization and higher packet injection latency associated with reserving enough buffer slots before injecting a packet. Variable-sized packet is also supported in WBFC. However, the packet length is limited since packet length cannot be more than the number of available buffering positions in a ring. In terms of implementation complexity, FBFC-L, FBFC-C and WBFC require additional logic for managing buffer allocation and resolving starvation that results when a node keeps injecting packets.

In order to remove the limitations of existing approaches, this thesis proposed an approach that is based on true wormhole flow-control. This provides adequate support for variable-sized packets, leads to reduce router area overhead, less complex implementation and enhances buffer utilization.

3.5.2 Shared-buffer routers

Router architectures capable of utilizing the available buffering resources in the router for performance gains have been proposed. Such routers allow input port buffers to be shared by multiple input ports. Other approaches dedicate a set buffers to the input port but allow input ports to share output port buffers.

A distributed and shared-buffered router was proposed in [95]. This router consists of dedicated input buffers and emulates output buffers shared by the input ports. Compared to input buffered router (IBR), the proposed router [95] provides a higher throughput. However, this performance gain comes at the expense of higher

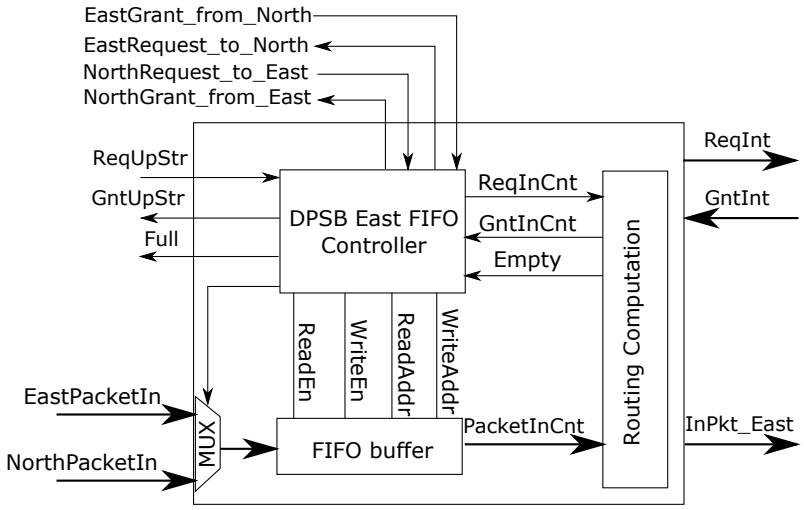


FIGURE 3.18: DPSB Router East Input Port [44].

power dissipation and area overhead introduced by the additional crossbar and complex arbitration scheme used in the router.

The RoShaQ15 [101] router is composed of additional queues known as *shared-queues* besides input buffers. Combining both queues (i.e. input/shared queues) leads to much simpler arbitration. The shared queues can be used by packets if empty or contain packets destined for the same output ports. RoShaQ15 uses a by-pass technique to allow packets from the input port bypass the queue so as to achieve reduced zero-load latency. However, RoShaQ15 requires an additional crossbar for allocating the shared queue which incurs a higher power and introduces additional area overhead compared to typical input buffer router. RoShaQ15 architecture is shown in Fig. 2.16.

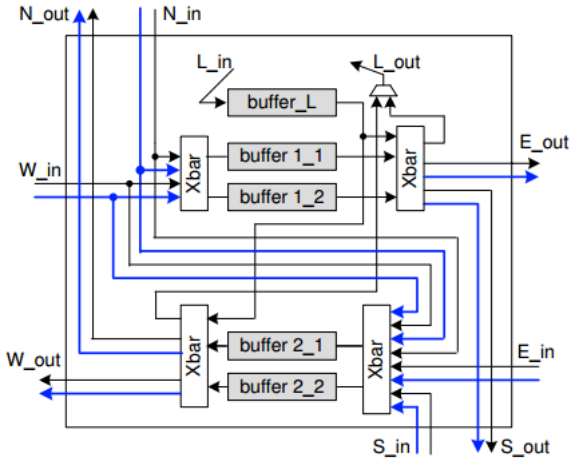


FIGURE 3.19: Dual-lane router architecture with two shared buffers and two interconnect links on each lane [102].

The DPSB [44] is a shared router architecture that allows sharing buffering resources between multiple input ports. Its architecture is shown in Fig. 3.18. Contrary

to existing routers with shared queues [101], DPSB shares buffers between two input ports, thereby eliminating the area and power overhead associated with shared queues. The DPSB architecture is composed of FIFO buffers that are controlled using the DPSB controller. Basically, buffering request for a given input port is granted if the buffer is not full. However, if the input port FIFO is full, the controller checks the packet destination using the *turn model* to determine if other input port FIFO can be used to store the packet. The dual buffer architecture leads to enhanced network performance offering higher throughput and lower latency compared to typical input buffered router. However, it leads to increase in power and area overheads compared to a conventional router.

The DLABS router [102] is composed of dual-lanes (i.e. router buffers on separate lanes) as shown in Fig. 3.19. In order to avoid deadlocks, input and output port links for the router ports connects to separate buffers. This breaks all loop in the router by ensuring that cyclic dependencies among shared resources do not exist. Data transfer between the different lanes is restricted to a single direction (i.e. unidirectional) in order to avoid deadlocks. In order to avoid performance loss caused by head-online (HOL) blocking, so common in wormhole router, DLABS uses multiple buffers for each router lane. Multiple links are also used at each router output port for performance improvements. However, DLABS incurs higher area overhead compared to typical input buffered routers due to additional buffers and control logic used in the router.

Authors [61] proposed a router architecture capable of sharing the buffers between two neighbouring input ports. In the proposed router, a set of buffers is shared between two input ports. Specifically, a set of buffer is shared between the north and east input ports, while the other set of buffers is shared between the south and west input ports. This approach utilizes the idle buffers in the router as opposed to introducing additional queues (i.e. shared queues). This approach reduces the need for complex arbitration and additional crossbar in the router.

Flexible router [91] exploits existing buffers in the router in order to realize a high network throughput. In the Flexible router, packets are allowed to use other available input ports if their desired input ports are not available as shown in Fig. 3.20. However, deadlock will occur if packets are allowed to use just any input ports when contention occurs. In order to avoid deadlocks, packets are allowed to use only certain input port buffers. This choice is based on the routing function *turn-model*. The flexible router provides higher network saturation throughput compared to typical input buffered router. However, it incurs additional area overhead.

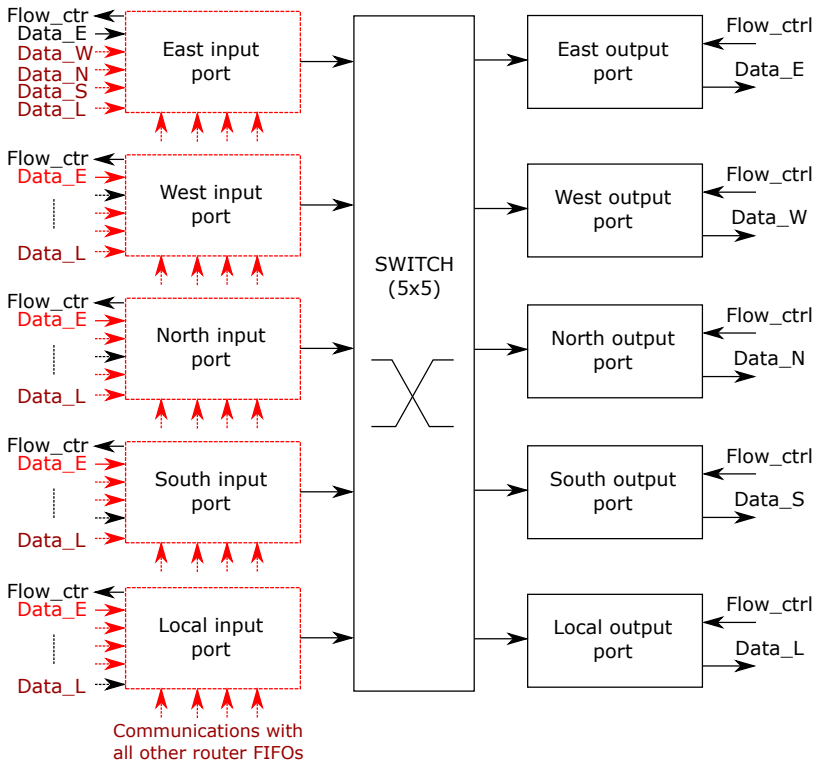


FIGURE 3.20: Flexible router architecture [91].

3.5.3 Roundabout-inspired routers

The Rotary [2] router shares similar *traffic-roundabout* and *ring-like* concept with the proposed the *R-NoC* router. It consists of two independent rings implemented using Dual-port FIFO Buffers (DBF) as shown in Fig. 3.21. The input stage determines which of the two rings to forward a packet when it enters the router. The decision is made based on the packet distance to a suitable output port at low load and the ring-occupancy at medium to high loads. The packet continues on the ring to its desired output port. On reaching the output port, the packet exits the router if the output port is free. Otherwise, it keeps circulating the ring until another suitable output port is later found.

A packet in Rotary adaptively uses the first available output port after making a specified number of turns in the router. This strategy helps to avoid head-on-line blocking since packets can move to the next DBF from the input port, thereby allowing packets that are behind to make forward progress. The Rotary router is topology agnostic and uses a low complexity adaptive routing. The Rotary relies on combined virtual-cut-through (VCT) and bubble flow-control (known as LBS) to avoid deadlocks in the ring. VCT allocates buffers to entire packets, thereby requiring large buffers in the router. Bubble flow-control is used to control packet injection ensuring that packets can only be injected into the ring if there exist two free DBFs.

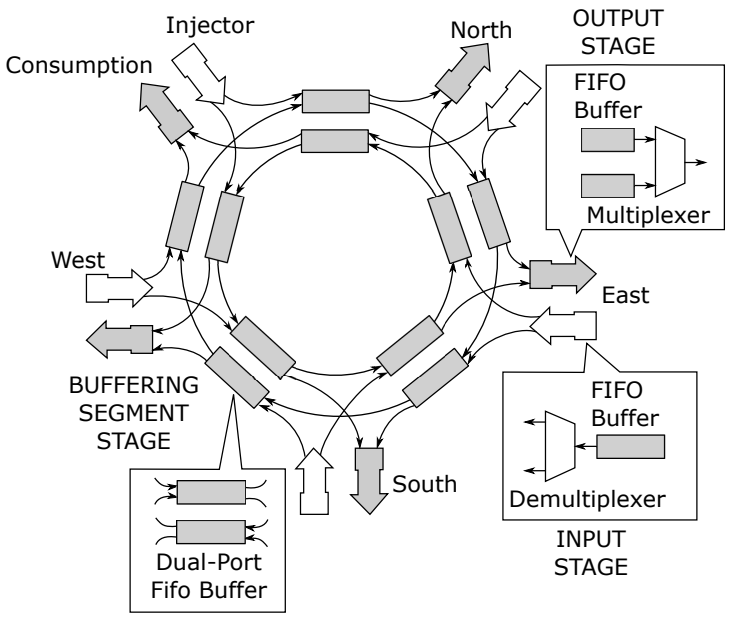


FIGURE 3.21: Rotary router architecture [2]

3.6 Summary

This chapter discusses existing Network-on-Chip router designs including input buffered, bufferless and shared buffered routers. Analysis of existing works show that although virtual-cut-through (VCT) based routers provide improved performance over wormhole routers, they typically incur area overhead and power consumption caused by large buffers. On the other hand, wormhole based routers require smaller buffers, which keep area low compared to VCT routers. However, they suffer from poor performance caused by blocked packets. In order to improve the performance of wormhole router, the physical channels can be multiplexed into multiple virtual channels (VCs). VC based routers generally provide improve network saturation compare to wormhole routers since head-on-line (HOL) blocking is removed and channels bandwidth are better utilized. However, the performance improvement has associated area cost due to larger crossbars. VC introduces additional router complexity due to VC allocation and suffers from poor resource utilization under low traffic.

In order to reduce on-chip area and power consumption, bufferless routing completely remove buffers from the router, thereby saving area and power. However, bufferless routers suffers from poor performance under high traffic due to increased packet deflections.

In order to maximize buffer resource utilization and to provide low-latency support for applications with diverse traffic characteristics, the concept of *roundabout* is considered. *R-NoC* architecture provides inherent resource sharing allowing for the buffering resource and link bandwidth to be shared by several input/output ports

for performance gain. *Roundabout NoC* ($R - NoC$) provides a highly-adaptive architecture that allows the router to be configured to provide varying performance/area trade-offs.

Chapter 4

The Roundabout concept for effective buffer resource sharing

«"Success is where preparation and opportunity meet."»

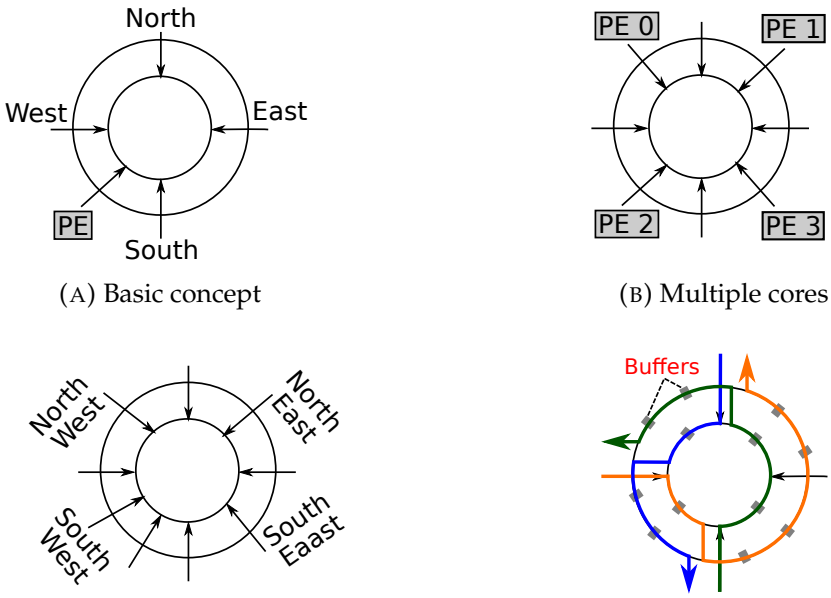
Bobby Unser

Contents

4.1 Introduction

The *Roundabout NoC* $(R - NoC)$ concept draws inspiration from real-life multilane traffic roundabouts, which allows for distributed control/arbitration contrary to traditional NoCs. The *R-NoC* concept provides a highly-adaptable architecture, which allows the router to be configured to meet numerous network topologies and application demands. This is depicted in Fig. 4.1 using highly conceptual diagrams. Fig 4.1a shows the router concept, where the available resources are shared by multiple ports. Sharing the available resources among several ports provides support for applications with varying traffic characteristics. As depicted in Fig. 4.1b, the architecture can be readily adapted to support multiple processing cores without incurring significant router modifications and crossbar cost. Connecting multiple cores to a router reduces the number of routers needed in the network, participates to decreasing latency since the number of hops in the network is reduced. A practical application is in the "X-Network" [105], where each router is connected to four neighbouring cores. This configuration is beneficial in terms of area reduction and performance.

In terms of network-on-chip topologies, the router concept can be readily extended to provide support for varying network topologies by the addition of multiple ports (depending on the targeted network topology) as depicted in Fig. 4.1c. This provides varying performance/area trade-offs. One of the objectives of this work is to explore various network topologies for performance improvement. For this reason, *R-NoC* is extended to provide support for various network topologies. *R-NoC* is scalable in terms of the number of lanes, internal topology as well as in the use of additional buffering resources that can be arbitrarily distributed over lanes



(A) Basic concept

(B) Multiple cores

(C) Topologies

(D) Adaptivity

FIGURE 4.1: The *R-NoC* architecture configurations

as depicted in Fig. 4.1d. The scalability of the router will be explored in greater depths in Chapters 6 and 7. *R-NoC* provides adaptive use of lane resources, where outer lanes get used as traffic grows. This is depicted in Fig. 4.1d, where outer lanes resources are utilized by packets.

This chapter introduces the *R-NoC* concept in detail using the base *R-NoC* router topology (i.e. $R - NoC$ with 4 lanes and 5 input/output ports). The deadlock-proneness of the base topology is identified and an algorithm for generating deadlock-free *R-NoC* router for varying network topologies is presented. The proposed algorithm is then applied to two case studies i.e. the mesh and diagonally-linked mesh *R-NoC* routers.

4.2 General principle

As stated earlier, the *R-NoC* router concept is inspired by real-life multi-lanes traffic roundabouts where cars go on a low-priority lane and switch to a higher priority lane should they miss their exit. In such traffic roundabouts, cars on the higher priority lanes have priority over cars on lower priority lanes to use the exit. Similarly, the *R-NoC* router consist of multiple lanes shared by multiple input/output ports in order to maximize buffer resource utilization for improving overall network performance.

Fig. 4.2 shows the initial conceptual *R-NoC* architecture. It consists of *N-number* of lanes arranged in increasing order of priorities. The innermost lane (i.e. lane 0 in the Fig. 4.2) has the lowest priority, while the outermost lane has the highest

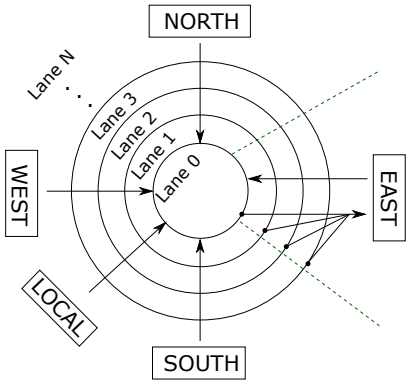
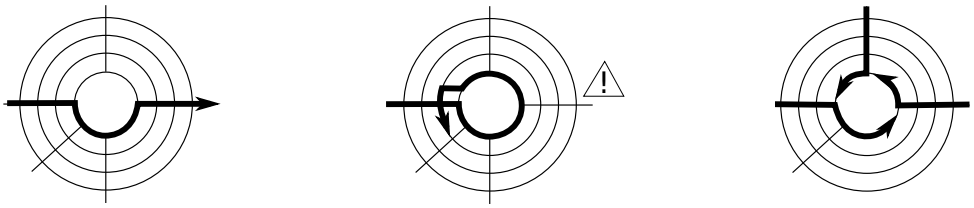


FIGURE 4.2: The initial deadlock-prone *R-NoC* topology

priority. The lanes are grouped into two categories i.e. *primary* lanes and *secondary* lanes depending on the input ports distributions. Similar to the lane priorities of real-life traffic roundabout, the primary lane resources are exploited at low traffic whereas the secondary lane resources are exploited at high traffic. The input ports are distributed only to the *primary* lanes, while the *secondary* lanes are exploited only whenever congestion occurs. In Fig. 4.2, lane 0 is the primary lane since the input ports are attached to it, while the other lanes are the secondary lanes (i.e. lane 1 to lane N in Fig. 4.2. The primary lane has the lowest priority in the router topology.



(A) Input to output port (B) Lane switching (C) Deadlock config

FIGURE 4.3: Data-flow scenarios in deadlock-prone *R-NoC* topology

Similar to the outermost lanes in real-life traffic roundabouts, packets on the secondary lanes are given priority to access shared resources whenever requests for a shared resource originate simultaneously from both lanes. When a packet enters the router, it first goes on the lowest priority lane (primary lane) since the input ports are only distributed to the primary lane. The packet continues on the lane and can make its way out of the router if the desired output port is free. This is illustrated in Fig. 4.3a, where a packet from the west input port is flowing out of the router via the east output port. On the other hand, if the desired output port is not free, the packet continues on the lane and then switches to a higher priority lane at the packet input as shown in Fig. 4.3b. Lane switching is beneficial as it frees-up the router resource for use by incoming packets.

Allowing the lane resources to be shared by all the input ports is beneficial as it ensures that the buffering resources assembled on the lanes are effectively utilized

which in turn leads to improved network throughput. However, this form of sharing can lead to a deadlock situation on any of the router lanes. This deadlock is due to cyclic dependencies that result when packets are simultaneously occupying lane resources and requesting for other lane resources occupied by other packets. In the scenario shown in Fig. 4.3c, none of the packets can make forward progress as their requested resource are currently held up by another packet.

4.3 Avoiding deadlock in *R-NoC*

Before describing how deadlock is avoided in *R-NoC*, this section first provides a general overview of deadlocks in NoCs and describes a well-known state-of-art deadlock avoidance technique that has been extensively applied for realizing deadlock-free adaptive routing in NoCs and deadlock-freeness in ring/tori network topologies.

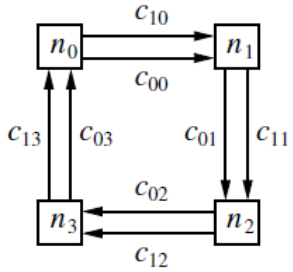


FIGURE 4.4: Unidirectional ring with two VCs [32]

Fig. 4.4 shows a unidirectional ring network with four nodes denoted as n_i, where $i = \{0, 1, 2, 3\}$. Each node connects to adjacent nodes using a pair of outgoing channels denoted as C_{0i} and C_{1i}. As discussed in Chapter 2, a network has routing function that determines the paths taken in the network. A routing algorithm must be *connected* meaning that each node in the network must be reachable [32]. As an example, consider a deterministic routing algorithm that states as follows: If the current node n_i is equal to the destination node n_j, store the packet. Otherwise, use C_{0i} or C_{1i}, $\forall\ j \neq i$. This means that for example, data from node n_3 destined for node n_0 can always use either channel C_{03} or C_{13}. Similar arguments can be made for the other nodes in the network. In order to assess the deadlock freeness of this network, a theoretical model for deadlock-avoidance proposed by Duato [32] and also by Dally [24] is considered. This method relies on building the channel dependency diagram of the network and checking for cycles. According to Duato's theorem [32], a routing algorithm (deterministic) for an interconnection network is deadlock-free if and only if there are no cycles in the corresponding channel dependency graph (CDG). The network topology and the routing algorithm (also known as routing function) are key inputs to building the channel dependency graph. Two resources

(channels/buffers) are dependent when a packet can occupy a resource and request access for the other resource.

(A) CDG with cycles (B) CDG without cycles

FIGURE 4.5: Channel dependency graphs (CDG) for Fig. 4.4

Fig. 4.5a shows the CDG diagram for Fig. 4.4. Note that the CDG is simplified since only the low channels (i.e. C_{0i}) are taken into account. A node on the CDG represents channels, while the edge represents the dependency. It is obvious from the CDG that the network is not deadlock-free as a cycle exist in its CDG according to Duato's theorem [32]. On the other hand, consider a new routing function for Fig. 4.4 that states as follows: If the current node n_i is equal to the destination node n_j, store the packet. Otherwise, use c_{0i} if $j < i$, or c_{1i} if $j > i$ [32]. Fig. 4.5a shows the new CDG for the network. It is observed that the network is deadlock-free since its CDG has no cycles [32]. Note that channels c_{00} and C_{13} are never used.

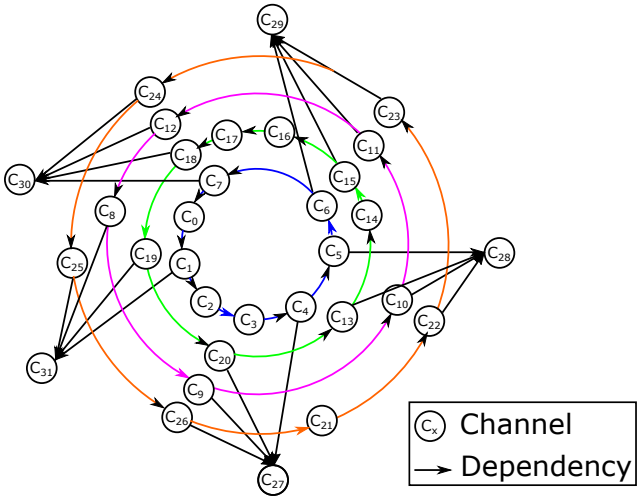


FIGURE 4.6: CDG of deadlock-prone *R-NoC* topology

Fig. 4.6 shows the CDG for the deadlock-prone topology shown in Fig. 4.2. It is visible from the graph that cycles (in colors) exist in the CDG. Therefore, the configuration is not deadlock free. Note that the output ports channels have been included in the CDG.

4.3.1 Topology generation algorithm

The initial *R-NoC* topology shown in Fig. 4.2 is deadlock-prone as cyclic dependencies can occur on the lanes as described in Section 4.2. The algorithm used in transforming the initial deadlock-prone *R-NoC* topology to a deadlock-free topology is presented here. The proposed algorithm is then applied to realize deadlock-free *R-NoC* topologies for the mesh and diagonally-linked mesh networks. The properties of the deadlock-free router for mesh network topology is also presented in this section.

In order to realize a deadlock-free *R-NoC* configuration, a combination of input ports that can share a set of lane resources without introducing cyclic dependencies in the router and the corresponding network must be obtained. The lanes must also be distributed such that the network is not disconnected (i.e. valid paths must exist from input ports to output ports in the router). For this reason, a deadlock-free *R-NoC* topology generation algorithm shown in Algorithm 1 is proposed. The goal of Algorithm 1 is to generate deadlock-free *R-NoC* configurations for varying network topologies. The algorithm execution proceeds in several steps.

In ***Step 1***, input data are declared. This algorithm takes a set of lanes (L), a set of input ports (I) and a set of output ports for a given input port (O_i) as inputs.

In ***Step 2***, sets required to store information about lanes and their shareable input ports are created.

Step 3 represents the heart of the algorithm. A given lane (i.e. l) is selected from the set of lanes in *line 2*. *Line 3* selects an input port from the set of input ports and associates it with the selected lane. In *line 4*, the path from the selected input port to all valid output ports is marked. At this stage, no cycle exists in the lane since only one input port is associated with the selected lane. Next phase is to find other input ports that can share the selected lane (i.e. l) with the currently selected input port (i.e. i). For this purpose, the algorithm loops through all the set of input ports in *line 5 to line 12*. In each iteration, an input port is selected and associated to the lane. The path from the selected input port (i) to all valid output ports (O_i) is marked in *line 6*. Then the algorithm check if any cycle exists in the lane L. If a cycle exists then the currently selected input port (i) cannot share the selected lane l resources with the other input port(s). On the other hand, if no cycle exists, then the currently selected input port can be shared with the other input port(s)). The iteration is repeated (for all unused input ports) until all the input ports have been successfully attached to a lane without any deadlock.

Finally, in ***Step 4***, switch links can be added only between lanes that are already shared. Also, care must be taken to ensure that introducing switch links do introduce cycles and unreachable output port. For this reason, a switch link can be added if and only if its addition will not lead to a deadlock or unreachable output port. As

Algorithm 1: *R-NoC* topology generation algorithm

Step 1: Input information. Let us consider the following: L: a set of lanes, I: a set of input ports, O_i: a set of output ports for a given input port i, $SLink$: a set of switch links;

Assumption: Each lane $l \in L$ (resp. input port $i \in I$) is associated with a Boolean attribute called "used". When a lane l (resp. an input port i) has been used, then $l.used$ (resp. $i.used$) is set to *true*, otherwise it is set to *false*;

Step 2: Local variables. Let us consider the following: $I_l^s \subseteq I$: a set of shareable input ports, $P = \{\langle l, I_l^s \rangle\}$: a set of pairs composed of a lane l associated with its shareable input ports $I_l^s \subseteq I$,

Step 3: Definition of lanes shared by multiple input ports;

1 **do**
2 select a lane $l \in L$ and set $l.used$ to *true* ;
3 select $i \in I$, associate i to l (i.e., $i \in I_l^s$), then set $i.used$ to *true*; // i belongs to the set of shareable input ports associated with l within a pair in P;
4 Label the path along lane l from input port i to all output ports in O_i that are reachable from i // The path from the selected input port i is to eligible output ports on the lane l is marked. ;
//Next step is to repeat the above process on other input ports that can share the selected lane l;
5 **foreach** $i \in I$ *s.t.* $i.used = false$ **do**
6 Label the path from i to its corresponding valid output ports O_i attached to the lane l;
//check if cycle exists in created paths;
7 **if** *no cycle exists in combined paths* **then**
//can share lane;
8 set $i.used$ to *true* and $i \in I_l^s$;
9 **else**
// input port i cannot share lane l with other input ports;
10 $i \notin I_l^s$;
11 Create a pair $\langle l, I_l^s \rangle$ and save it in P;
12 **while** *(there exists an input port* $l \in L$, *s.t.* $l.used = false$*)*;

Step 4: Switch links insertion without deadlock or unreachable output port;

13 **do**
14 Pick a switch link and insert it between two shared lanes if no cycle along resulting combined paths and no unreachable output port;
15 **while** *(there is any unused switch links in SLink)*;

Output: Deadlock free *R-NoC* topology

discussed earlier, CDG can be formally used to check for the presence of deadlocks. Algorithm 1 assumes a DOR or partially adaptive routing based on turn model. Also, generated *R-NoC* router configurations can only be plugged into a deadlock-free NoC network i.e. network using deterministic or minimal-adaptive routing algorithms. This guarantees deadlock-freeness at network level. The current version of *R-NoC* does not support fully adaptive routings as they deadlock-prone.

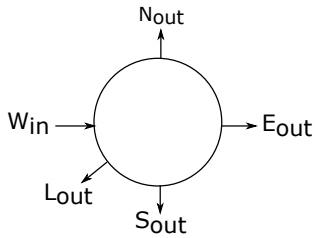
4.3.2 Application to mesh-based topology

As stated earlier, the goal of the deadlock-free *R-NoC* topology generation algorithm is to generate deadlock-free *R-NoC* router configurations for various network topologies. This subsection applies the algorithm in order to realize a deadlock-free *R-NoC* router configuration (with 4 lanes) for the mesh network topology using XY routing algorithm.

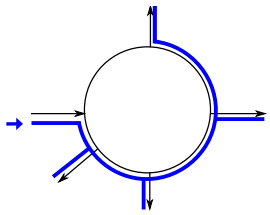
TABLE 4.1: *XY routing algorithm* possible source and destinations

Source (input ports)	Destinations (outputs ports)
Local	West, South, East, North
West	Local, South, East, North
East	West, South, Local, North
North	South, Local
South	North, Local

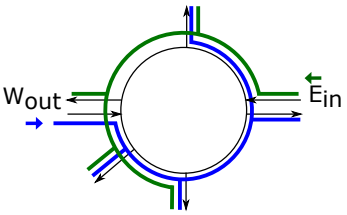
Table 4.1 gives the list of input ports and their corresponding output ports for *XY routing algorithm*. Thus, in ***step 1*** of Algorithm 1, the input information are a set of 4 lanes, a set of input ports and a set of output port for each input port according to Table 4.1. Next, ***step 3*** of Algorithm 1 can be applied to devise the list of input ports than can share the lanes such that cyclic dependencies are avoided on each of the shared lane.



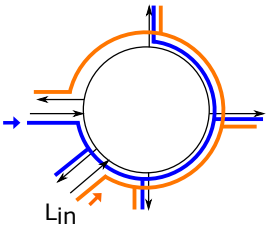
(A) West input and possible output ports



(B) Output ports label for west input port



(C) Output ports labels for west and east input ports



(D) Output ports labels for west and local input ports

FIGURE 4.7: *R-NoC* application example for Mesh network topology

Assuming that the *west input port* is selected from the set of inputs port in *line 2* of Algorithm 1 and associated to a selected lane (l) in *line 3*, then the resulting configuration is shown in Fig. 4.7a, where the west input port and its corresponding output ports are associated to the lane. Upon executing *line 4* of the algorithm, the

resulting configuration is shown in Fig. 4.7b with "blue label" indicating the possible output ports a packet flowing from the west input port can use, as displayed in Table 4.1. It is visible from Fig. 4.7b that no cyclic dependency exists as the path between *north output port* to *west input port* is not labeled.

The next step is to determine the other input ports that can share the selected lane with the *west input port*. For this reason, assuming that *line 5* (first iteration) selects the *east input port*, executing *line 6* gives the configuration shown in Fig. 4.7c. In Fig. 4.7c, the lane has both blue and green labels. The green label represents valid output ports for packets flowing from the east input port. Next step is to check if a cycle exists in the lane. It is visible from the figure that the both labels form a cycle on the lane (i.e. all the paths on the lanes are labelled). Therefore, the *west input port* cannot share the selected lane resources with the *east input port*. A more formal means of checking for cyclic dependencies is to plot the channel dependency graph of the lane as explained earlier.

Algorithm 1 control returns to *line 5* in order to find possible input ports that can share the lane resource with the *west input port*. If the local input port is selected in *line 5* and the lane is labeled "orange" (i.e. *line 6*), then the resulting configuration is shown in 4.7d. It is obvious from Fig. 4.7d that no cycle exists in the lane as the path between the *west output port* and the *east input port* is not label. Thus, sharing the lane between the west input port and the local input port does not result in a deadlock configuration. Lines *1 to 11* are repeated until all the input ports have been attached to a lane. If only an input port is left after many iterations, it can be attached to a single lane.

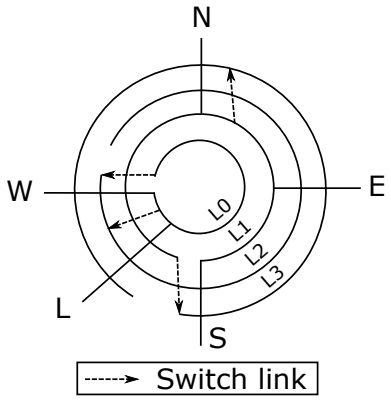


FIGURE 4.8: Generated *R-NoC* topology

Fig. 4.8 shows the final output of the algorithm. Note that according to ***step 4*** of Algorithm 1, only input ports that share a given lane can switch to a new lane so as to avoid deadlocks in the *secondary lanes*. Also, switch links should not introduce unreachable output ports. Thus, in Fig. 4.8, lane 2 (i.e. L2 in Fig. 4.8) is shared only between the west and local input ports since they both share lane 0. A similar

argument can be made for the south, east and north input ports sharing lane 1 and 3. In the topology shown in Fig. 4.8, *lane 0* (i.e. L0 in Fig. 4.8) hosts the local and west input ports, while *lane 1* (i.e. L1 in Fig. 4.8 hosts the east, south and north input ports.

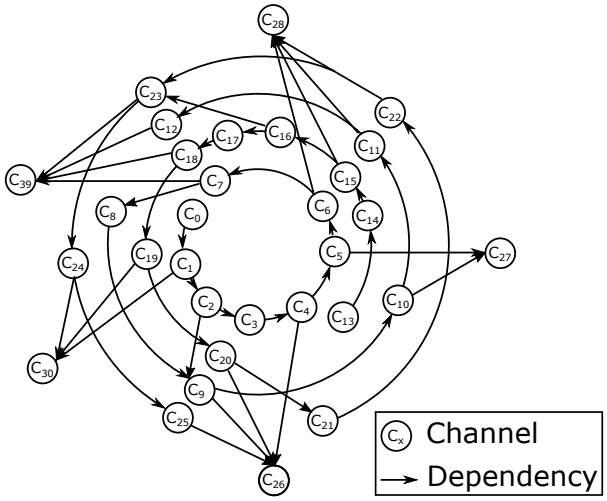
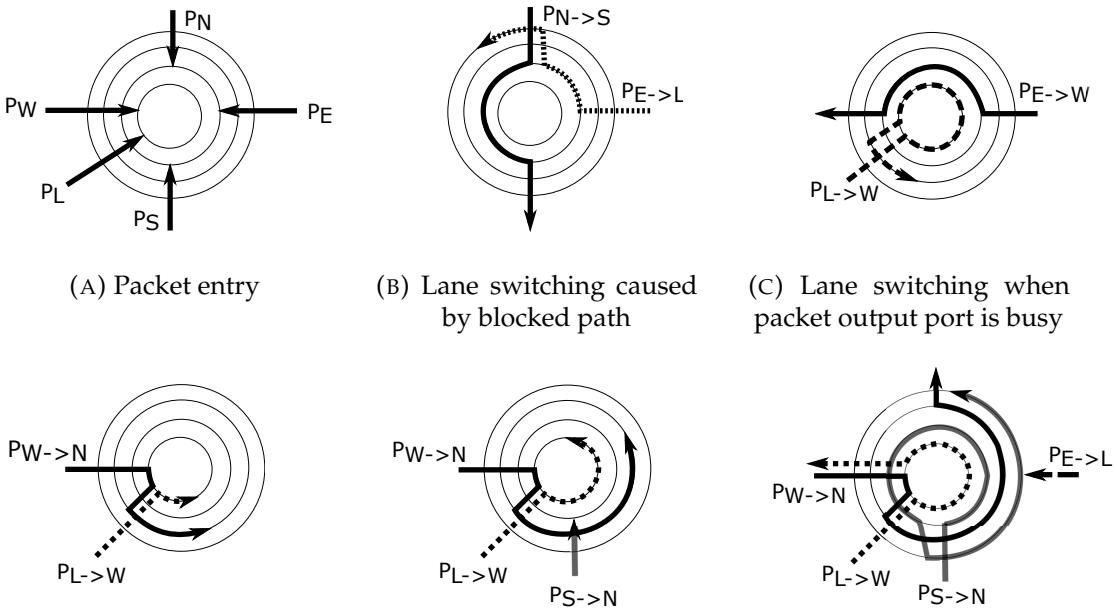


FIGURE 4.9: CDG of deadlock-free *R-NoC* topology

Fig. 4.9 shows the CDG (output channels included) for the deadlock-free *R-NoC* topology in Fig. 4.8. The topology is deadlock-free as there are no cycles in its CDG as shown in Fig. 4.9. Notice that there are no dependencies between lane 1 and lane 3 to the east output port. This is because lane 1 and 3 are shared among the east, north and south input ports and packets from any of these input ports do not use the east output port according to the information displayed in Table 4.1. Thus, Algorithm 1 can generate a cycle-free *R-NoC* topology.

Properties of the deadlock-free *R-NoC* topology

Fig. 4.10 shows packet flow scenarios in the deadlock-free *R-NoC* configuration shown in Fig. 4.8. Here, when a packet flowing on a *primary* lane is blocked or not granted access to the output port, it switches to a *secondary* lane via the *switch link*. A scenario where a packet switches to a *secondary* lane, via a switch link, because its path is blocked is shown in Fig. 4.10b. In Fig. 4.10c, a packet from the east input port destined for the local output port switches to a *secondary* lane because the path is being used by another packet. This design choice avoids queuing packets on the lane (if possible) whenever multiple flows compete for shortest path resources, which increases router resource utilization and packet throughput. Fig. 4.10c shows a scenario where a packet from the local input port and destined for the west output port switches to a *secondary* lane because the west output port is occupied by a packet flowing from east input port to west output port. Packets can only make forward progress (i.e. continue on a given lane or switch to a secondary lane) if the



(A) Packet entry

(B) Lane switching caused by blocked path

(C) Lane switching when packet output port is busy

(D) Low traffic

(E) Medium traffic

(F) High traffic

FIGURE 4.10: Packet flow scenarios in deadlock-free *R-NoC* configuration. $P_{X \to Y}$: a packet from X input port and destined for Y output port.

desired lane resource is available. Otherwise, they are queued on the lane. The output port is granted in a *round-robin* manner if two simultaneous request for a given output port originates from either two primary or two secondary lanes.

More generally, the main properties of *R-NoC* can be summarized as follows:

1. The router can have N lanes (where $N \geq 2$). The lanes are partitioned into *primary* and *secondary* lanes with an arbitrary lane count in each.

2. The output ports are connected to both primary and secondary lanes, while the input ports are only connected to the primary lanes.

3. Packets can switch from *primary* to *secondary* lanes when either their path is blocked or their output is unavailable.

As shown in Fig. 4.10d to 4.10f, almost all the lane resources can be used when network load is high.

The deadlock-free topology can be plugged into a mesh network topology using XY routing without deadlocks. As an example, Fig. 4.11 shows data-flows on a 2x2 mesh network topology. The *transpose* traffic pattern is considered for this example. Here, it is assumed that the packets, i.e. $P0$, $P1$, $P2$, *and* $P3$ were injected into the network at the same time. As shown in Fig. 4.11, no deadlock occurs on this network and all packets will eventually get to their destinations. Notice that $P1$ destined for node *R2* waits just before the east input port because $P3$ is currently occupying the lane. However, after sometime $P1$ can advance and make its way to the local port

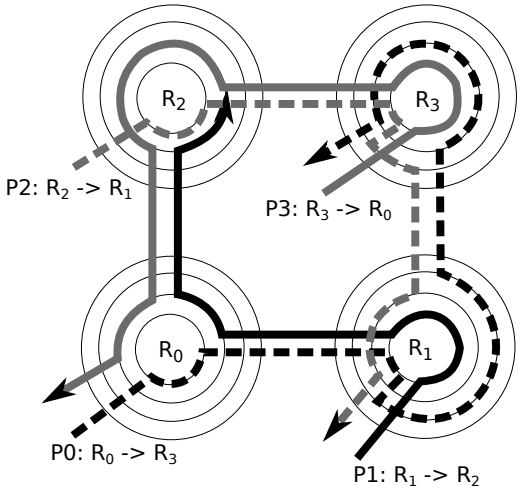


FIGURE 4.11: Data-flow example on a 2x2 mesh network topology. $PX_{Y \to Z}$: X is packet number, Y is packet source node and Z is packet destination node

of node *R2* when *P3* is completely transmitted. The packets can successfully make forward progress and deadlocks do not occur in this network.

4.3.3 Application to diagonally-linked mesh-based topology

The goal of this subsection is to realize a deadlock-free *R-NoC* router configuration for the Diagonally-linked mesh (DMesh) [50] network topology using Algorithm 1. Before applying the algorithm, a general overview of the DMesh network topology in terms of links and routing function is first provided.

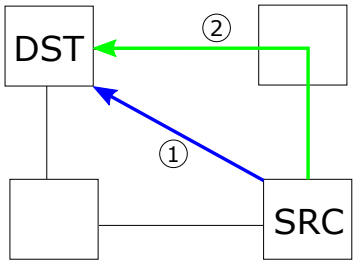


FIGURE 4.12: Adaptive routing in Diagonally-linked mesh topology. SRC: source. DST: destination

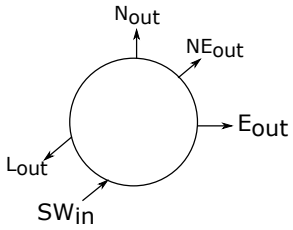
Unlike classical mesh network topology with 5 input/output ports, the DMesh has additional 4 diagonal ports as stated earlier. The diagonal links in the DMesh provide shorter network paths and help to reduce network congestion which leads to enhanced network performance [50]. The DMesh uses quasi-minimal adaptive routing where packets first attempt to use the diagonal ports if available. Fig. 4.12 shows packet flow from a source to destination node. In Fig. 4.12, *path 1* is the minimal path, while *path 2* is the non-minimal path. If packets always use the minimal path then the diagonal links will be congested while the non-diagonal links will

be left unutilized. Thus, leading to uneven distribution of network loads. For this reason, a non-minimal path can be used when the minimal path is busy [50].

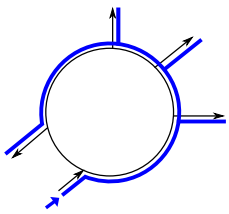
TABLE 4.2: *Quasi-minimal adaptive routing algorithm* possible source and destination ports

Source (input ports)	Destinations (outputs ports)
West	East, Local
South	West, East, North, Local, North-west, North-east
East	West, Local
North	West, South, East, Local, South-east, South-west
Local	West, South, East, North, North-east North-west, South-east, South-west
North-east	West, South, Local, South-west
North-west	East, South, Local, South-east
South-east	West, North, Local, North-west
South-west	East, North, Local, North-east

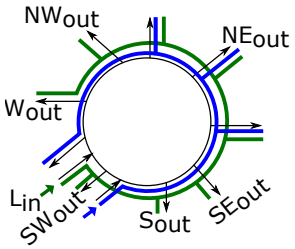
Table 4.2 shows the valid input ports and their corresponding valid output ports for *quasi-minimal adaptive routing algorithm*. Therefore, in ***step 1*** of Algorithm 1, the input information are a set of lane, a set of input ports from Table 4.2 and a set of output port for each input port as displayed in Table 4.2. Given the input information, the algorithm 1 can be applied to devise the list of input ports that can share lanes such that cyclic dependencies are avoided on each of the shared lane.



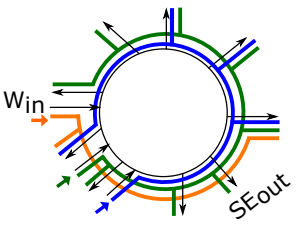
(A) South-west input and possible output ports



(B) Output ports label for south-west input port



(C) Output ports labels for south-west and local input ports



(D) Output ports labels for south-west, local and west input ports

FIGURE 4.13: *R-NoC* application example for Mesh network topology

Assuming that the *south-west input port* is selected from the list of inputs in *line 2-3* of Algorithm 1, then the resulting configuration is shown in Fig. 4.13a, where the

south-west input port and its corresponding output ports are attached to the lane. When *line 4* of the algorithm is executed, the resulting configuration is shown in Fig. 4.13b with "blue label" indicating the possible valid output ports for a packet flowing from the south-west input port as shown in Table 4.2. It's visible from Fig. 4.13b that no cyclic dependency exist as the path between *local output port* to *south-west input port* is not labeled.

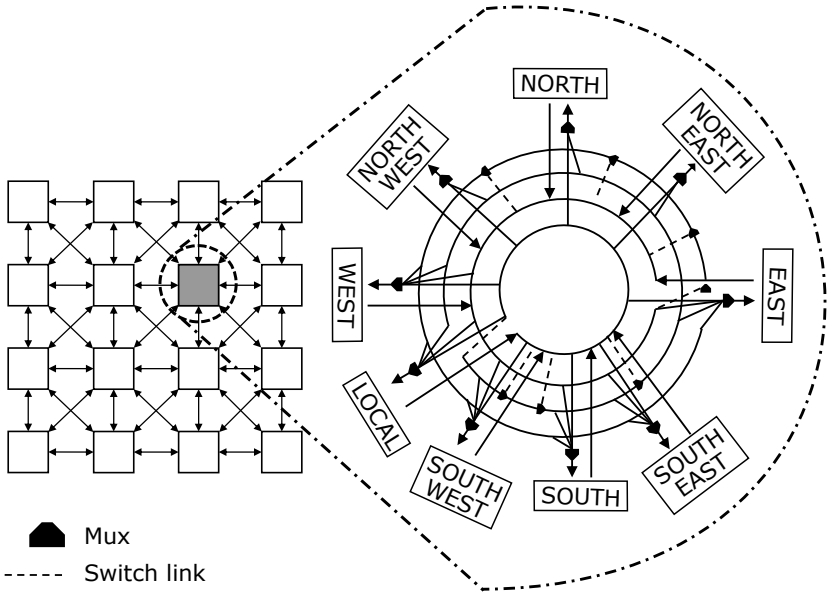


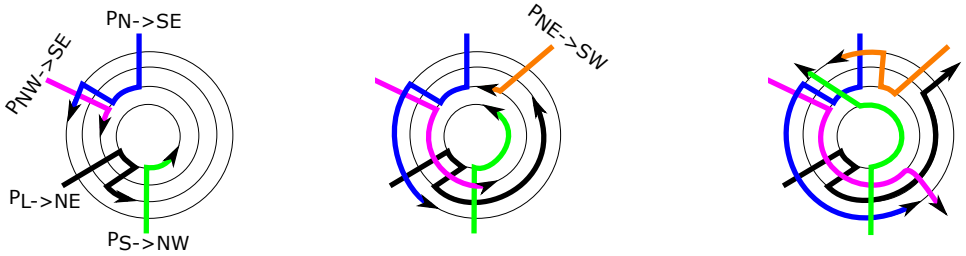
FIGURE 4.14: DMesh network topology and *R-NoC-DM* architecture

The next step is to determine the other input ports that can share the selected lane resources with the *south-west input port*. For this reason, if we assume that *line 5* selects the *local input port*, executing *line 6* results in the configuration shown in Fig. 4.13c. In Fig. 4.13c, the lane now has both blue and green labels, representing the valid output ports for packets flowing from the south-west input port and local input port respectively. Next step is to check if cycle exist in the lane. It is visible from the figure that both labels do not form a cycle on the lane. This is so as the path between the local output port and the local input port is not labelled. Therefore, the *south-west input port* can share the lane resources with the *local input port* without resulting to a deadlock configuration.

The Algorithm 1 control returns to *line 5* in order to find other possible input ports that can share the lane resource with the *south-west and local input port*. If the west input port is selected in *line 5* and the lane is labeled "orange" (i.e. *line 6*, then the resulting configuration is shown in 4.13d. It is visible from the figure that the labels form a cycle on the lane (i.e. all the paths on the lanes are labelled). Thus, sharing the lane between the south-west, local and west input port results to a deadlock. Lines *1 to 11* is repeated until all the input ports have been attached to a lane.

***R-NoC-DM* router for DMesh network topology**

Fig. 4.14 shows the DMesh network topology and the corresponding router (i.e. *R-NoC-DM*). As displayed in Fig. 4.14, the local, south-west, south, south-east input ports share similar lane resources (i.e. lane 0 and lane 2), while the east, north-east, north, north-west and west share lane 1 and lane 3. Thanks to the Algorithm 1, the generated router topology is deadlock-free.



(A) Low traffic (B) Medium traffic (C) High traffic

FIGURE 4.15: Packet flow scenarios in *R-NoC-DM*. $P_{X \rightarrow Y}$: a packet from X input port and destined for Y output port.

Figs. 4.15 shows traffic flow scenarios in *R-NoC-DM*. As shown in the figure, packets can switch lanes when their path is blocked by another packet. An example is in Fig. 4.15(a) where a packet from the north input ports switch to a secondary lane because the path is blocked by a packet flowing from the north-west input port. The router offers good resource utilization. This is illustrated in Fig. 4.15c, where almost all of the lane resources are utilized at very high traffic.

4.4 Discussion

As stated earlier, designing communication architectures with shared resources can be challenging due to their proneness to deadlocks. The deadlock problem is exacerbated in *R-NoC* due to its *ring-like/cyclic* topology and the use of wormhole flow-control. One possible means of avoiding deadlock in *R-NoC* is to utilize combined virtual-cut-through (VCT) and bubble flow-control as in the Rotary router [2]. However, VCT requires large buffering resources as buffers in VCT are allocated to packets instead of flits (as in wormhole). Large buffers introduce higher silicon-area overhead and power consumption. Hence it is not suitable for communication architectures with limited power/area budgets.

In order to avoid the power/area overhead associated with VCT, an algorithm for generating deadlock-free router configurations using wormhole flow-control is proposed. The algorithm is then applied to realize deadlock-free router configurations for the Mesh and Diagonally-linked mesh network topologies.

Chapter 5

Implementations of Roundabout Network-on-Chip

«"Action is the foundational key to all success."»

Pablo Picasso

Contents

5.1 Introduction

Having presented the *R-NoC* router concept in Chapter 4, this chapter provides its synchronous and asynchronous implementations. *R-NoC* concept relies on effective handshaking between individual components assembled in lanes. For this reason, its implementation is based on synchronous elastic design (instead of purely synchronous design using handshake FIFOs) and asynchronous logic. Synchronous elastic design is based on elastic buffer flow-control discussed in Chapter 2. EB flow-control eliminates the need for explicit input and/or output buffers at the router by using existing pipelined flip-flops in the channels to implement elastic FIFOs [17, 73]. On the other hand, asynchronous logic provides intrinsic flow-control mechanism which makes for a good fit for *R-NoC* implementation.

The remainder of this chapter is as follows: *R-NoC* synchronous implementation is provided in Section 5.2. Asynchronous design principles is presented in Section 5.3, while section 5.4 provides the asynchronous implementation of *R-NoC*. Section 5.5 summarizes the chapter.

5.2 Synchronous elastic implementation

This section presents the synchronous elastic implementation of the *R-NoC* router base on elastic buffers. First, an in-depth coverage of synchronous elastic circuits

is given. Next, the synchronous elastic implementation of the different blocks in *R-NoC* is detailed.

5.2.1 Elastic synchronous design

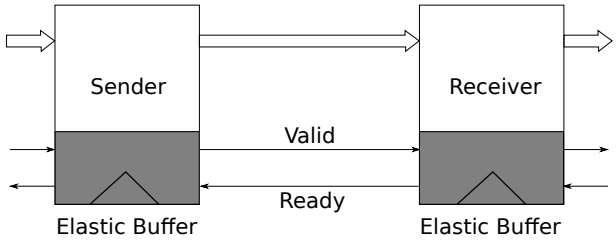


FIGURE 5.1: Synchronous elastic communication

Synchronous elastic communicating blocks, besides the data, carry extra signals that are required to implement the elastic handshake between sender and receiver [17, 29, 92]. Fig, 5.1 shows two elastic blocks (i.e. sender and receiver) that have two additional signals i.e. *valid and ready* required to ensure reliable data-flow in the communicating channel. The sender asserts the valid signal indicating the presence of a valid data on the channel. The receiver can store the data upon seeing the asserted valid signal. Similarly, if the receiver cannot accept a new data it needs to be able to inform the sender to stop sending data. For this purpose, the receiver de-asserts the *ready* signal. The sender stops sending data upon receiving the *ready* signal. Therefore, data transfer in this channel typically takes place when the sender is sending a valid data i.e. valid is asserted and when the receiver can accept a new data i.e. *ready* is asserted. Otherwise, the channel can be in either an idle or wait state. The channel is said to be in an idle state if the sender is not providing a data, while the channel is said to be in a wait state if the receiver is not ready to accept a new data, although the sender is providing a valid data. When the channel is in a wait state, the data on the channel must not change until the receiver can accept and store the data. If a new data is put on the channel when in a wait state, the previous data would be overridden [29].

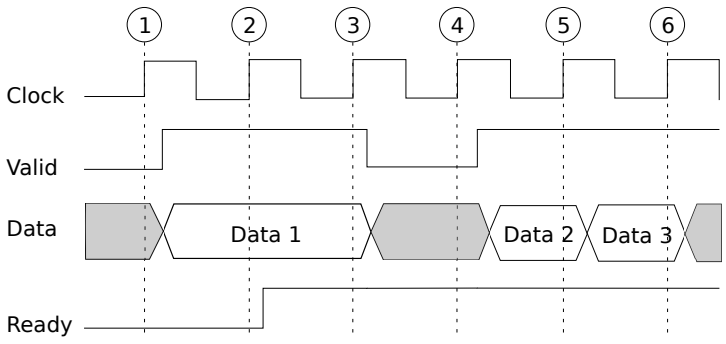
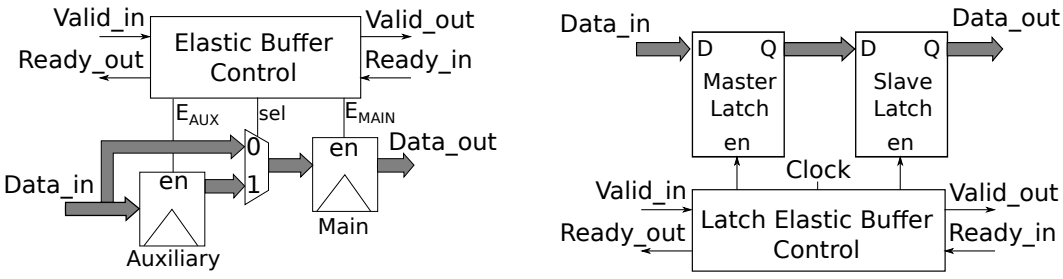


FIGURE 5.2: Example of synchronous elastic handshake

Fig. 5.2 shows an example of handshake communication between a sender and a receiver for 6 clock cycles. In cycle 1, the channel is in an idle state as the sender is not producing any valid data. Hence, the valid signal is 0. In cycle 2, the sender is producing a valid data but the receiver is not ready to receive the data. Hence, the channel moves into a wait state. It continues in the wait state until the receiver is able to read the data. This occurs in cycle 3 where both valid and ready signals are 1. Therefore, the channel is in a transfer state. Notice that the data is persisted until the receiver is able to store the data. The channel returns to an idle state in cycle 4. The channel moves to transfer state in cycle 5 and continues in that state in cycle 6 [17, 29]

An elastic channel needs to be able to store in-flight data during a stall. This is achieved by using a special kind of buffers known as elastic buffers (EB) that implement the elastic handshake protocol. Chains of EBs form a synchronous elastic pipeline that controls data-flow from one pipeline stage to another. Each EB in a synchronous pipeline has the capacity to store two data. When a synchronous pipeline stalls, the data stored in the EB during the stall must be kept for next cycle (assuming that it takes a cycle to propagate data from one pipeline stage to another). However, *in-flight* data from the previous EB must also be preserved to avoid losing data. The additional buffer slot is used for storing such an *in-flight* data during a pipeline stall. Therefore, a 2-slot EB can either be *empty* if no data is stored, *half-full* if it is storing only one data and *full* if it is storing two data.



(A) Flip-flop based EB

(B) Latch-based EB

FIGURE 5.3: Flip-flop and latch based EB implementations

Elastic buffer implementation

Although the synchronous implementation of *R-NoC* could be based on conventional FIFO queues with handshake capability, the efficient elastic buffer (EB) flow-control is chosen. EB flow-control eliminates the need for explicit input and/or output buffers at the router by using existing pipelined flip-flops in the channels to implement elastic FIFOs. Elastic buffers can be implemented using either edge sensitive flip-flop or level-sensitive latches. Fig. 5.3a shows the flip-flop based implementation of an EB. It consist of two edge-triggered flip-flops (i.e. *auxiliary* and

main). The EB control provides the *enable* signal to control which of the two flip-flops data is to be read from. As discussed earlier, the auxiliary flip-flop is used to store *in-flight* data during a stall operation. The *mux* is used to select data from either the main flip-flop or the auxiliary flip-flop. Data is written to the main flip-flop (the EB implements a bypass logic) during a write operation if the EB is empty else data is written to the auxiliary flip-flop.

An alternative EB implementation is shown in Fig. 5.3b. This uses level-sensitive latches in series, instead of edge-triggered flip-flops. It works on the principle that each edge-triggered flip-flop is made up of two level-sensitive latch (i.e. *master* and *slave* latches in Fig. 5.3b) that can store different data if they are controlled differently. The latch EB control generates the enable signals that are used to drive the latches. The master latch is transparent on the low clock phase, while the slave latch is transparent on the high phase of the clock. Each enable signal is emitted on opposite phase of the clock and remains stable during at latch active phase [17]. The latch-based EB provides lower area/power overhead and faster than the flip-flop based implementation [17]. On the other hand, the flip-flop based is more suitable for timing analysis and for field-programmable gate arrays (FPGAs) that do not support latches. The latch based implementation is preferred here because of its area, delay and power benefits over the flip-flop-based counterpart [17]

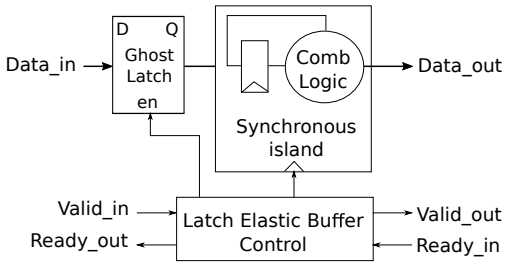


FIGURE 5.4: Elasticization of synchronous systems

Elasticization of synchronous circuits

Synchronous elasticity can be applied at different levels of granularity [41]. One of such application is to provide elastic communication among existing synchronous blocks. Fig. 5.4 shows the elasticization of existing synchronous block where the global clock of the synchronous block is replaced by the gated clock provided by the EB controller. A "ghost latch" is added to the input of the synchronous block to store data in-case of pipeline stall or back-pressure. This ghost latch has the same polarity with the synchronous block and does not introduce any additional delay since they are redundant during normal system operation [17, 41]. This is adhered to in the synchronous elastic implementation of *R-NoC*.

Another possibility is to apply elasticity at a very low granularity level, where each gate in a synchronous design is viewed as a separate elastic island and communicates with other gates using the elastic handshake protocol. Such finer-grain application of elasticity would incur a very high cost associated with each gate elastic control circuit [41]. A more intermediate application of elasticity is at the register transfer level (RTL), where each sequential element is associated with an elastic control and can communicate with other sequential element using the handshake protocol.

Several Network-on-Chip (NoC) routers using the elastic buffer protocol have been proposed [72, 73, 92]. EB routers do not use explicit input buffers so common in typical input buffered routers (IBR). The elastic chains of elastic buffers act as distributed FIFOs producing implicit storage. EB routers realize lower area and energy compared to IBR and they have far more simpler design [72, 73]. EB router was extended to support virtual-channels (VCs) [92]. This router is known as *ElastiStore*. ElastiStore minimizes the number of buffers per channel and realizes similar performance as typical VC router (without EB) but at a lower area and power cost since the number of buffers per channel is reduced.

5.2.2 *R-NoC* synchronous elastic building blocks

The *R-NoC* router is composed of several *building-blocks* that can be implemented using synchronous or asynchronous logic design style. First, the general building blocks are presented, then their synchronous implementation is given. The *R-NoC* router utilizes similar packet format regardless of the implementation style, therefore the packet format is also presented here.

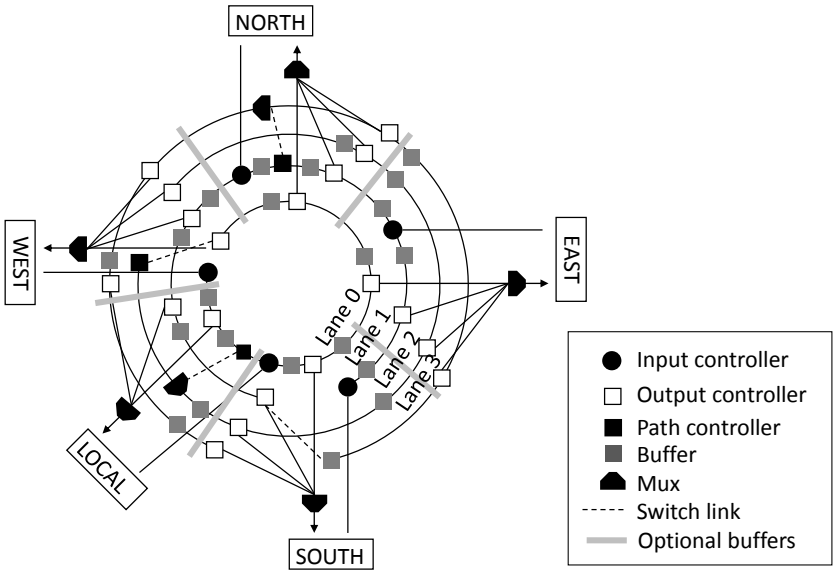


FIGURE 5.5: A 4-lane *R-NoC* topology

Fig. 5.5 shows a high level 4-lane *R-NoC* topology shown in Fig. 4.8. It consists of several sub-blocks such as controllers (input, output, path), buffers and mux. The buffers are distributed on the lanes as shown in Fig. 5.5. Thus, packets from several input ports can exploit the buffering resources on the lanes for performance benefits. Basically, when a packet enters the router, it is received by the input controller. The route computation and packet output port encoding take place here before the packet is forwarded to the lane. The path controller switches packet to a secondary lane if its path is blocked, while the output controller forwards a packet to the router output port if destined for the corresponding output, hence it is forwarded to the lane. *R-NoC* provides a highly distributed and adaptive topology allowing for additional lanes/ports to be added without incurring a significant increase in arbitration, area/power overhead. This is explored in Chapter 7

Packet format

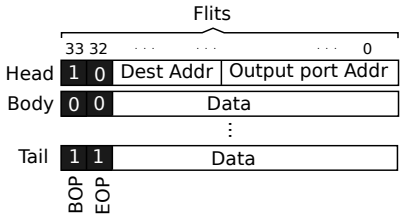


FIGURE 5.6: *R-NoC* packet format.

The router employs *wormhole flow-control*, where the buffering resources are allocated to individual *flits* of a packet. *R-NoC* packets consist of *N-flits* of 32 data-bits, encoded with two control bits (i.e. the begin-of-packet (BoP) and end-of-packet (EoP information) as shown in Fig. 5.6. A packet has three types of flits: head, body and tail. The head flit encodes the destination address and output port in the current router. The body/tail flits contain actual data. The *BoP and EoP* information identify the incoming flit type and help to allocate and deallocate network resources. Resources allocated by the head-flit are deallocated after the tail flit is transmitted.

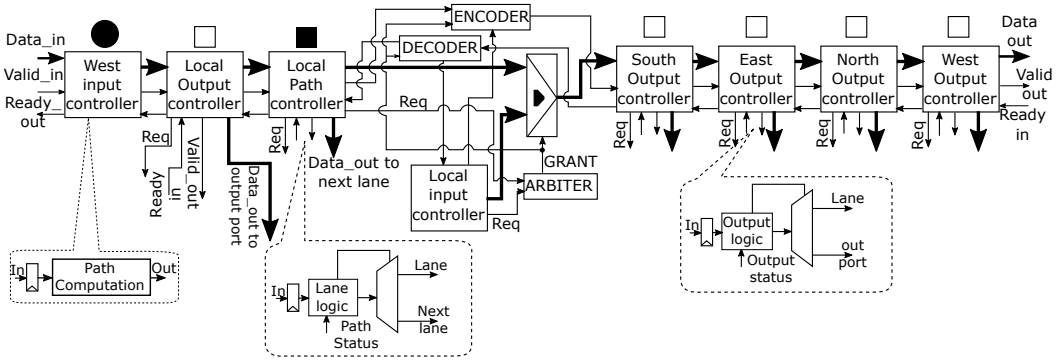


FIGURE 5.7: *R-NoC Lane 0* pipeline

Lane pipeline

The entire pipeline for *lane 0* is shown in Fig. 5.7. The blocks (i.e. input, output, lane controllers) make up the individual pipeline stages and communicate via *ready/valid* handshake protocol. Each block incurs only a cycle each, hence the entire pipeline is only *6 clock cycles* for the longest path on the lane (i.e. from west input port to north output port).

Input controller

As depicted also in Fig. 5.7, the input controller block consists of an EB and a *path computation* (PC) block. The area overhead of this block is minimal and this block incur only one clock-cycle as discussed earlier in this chapter. When a flit arrives at the input port of the router, it is forwarded to the PC block. The PC block is responsible for computing the packet output port using *XY-routing*. If the flit is a header-flit, the PC block decodes the packet destination address encoded in the flit and uses this information to compute the packet output port in the current router. The output port routing information is encoded in the header-flit of the packet before it is transmitted. The other flits follow the path already reserved by the header-flit since only the header-flit contains the routing information.

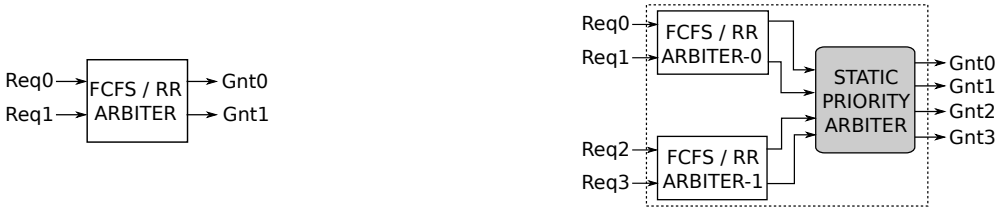
Output and path controllers

The output controller is also displayed in Fig. 5.7. It consists of an elastic-buffer, an *output logic* block and a *demux*. When a flit is received, it is forwarded to the output logic block. Depending on the packet output information encoded in the packet header and the output status, the block selects one of two possible paths to forward the packet.

1. output port address matches and output port is free;
2. output port address matches but output port is busy and is not *local*;
3. output port address matches but output port is busy and is *local*;
4. output port address does not match;
5. output port address matches and output port controller is located on the secondary lanes, i.e., *Lane 2 or Lane 3* shown in Fig. 5.5.

For cases (1), (3) and (5) the path labelled "out port" is selected and the flit is forwarded to the output port. However, the flit is forwarded to the lane for case (3) if the output port controller is located on *lane 1* (see Fig. 5.5). For cases (2) and (4), the path labelled "Lane" is selected and the flit is forwarded to the lane. The reserved

path is kept active for the other flits transmission. The *path controller* is similar to the output controller in terms of functionality. It forwards a packet to the lane if the packet path is not blocked. Otherwise, it switches the packet to a secondary lane.



(A) FCFC/RR arbiter

(B) Static priority arbiter

FIGURE 5.8: First-come-first-serve (FCFS) /round-robin (RR) and static-priority arbiter

Arbitration

An arbiter is required to grant access to shared network resources. Two kinds of arbiters are used in *R-NoC* to grant access to shared resources i.e. lane buffers and the router output ports. A *2 input-request* arbiter is used to control access to shared lane resource as displayed in Fig. 5.8a. This arbiter operates in two modes i.e. first-come-first-served mode for sequential requests and round-robin mode for simultaneous requests. An example is shown in Fig. 5.7, where arbitration between the local lane controller and the local input controller is provided by an arbiter to control access to the next shared lane resource (i.e. south output controller buffer). For this purpose, the arbiter operates in a *first-come-first-serve* (FCFS) manner or *round-robin* (RR) if request from both controllers are simultaneously asserted.

On the other hand, a static priority arbiter is used to control access to *R-NoC* output port as shown in Fig. 5.8b. This arbiter is composed of two *FCFC/RR* arbiters and a *static priority arbiter* that grants the output request base on *priority*. The *FCFC/RR arbiter-0* arbitrates between requests coming from two *primary lanes* (i.e. lane 0 and lane 1), while the *FCFC/RR arbiter-1* arbitrates between requests coming from two *secondary lanes* (i.e. lane 2 and lane 3) using policies described previously. As shown in Fig. 5.8b, the outputs of the two *FCFC/RR* arbiters are fed into the *static priority arbiter*. Note that only two inputs (to the *static priority arbiter*) can be asserted simultaneously (i.e. outputs of *FCFC/RR arbiter-0* and *FCFC/RR arbiter-1*). The output port is granted depending on the priority of the request. A request from *FCFC/RR arbiter-1* is given priority over a request from *FCFC/RR arbiter-0*, even in cases of simultaneously requests, since a request from *FCFC/RR arbiter-1* represents output port request from a controller located on the secondary lane. The arbiters are implemented using *FSM* with a combinational output (Mealy FSM), hence it can process requests as soon as they arrive without waiting for a clock edge.

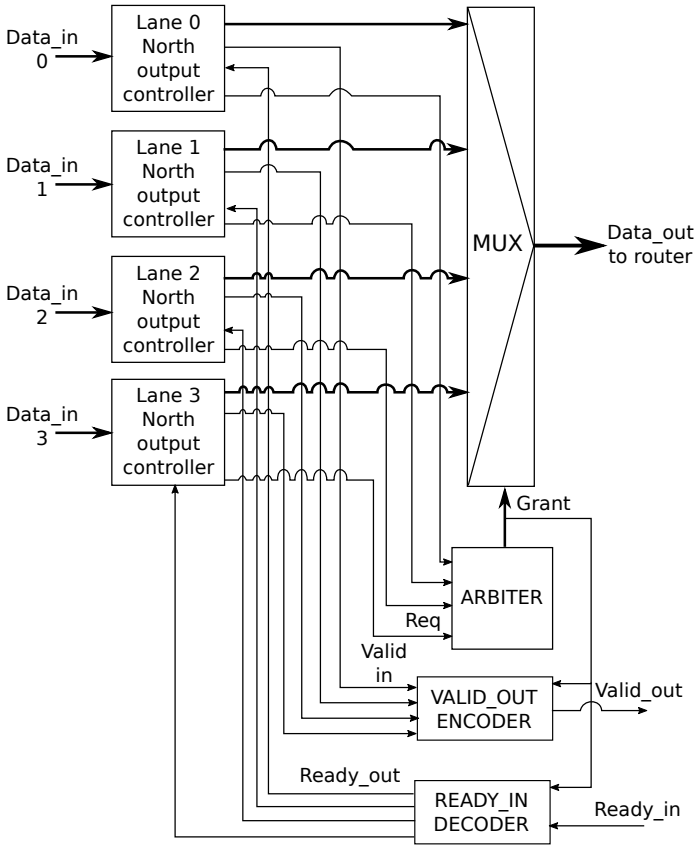


FIGURE 5.9: *R-NoC output port block*

Output port structure

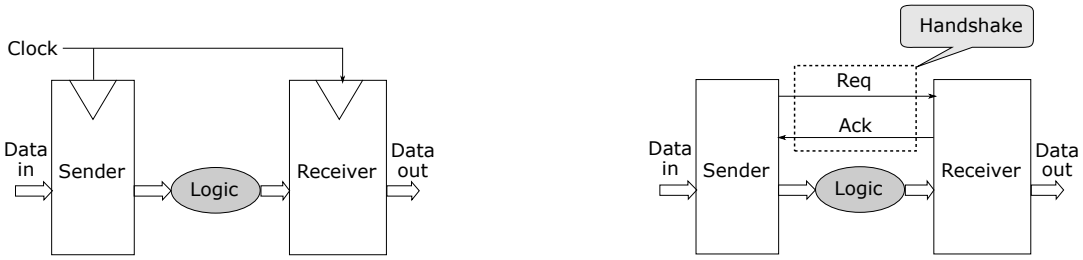
Fig. 5.9 shows *R-NoC north output port module* which is typical of all the output ports in the router. The block receives data-flows from each north output controller located on lane 0 to lane 3. As displayed in the figure, the *grant* signal from the *Arbiter* is used to *select* one of the four data-flows that are inputs to the *Mux*. The selected data is forwarded to the *input controller* (precisely south input controller) of the adjacent router. Remembering that each block communicates using the *ready/valid* handshake protocol, the output port block must ensure the following for the correct functioning of the circuit:

- only the *valid* signal of the output controller that has been granted access to the output port is sent to the receiver
- the receiver's status (i.e. indicated by the *"ready_in"* signal) is communicated to only the output controller that has been granted access to the output port

For this reason, an *Encoder* is needed to generate the correct *valid_out* signal, while the correct *ready_out* signals is generated using the *Decoder* circuit as shown in Fig. 5.9. A possible implementations of the blocks can be found in [53].

5.3 Asynchronous implementation

The role of the clock in a synchronous circuit is to define time instances where a signal is stable and valid i.e. all the interacting components assume a common notion of time [96]. Unlike synchronous circuit, asynchronous circuits belong to the class of sequential circuits that do not rely on a clock for synchronizing events of communicating blocks [96].



(A) Synchronous communication (B) Asynchronous communication

FIGURE 5.10: Synchronous vs. Asynchronous communication

Fig. 5.10a shows, as an example of synchronous circuit, a 2-stage synchronous pipeline where data flows sequentially from one memory element (i.e. *Sender*) to another (i.e. *Receiver*). As shown in Fig. 5.10a, a global clock is used to synchronize the movement of data in this pipeline circuit. Conversely, an asynchronous circuit (also known as a self-timed circuit) relies on *local handshaking* for controlling the movement or flow of data in the circuit.

This is depicted in Fig. 5.10b, where a specific class of asynchronous protocol is used to control data-flow in the asynchronous pipeline. The handshaking protocol is similar to the synchronous *read-valid handshake* discussed previously. Here, the communication proceeds as follows: the sender issues data and assert the *request* signal if the data is valid. Upon receiving the data and request signal, the receiver stores the data and asserts the *ack* signal. The sender can then initiates the next data-transfer request. On the other hand, if the *request* signal is de-asserted, the receiver cannot store the data since the data is invalid. Similarly, the sender cannot initiate the next data transfer if the previous data transfer has not been acknowledged by the receiver. In general, a stage can accept a new data if its left neighbouring state is providing a new data and its right neighbouring state has successfully stored the previous data [96, 77].

Data transfer in asynchronous circuit occurs between communicating blocks using bundles of wires known as channels [6]. The handshake protocol can be used for data transfer or to synchronize circuits. An asynchronous channel for data transfer can either be a *push* or *pull channel* depending on the flow of data relative to the request signal as shown in Fig. 5.11. In a push channel, data transfer request is initiated by the sender and the direction of data-flow and request are the same. On the

(A) Handshake push channel (B) Handshake pull channel

FIGURE 5.11: Handshake push vs. pull channel

other hand, in a pull channel, the receiver initiates the data transfer and the direction of data-flow and acknowledgement are the same [6, 86].

Subsection 5.3.1 introduce the general concept of asynchronous logic design protocols. Subsection 5.3.2 discusses the *Bundled-data* protocols used in the MANGO [11] and QNoC [30] NoCs. Subsection 5.3.3 discusses a class of asynchronous protocols known as *delay-insensitive (DI) protocols.* It presents the *4-phase dual-rail* DI protocol used in Hermes-A [85], ANoC [7], FAUST [5] NoCs, and *Level-Encoded Dual-rail (LEDR)* protocol used in Onizawa [78].

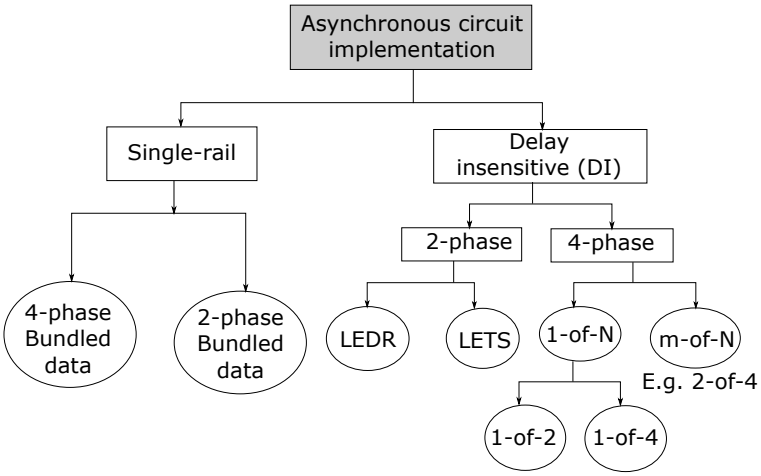


FIGURE 5.12: Asynchronous circuit implementation

5.3.1 Asynchronous circuit design

The goal of this subsection is to present some of the well-known asynchronous handshake protocols and their practical implementation in Network-on-Chip router. The focus is not to present all the principles of asynchronous circuits but to give a general overview that will enable the reader to appreciate and evaluate our proposal and that of the related works.

Depending on the data-encoding scheme, an asynchronous circuit can be implemented either as *single or delay-insensitive (DI)* circuit as shown in Fig. 5.12. In single rail implementation style, data is encoded with conventional single rail boolean logic, while multiple rails are used to encode data in DI circuit implementation [96].

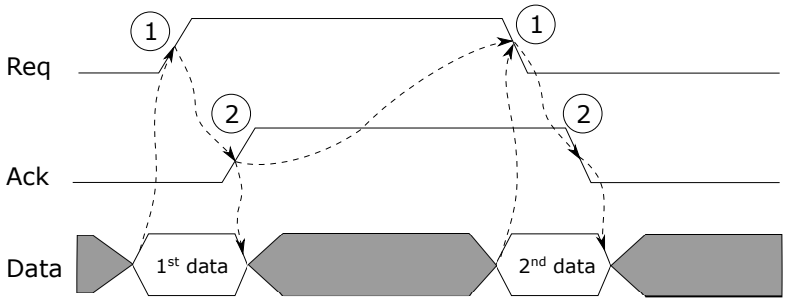


FIGURE 5.13: Two-phase handshake protocol control signaling

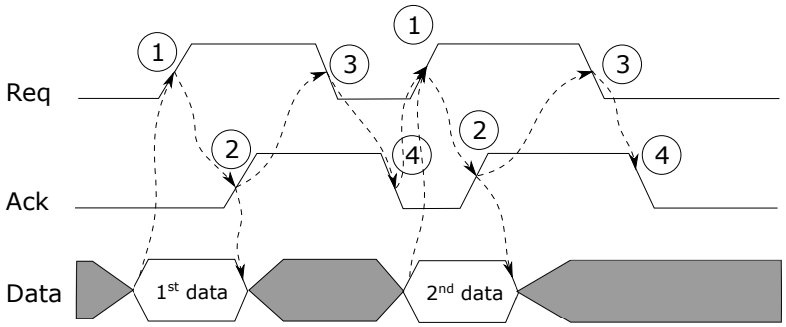


FIGURE 5.14: Four-phase handshake protocol control signaling

Handshake protocol control signaling

Generally, asynchronous handshake protocols control signaling can occur in *2-phase* or *4-phase*. This is true for both the Bundled-data (where data is bundled with separate *req* and *ack* wires) and DI protocols. An example of both will be provided for only Bundled-data protocol since it is also applicable to the DI. The difference is mainly in their data encoding. Fig. 5.13 shows an example of data transfer using the *2-phase* control signaling. In the *2-phase* mode, the *Req and Ack* make one transition each per data transfer. Also, there is no *return-to-zero* phase where the control signals are reset before the next data transfer can be initiated. Contrary to the *2-phase* handshake signaling, the *4-phase* requires four wires transition events for each data transfer transaction. Also, each control signals must *return-to-zero* before the next data transfer.

In General, the *4-phase* protocol is most common due to its ease of design compared to the *2-phase* counterpart [96]. The *4-phase* uses *return-to-zero* which introduces lower throughput (control signals are reset to zero before the next data transfer transaction can be initiated) and higher power (switching all the control signals to zeros before next transaction increases circuit switching activities) compared to the *2-phase* protocol. Conversely, the *2-phase* protocol implementation could lead to faster circuits with lower power consumption. However, implementing control circuits for the two phase protocol is complex since every wire transition represent the presence of valid data in the channel and the receiver is expected to respond

accordingly (be able to respond to the events) [96].

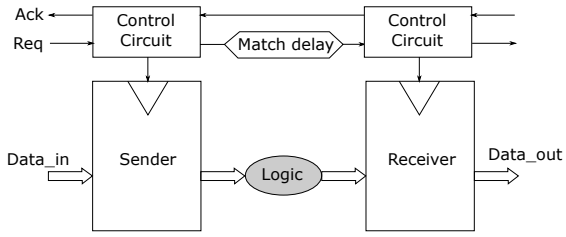


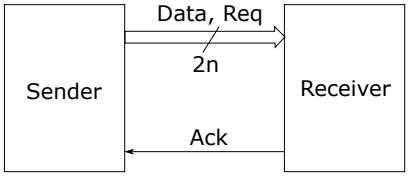
FIGURE 5.15: Bundled-data asynchronous protocol

5.3.2 Bundled-data protocol

The *Bundled-data* asynchronous protocol uses one wire to represent one bit of information to be transferred on an asynchronous channel, where additional wires (req and ack) are bundled with the data. The bundled-data protocol is similar to a conventional synchronous circuit in terms of data encoding (i.e. both uses 1 wire to represent one bit of data) and in structure. In bundle data, the global clock is replaced by handshaking control circuits as displayed in Fig. 5.15. A possible implementation of the control circuit is given in [96]. Since the synchronous clock has been replaced by local handshake, timing information is carried on two separate signals known as the *control signals* i.e. the *req* and *ack* signals. The *req* enables a sender to indicate the availability of valid data while the *ack* signal enables the receiver to indicate its readiness to receive and store new data [96, 6]. To ensure proper functioning of the bundled-data circuit, a delay element (i.e. *matched delay* in Fig. 5.15) is added between registers for all computation blocks in the design i.e. *logic* in Fig. 5.15 and the delay of the computation block must be smaller than the match delay. This ensures that the receiving register does not receive the *req* signal too early i.e. before a valid data is produced at the output of the computation block. If *req* is asserted too early, the receiving register might latch intermediate (invalid) data produced by the computation block.

Network-on-Chip routers based on Bundled-data protocol

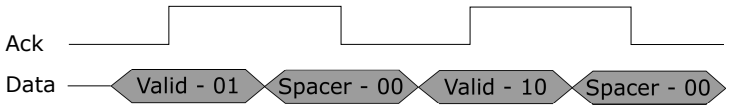
Several Network-on-Chip (NoC) routers in the literature have been implemented using the *Bundled-data* protocol [11, 30, 45]. The Mango [11] clock-less router is based on the 4-phase Bundled-data asynchronous protocol. The Mango NoC router based on *120nm CMOS technology* provides a reasonably high throughput and a small silicon area-overhead since it is based on Bundled-data protocol which leads to reduced area overhead. However, as will be discussed next, the bundled data protocol is not timing robust and subject to temperature, voltage variations which can affect signal integrity [86].



(A) Communicating channels

Encoding		Value
d_0	d_1	d
0	0	Reset/Empty
0	1	Valid 0
1	0	Valid 1
1	1	Not used

(B) Data encoding



(C) Data transition

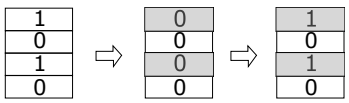
FIGURE 5.16: 4-phase dual-rail asynchronous protocol

5.3.3 Delay-insensitive protocol

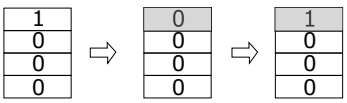
Unlike the Bundled-data protocol, the *delay-insensitive (DI)* protocol is timing robust ensuring that two asynchronous blocks can reliably communicate even when delays are present in the wires connecting them [96]. Fig. 5.16a shows communication using the *4-phase dual-rail* protocol. This protocol does not use separate *Req* signal to indicate data validity. Instead, the *Req* is encoded instead the actual data to be transferred on the channel. This protocol requires two wires (i.e. d_0 and d_1 in Fig. 5.16b) to encode *1 bit* of data to be transmitted. Hence, sometimes called *dual-rail 1-of-2 protocol*. As shown in Fig. 5.16a, *2xn* wires are needed to transmit *n-data* through the channel. Fig. 5.16b shows data encoding for this protocol. For example, a valid logic bit *1* is encoded as *"10"*. This protocol forbids the transition from one valid data to another. Hence, each valid data is separated by an empty data or spacer. This is depicted in Fig. 5.16c.

Encoding				Value
d_3	d_2	d_1	d_0	d
0	0	0	0	Reset/Empty
0	0	0	1	00
0	0	1	0	01
0	1	0	0	10
1	0	0	0	11
Others				Not used

FIGURE 5.17: Four-phase 1-of-4 protocol data encoding



(A) Example of *1-of-2*



(B) Example of *1-of-4*

FIGURE 5.18: Transition of two logical bits from (1,1) to (1,1) using dual-rail 1-of-2 and 1-of-4 protocol. Shaded box represent signal transition

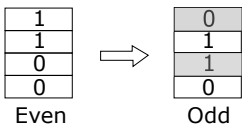
The *4-phase 1-of-4* protocol is similar to the *4-phase 1-of-2* in terms of data transitioning. However, contrary to the *1-of-2* protocol, this protocol requires 4 wires (i.e. d_0 to d_3 in Fig. 5.17) to encode *1 bit* of data to be transmitted. Fig. 5.18 shows the transition of two logical bits (i.e. from [1,1] to [1,1]) using the *1-of-2* and *1-of-4* protocols. A total of 4 wire transitions occur for *4-phase 1-of-2* as shown in Fig. 5.18a, while only two wire transitions is observed for *4-phase 1-of-4*. Hence, *4-phase 1-of-4* provides lower power consumption (due to fewer signal transitions) compared to the *4-phase 1-of-2*. However, the *1-of-2* is mostly used due to its lower implementation complexity. The whole class of *1-of-n* handshake protocol get very expensive beyond *1-of-4*.

Network-on-Chip routers based on 4-phase dual-rail protocol

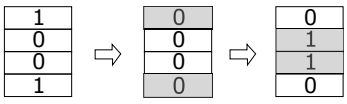
Several routers using the *4-phase dual-rail* protocol exist in the literature [85, 7, 5, 99]. Similar to the *R-NoC* (asynchronous implementation), the Hermes-A router [85] uses distributed routing and independent arbitration at the router ports. The Hermes-A router [85] achieves a smaller area-overhead compared to *R-NoC* due to multiple lanes in *R-NoC*. However, the Hermes-A router [85] does not support shared-buffer like the *R-NoC*.

Encoding			Value
Phase	Data rail	Parity rail	d
Even	0	0	0
	1	1	1
Odd	0	1	0
	1	0	1

FIGURE 5.19: Level-encoded dual-rail (LEDR) encoding



(A) Example of *LEDR*



(B) Example of *1-of-2*

FIGURE 5.20: Transition of two logical bits from (1,0) to (0,1) using LEDR and 1-of-2 protocol. Shaded box represent signal transition

Level-Encoded Dual-Rail (LEDR) protocol

Contrary to the *4-phase dual-rail* protocol that requires resetting all bits to zero (i.e. return-to-zero) before the next data transfer can be initiated), *Level-Encoded-Dual-Rail (LEDR)* is a *2-phase* protocol as it completely eliminates the return-to-zero phase associated with the *4-phase* protocol [26]. Similar to *4-phase 1-of-2* protocol, LEDR uses two wires to encode *1-bit* of data to be transmitted. This is shown in Fig. 5.19, where one wire is used as the *parity-rail* and the *data-rail* carries actual data. LEDR

is an alternating phase protocol requiring data to transition from one phase to the other. Fig. 5.20 shows the transition of two logical bits (i.e. from [1,0] to [0,1]) using the *LEDR* protocol and *4-phase dual-rail 1-of-2* protocol. As shown in Fig. 5.20a, only one rail changes per 1-bit transition in LEDR, while for each logic bit transition in *1-of-2*, two rails changes due to the additional *return-to-zero* step. In the example shown in Fig. 5.20, a total of four rails change is observed for the transaction (i.e. 1-of-2 protocol) as shown in Fig. 5.20b. LEDR leads to higher throughput and lower power since the communication step is reduced by half and has fewer rails change per transaction compared to the *1-of-2*. However, it is more complex to realize circuits using the 2-phase protocol [96].

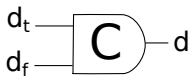
Network-on-Chip router base on LEDR

A Network-on-Chip (NoC) router based on LEDR has been proposed [78]. The LEDR based router realized a throughput that is almost twice that of their 4-phase implementation of the same router and dissipates less energy per-flit. Surprisingly, in terms of *head-flit* latency, the *1-of-2* based router achieves a shorter head-flit transmission time and reduced power dissipation compared to the LEDR router. According to the authors, the reason for this is linked to the *pipeline-latch* implementation. The Level-Encoded-Transaction Signaling (LET) is an extension of LEDR. Hence LEDR can be seen as *1-of-2* LETS. Compared to LEDR, it has an additional benefit of lower power consumption since it uses fewer wire transitions for each data-transfer [68].

5.4 *R-NoC* delay-insensitive implementation

The asynchronous version of the *R-NoC* router is implemented using the *4-phase dual-rail* protocol described in Section 5.3. Delay insensitivity and lower implementation complexity [96] motivate the choice of using this protocol. Unlike synchronous circuit implementation, asynchronous designs provide intrinsic flow-control properties that make for a perfect fit for *R-NoC* that heavily relies on dynamic internal flow routing. This subsection provides an in-depth *4-phase dual-rail* implementation of the different blocks used in *R-NoC* topology shown in Fig. 5.5.

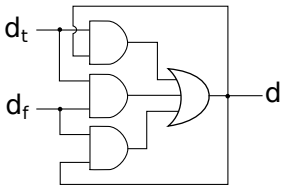
The *muller gate* or *c-gate* is extensively used in asynchronous design. The symbol for a *2-input* c-gate is shown in Fig. 5.21a where, d_t and d_f are the input and d is the output of the gate. The output of the c-gate is indicated (output is produced) only when both input events have taken place. The truth table for a 2 input c-gate is given in Fig. 5.21. The output of the gate indicates "0" when both inputs are "0" and a "1" when both inputs are "1". The previous output value is retained for other input combinations. A possible implementation of a 2-input c-gate is given in Fig. 5.21c.



(A) C-gate symbol

d_t	d_f	d
0	0	0
0	1	Hold
1	0	Hold
1	1	1

(B) C-gate truth table



(C) C-gate implementation

FIGURE 5.21: Muller gate symbol, truth table and gate-level implementation

```
1  --use IEEE.std_logic_1164.all;
2  --use ieee.numeric_std.all;
3  --
4  entity cgate2to1 is
5      Port (
6     i0  : in  STD_LOGIC;
7     i1  : in  STD_LOGIC;
8     o   : out STD_LOGIC;
9     Reset : STD_LOGIC
10   );
11 end cgate2to1;
12
13 architecture cgate2to1_arch of cgate2to1 is
14  signal and0_out, and1_out, and2_out, or_out : STD_LOGIC;
15 begin
16
17  process (Reset, i0,i1)
18  begin
18   if(Reset = '1') then
20    and0_out <= '0';
21    and1_out <= '0';
22    and2_out <= '0';
23    or_out <= '0';
24   else
25    and0_out <= i0 AND or_out;
26    and1_out <= i0 AND i1;
27    and2_out <= i1 AND or_out;
28    or_out <= and0_out OR and1_out OR and2_out;
29    o <= or_out;
30   end if;
31  end process;
32
33 end cgate2to1_arch;
```

FIGURE 5.22: VHDL description of a C-gate

Fig. 5.22 displays an HDL description of a typical 2 inputs C-gate used in the design. Although implementing the entire asynchronous design (using HDL) does not result in an optimal implementation, it has been used for faster design evaluations.

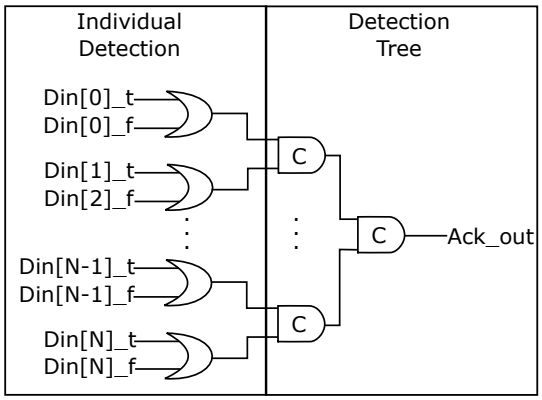


FIGURE 5.23: 4-phase dual-rail completion detector

As stated earlier, the *4-phase dual-rail* protocol uses 2 wires to encode *1-data bit* to be transmitted and each valid data is separated by empty data or spacers. A receiver detects valid data when all the wire pairs are not empty. This is accomplished with the completion detection circuit shown in Fig. 5.23. This circuit is composed of *OR-gates* used for detecting the validity of individual dual-rail code and a tree of *c-gates* used to synchronize the individual detection and to signal a valid data reception. The output of this circuit gives a logical "1" only when all wire pairs contain valid data and a logical "0" when all wire pairs are empty. Hence, it is widely used in DI protocol to signal the receipt of valid or empty data.

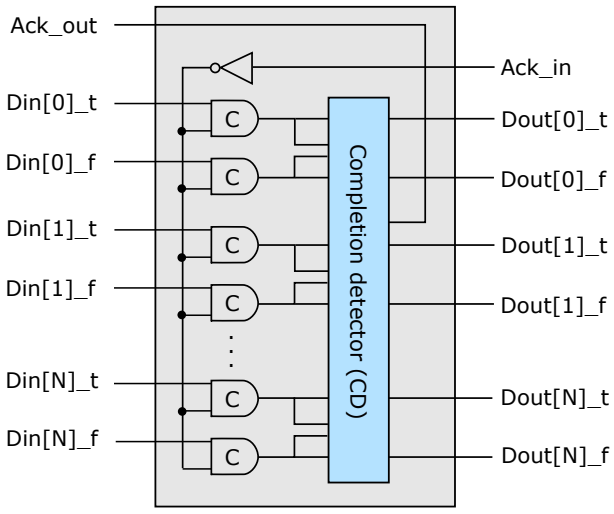


FIGURE 5.24: 4-phase dual-rail half-buffer

Fig. 5.24 shows a 4-phase dual-rail buffer. This buffer is known as *half-buffer* because it can store only one token (either a valid data or a spacer). A buffer that can store both valid data and spacer is known as *full-buffer*. The buffer is used as

temporary data storage and communicates with neighbouring buffers using the 4-phase dual-rail handshake protocol.

5.4.1 4-phase dual-rail input controller

Fig. 5.25 shows the router input controller structure. When a flit arrives at the router input, the *fork* block reads the flit type information encoded in the flit. The *encoder* uses the information to generate a *select* signal for the *demux*. If the flit type is "head", it is forwarded to the *path computation* (PC) block. Other flit types i.e. body and tail flits are forwarded out of the controller since only the head-flit contains the routing information. A routing algorithm is applied in the PC block to compute the packet output port destination in the current router. Each output port in *R-NoC* has a unique *1-bit single-rail address*. The output port information is encoded in the flit and then it is transmitted. The PC block is similar to the one proposed in [85].

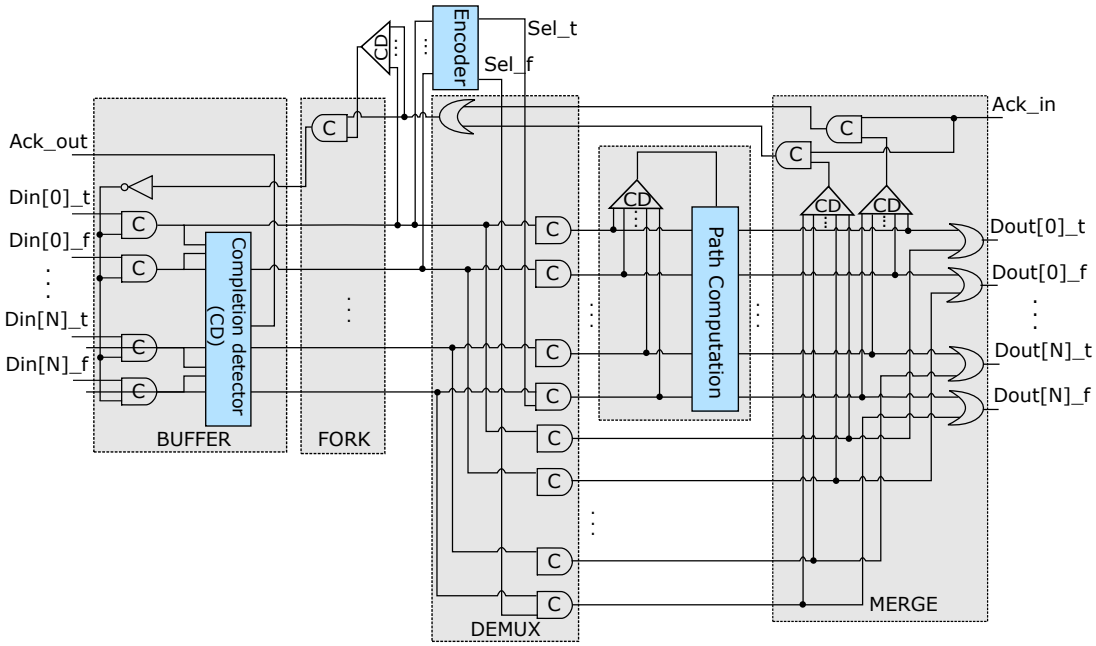


FIGURE 5.25: *R-NoC* input controller

5.4.2 4-phase dual-rail output and path controllers

Fig. 5.26 shows *R-NoC* output controller structure. When a flit is received, the first *demux* selects one of two paths to forward the flit depending on the flit type. If a head flit is signaled, it is forwarded to the *output logic* block. Otherwise, the flit is forwarded to the lane. When a valid head-flit is detected at the *output logic* block (i.e. output of *completion detection* circuit is '1'). The block checks if the output port address (encoded in the flit) matches the current output port and decides where to forward the flit. The reserved path must be kept for the other flits transmission.

Hence, last *demux* is purely combination allowing for transmitting both valid and spacer data while keeping the *select* signal of the *demux* active. The *path controller* is similar to the output controller in terms of functionality and circuitry. It forwards a packet to the lane if the packet path is not blocked on the primary lane. Otherwise, it switches the packet to a secondary lane.

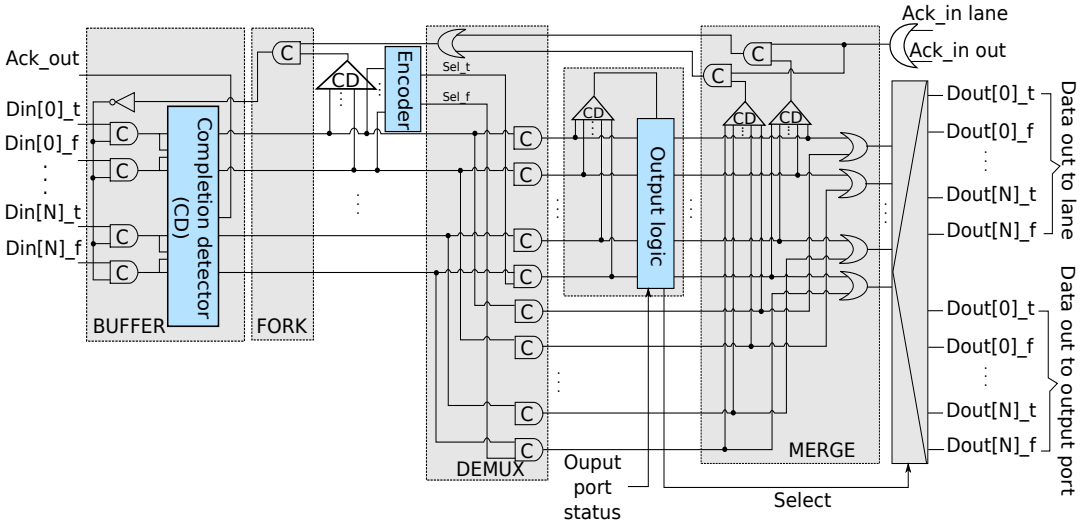


FIGURE 5.26: *R-NoC* output controller

5.5 Summary

R-NoC has been implemented using two alternatives design styles: synchronous elastic (not purely synchronous) and asynchronous because of the underlying specific intricate flow-control mechanism of *R-NoC*. The synchronous elastic implementation is based on elastic buffers that uses existing pipelined flip-flops in the channels to implement elastic FIFOs [17, 73]. Elastic buffers can be implemented using either edge sensitive flip-flop or level-sensitive latches. The latch based implementation is preferred here because of its area, delay and power benefits over the flip-flop-based counterpart [17]. The asynchronous implementation of *R-NoC* is based on the *4 phase dual-rail* protocol since the protocol is delay insensitive and allows for less complex implementation compared to the *2 phase* delay insensitive protocol [96].

Chapter 6

Evaluation of Roundabout Mesh Network topology

«"Stop chasing the money and start chasing the passion."»

Tony Hsieh

Contents

6.1 Introduction

This chapter presents the evaluation of the synchronous elastic (*R-NoC-SE*) and asynchronous (named *R-NoC-A*) versions of *R-NoC*. The evaluations are carried out at router and network levels using the metrics discussed in Chapter 2. Besides performance, area and power results are also presented. In addition, this chapter provides a quantitative comparison of *R-NoC-SE and R-NoC-A* with existing synchronous and asynchronous solutions respectively, in order to show the benefits of the proposal.

Fig. 6.1 shows the *R-NoC-SE* design flow. It begins with laying down the specification of the system. Then, a register transfer level (RTL) description of the design is produced in VHDL and simulated using Mentor Graphics ModelSIM simulator [43]. The functionality of the design is validated using *testbench* described in VHDL as shown in Fig. 6.1. If the design is not functionally correct, the flow returns to the previous stage (i.e. RTL description). This process is iterative and continues until a functionally correct design is produced. On the other hand, if the design is functionally correct, the performance results such as *latency and throughput* discussed in Chapter 2 is produced at router/network level. The next step is to produce the gate-level description of the design (i.e. *netlist*, which is basically a connection of gates that makes up the design) so as to be able to obtain more realistic evaluations contrary to using high-level area/power estimators. For this reason, the functional VHDL description of the design is fed into the synthesis tool which then produces

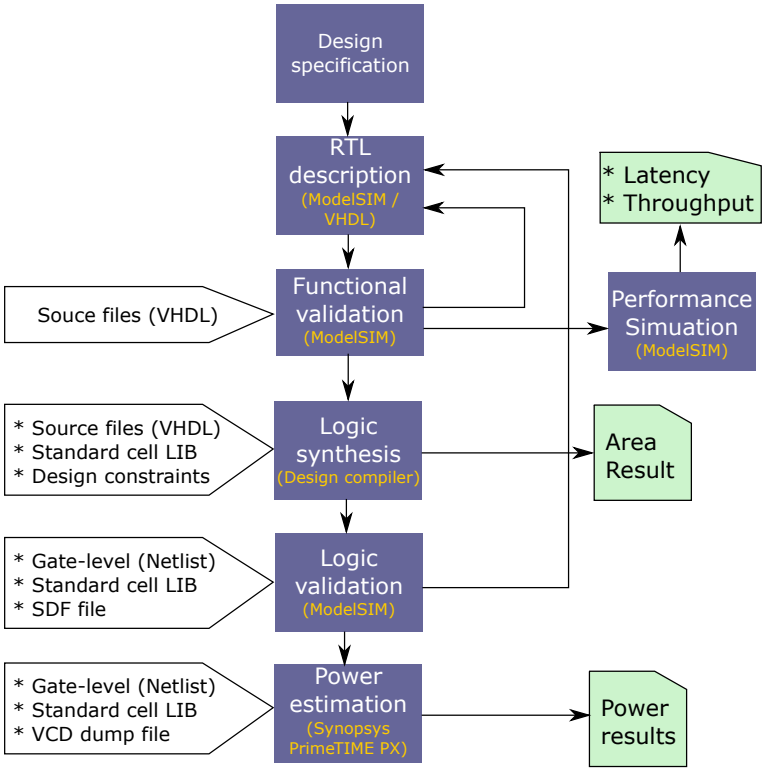


FIGURE 6.1: *R-NoC-SE* elastic design flow

the design netlist. Two synthesis tools were considered: Synopsys Design Compiler [97] for 180nm CMOS standard cell library and Cadence RTL Compiler [16] for 45nm CMOS cell library.

The produced netlist is validated using VHDL testbench in ModelSIM environment to ensure it is functionally correct. Note that area results are generated from the functionally correct design netlist. In order to produce the power result, the design netlist, standard cell library and value change dump (VCD) file is fed into *Synopsys PrimeTime PX* as shown in Fig. 6.1. The VCD file contains information about the switching activities or signal changes in the design during simulation. Power results are obtained only for 45nm CMOS cell library. The described design flow is similar to that of the asynchronous implementation (i.e. *R-NoC-A*). The main difference is that in the asynchronous case, the router blocks were back-annotated with delay values from the synthesis tools for performance simulations. This is because asynchronous logic works on the principle of propagation delay of individual gates since there is no clock.

The reminder of this chapter is as follows: section 6.2 presents the evaluation of the synchronous routers and networks, while section 6.3 presents the evaluation of the asynchronous routers and corresponding networks.

6.2 Synchronous elastic evaluation

This section evaluates the synchronous implementation of *R-NoC-SE* router and network. This section also explores further *R-NoC-SE* topologies in terms of area and performance. *R-NoC-SE* is also compared to existing wormhole, virtual-channel and single cycle routers.

6.2.1 Synchronous elastic router

This subsection presents the performance and area evaluation of *R-NoC-SE*. In terms of performance, it provides insight into the impact of lane switching on packet latency. The power results for different paths taken in the router is also presented here.

Performance and Area Evaluation

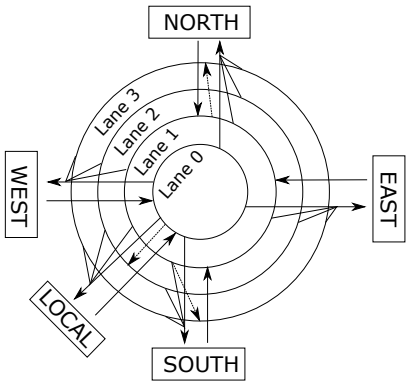


FIGURE 6.2: A conceptual 4-lane *R-NoC* topology

For *R-NoC-SE* area and head-flit latency evaluation, the *4-lanes* router version (i.e. base *R-NoC router*) with 5 input/output ports shown conceptually in Fig. 6.2. A more detailed figure and description of router topology have been provided in Chapter 5.

TABLE 6.1: Performance of 34-bits *R-NoC-SE routers*.

Head-flit latency (cycles)		Operating freq. (MHz)	Total area (mm^2)
Min	Max		
2	6	650	0.03

In Table 6.1, the result for the min and max head-flit latencies are displayed. The min and max head-flit latencies correspond to latencies incurred when the head-flit takes a short path (e.g. west input to local output port in Fig. 6.2) and a long path (e.g. west input to north output port in Fig. 6.2) on a given lane in the router. Thanks to the short router pipeline, the head-flit latency for the router is only 2 clock cycles

for short path (a cycle each for input and output controllers). Output port arbitration is performed in the same cycle when output request is sent by output controllers.

Compared to most typical input buffered NoC router, the router pipeline for a short path is twice less. As an example, the Hermes [74] router has a router pipeline of 5 clock cycles which is only one clock cycle less than the pipeline cycle period for the longest path on a given lane in *R-NoC-SE*. The 4 or 5 stage pipeline in typical input buffered routers is often due to series of steps such as buffer allocation, route computation, switch allocation, switch traversal and link traversal (often requiring a cycle each) needed to route a packet. On the other hand, single-cycle routers discussed in Chapter 2 (subsection 2.1.4) can achieve a pipeline stage of one clock cycle especially for light traffic since the possibility of successful speculation is high. However, the pipeline stage increases when network-traffic is heavy. *R-NoC-SE* distributed architecture provides a much simpler design (no centralized crossbar) and shorter pipeline stages to route a packet. The router is clocked at $500MHz$ and has a small area overhead as displayed. The max operating frequency of the design is reported in Table 6.1.

In order to investigate the impact of lane switching (i.e. switching from primary to secondary lanes) on the router on performance, the router configurations with and without lane switching are compared. Unlike the Rotary [2] router which incurs power and performance overheads caused by allowing packets to make multiple turns in lanes as discussed in Chapter 4, *R-NoC-SE* avoids these overheads by allowing packets to immediately switch to secondary lanes via the switch link.

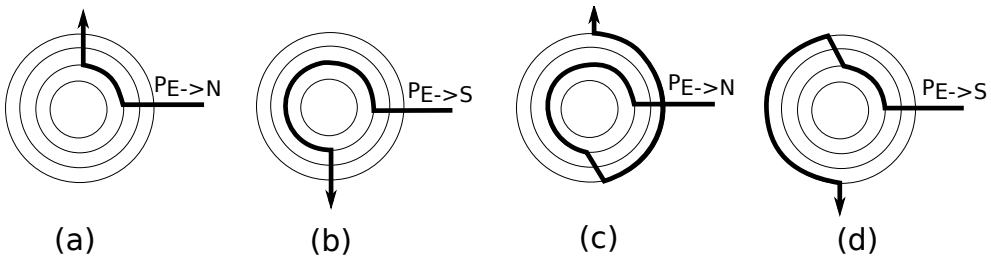


FIGURE 6.3: A packet from east port to north port: (a) without lane switching, (c) with lane switching. A packet from east to south: (b) without lane switching (d) with lane switching.

TABLE 6.2: Impact of lane switching on latency

Without lane switching		With lane switching	
Scenario-A	Scenario-B	Scenario-C	Scenario-D
2 cycles	6 cycles	9 cycles	6 cycles

Fig. 6.3(a) - (d) shows the considered data-flow scenarios, where only primary lanes are used in Fig. 6.3(a) and (b) i.e. no lane switching, while lane switching occurs in Fig. 6.3(c) and (d). The head-flit is considered for the experiment. Table 6.2 shows the corresponding latency incurred for transmitting the head-flit for each

scenario. Here, two important observations are made: (i) taking longer routes in the router does not necessarily incur significant latency overhead. As an example, *scenario-B* incurred only a small additional latency of 4 clock cycles when compared to *scenario-A* where the head-flit took a short route. This observation is also valid for all the other scenarios, (ii) similarly, lane switching does not incur significant packet latency overhead. On the contrary, lane switching avoids additional latency incurred when packets are queued and improves the router throughput since switching lanes frees up the router resource for use by other packets. The network packet latency at low-load is expected to be minimal since only few packet will switch lanes. Thus, lane switching does not significantly impair *R-NoC-SE* performance.

Power estimation

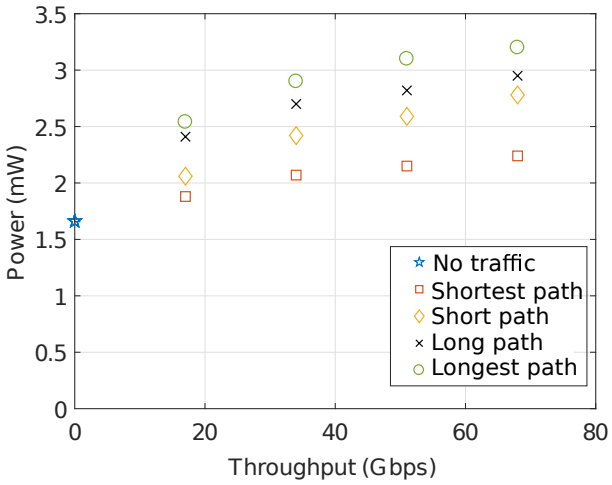


FIGURE 6.4: Power consumption based on paths taken in the router

R-NoC-SE power estimation was carried out at Gate level using *Synopsys Prime-Time PX* and the 45nm CMOS cell library. Fig. 6.4 shows the power results for different packet throughput depending on the paths (i.e. shortest, short, long, longest) taken by packets in the router. *Shortest*, *short*, *long*, *longest* paths correspond to a path from west input to local, south, east and north output ports respectively. The power results are obtained when the router links are operating at their maximum throughput. This plot shows one of the interesting benefits of the resource sharing and adaptive features of *R-NoC-SE*, where the power consumption does not solely depend on the packet throughput but also on the path taken by packets in the router. Hence, less power is consumed for similar packet throughput when shorter paths are utilized in the router, i.e. under light load. Additionally, it is visible from the plot that the power consumption increases proportionally with traffic, which is a desirable feature achieved thanks to the traffic-proportional utilization of lanes and buffering resources.

6.2.2 Baseline synchronous elastic network

For network-level performance exploration, a 4x4-mesh network consisting of *R-NoC-SE* router is considered. A packet length of 10 flits, with flit size of 32 data bits is considered for simulations. In the VHDL testbench, processing blocks attached to the routers local port serve as both producers and consumers of packets. A simulation model where each block keeps sending packets until a stable network reaches a stable latency is considered.

TABLE 6.3: Comparison of Hermes and base *R-NoC-SE*

Router	Net. Sat. (%)	Area (mm^2)	Power (mW)	
Hermes	31	0.05	3.6	
R-NoC-SE (C1)	**20**	**0.03**	Min	Max
			1.8	**2.6**

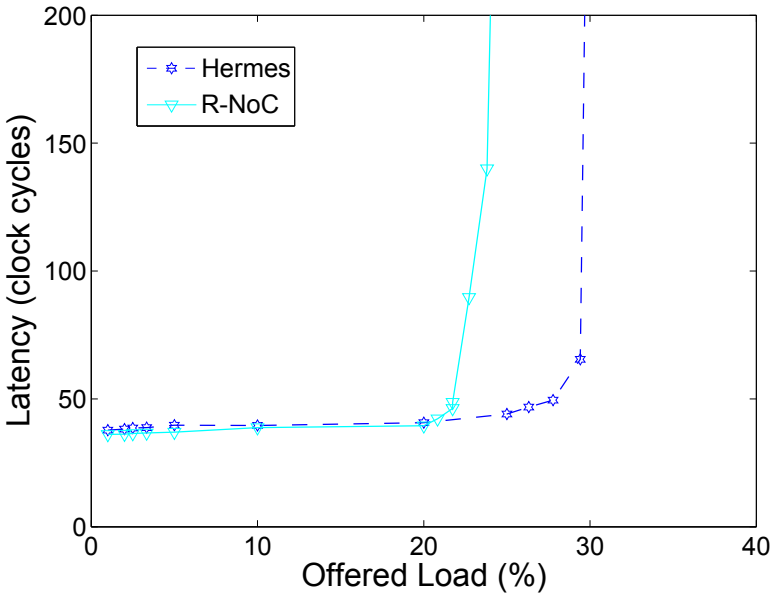


FIGURE 6.5: Performance of base *R-NoC-SE* vs. Hermes for a similar configuration

Table 6.3, shows the evaluation of the base *R-NoC-SE* and Hermes [74]. The performance result (in terms of network saturation throughput) was obtained for uniform traffic pattern explained in Chapter 2, subsection 2.1.5. It is observed that the network performance (4x4 mesh topology) is improved by 55% for a network consisting of Hermes routers compared to a network consisting of base *R-NoC-SE* routers as seen in Fig. 6.5. The main reason for base *R-NoC-SE* poor performance and the solution to significantly improving *R-NoC-SE* performance is presented in the next subsection (i.e. subsection 6.2.3).

6.2.3 Exploring further synchronous elastic router topologies

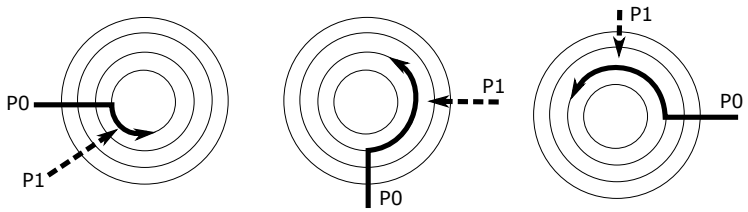


FIGURE 6.6: Blocked packet caused by different arrival times

The evaluation of a four-lane *R-NoC-SE* architecture (i.e. base *R-NoC-SE*) has been presented in Subsection 6.2.2. The resource sharing feature allows packets to switch lanes and use secondary lane resources when contention occurs on the lane. In the topology shown in Fig. 5.5, lane switching frees-up the primary lane resources for use by incoming packets (assuming the secondary lane is free and packets can exit). However, packets cannot always switch lanes when contention occurs on the primary lanes.

Fig 6.6 shows three different scenarios where a packet with a later arrival time (i.e. *P1*) is temporarily blocked because the lane is occupied by an earlier packet (i.e. *P0*). In the scenarios, the switch links are not available for use by the blocked packets. This situation occurs because the parallelism level on the primary lanes is limited. Therefore, the router cannot always support five concurrent data-flows supported in typical router architecture like Hermes [74].

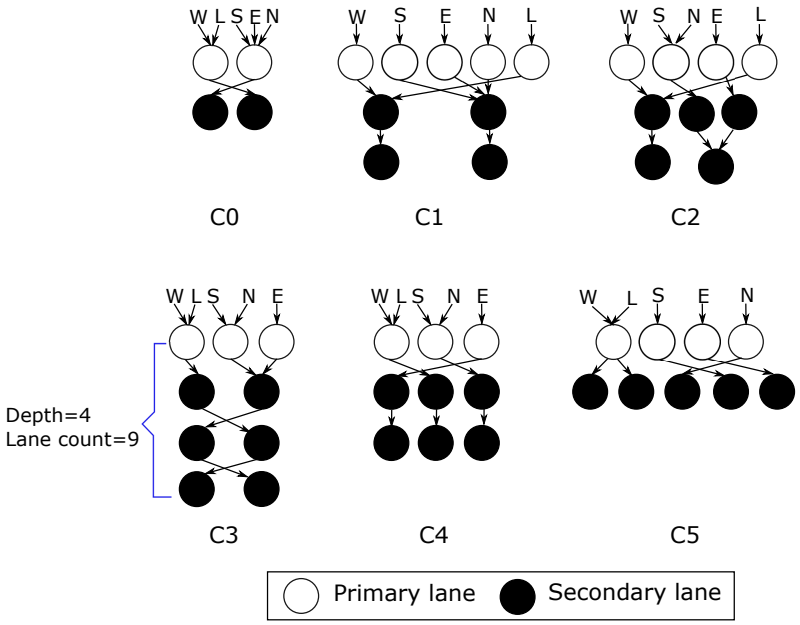


FIGURE 6.7: Considered *R-NoC-SE* configurations

A possible solution to mitigate performance loss caused by temporarily blocked packet is through the use of additional buffers. Additional buffers can be added to

the lanes to improve packet throughput. However, this will incur significant area overhead and latency since packets will have to traverse many buffers assembled on the lanes. *R-NoC* is a highly parametric architecture template that can produce different router configurations i.e. with varying topological trade-offs in terms of number of primary/secondary lanes, input port distribution on the primary lanes and the parallelism level on the lanes, etc. Since *R-NoC* can have *N-number* of lanes, *9 lanes* versions is chosen for exploration because it allows for high level of parallelism on the primary lanes and a good number of shared secondary lanes.

TABLE 6.4: *R-NoC-SE* configurations (P/S denotes the ratio of primary to secondary lanes)

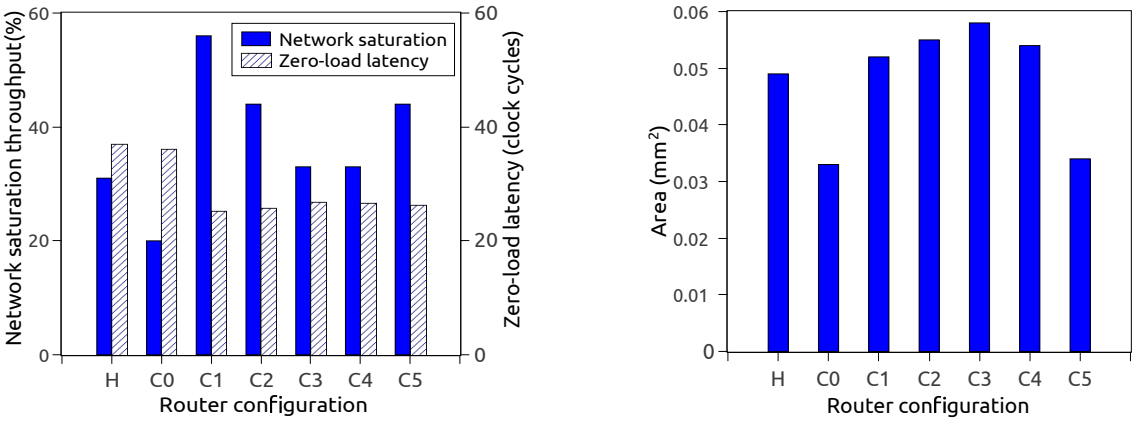
Config.	Lane depth	Parallelism level	P/S (%)	No. of lanes
C0	2	2	50	4
C1	3	5	56	9
C2	3	4	44	
C3	4	3	33	
C4	3	3	33	
C5	2	4	44	

Fig. 6.7 gives the schematics of the considered *R-NoC-SE* router configurations. Configuration *C0* is the 4 lanes version shown in Fig. 6.2 with two primary lanes and two secondary lanes. The west/local input ports are connected to one of the primary lanes and share one secondary lane, while the other input ports (east, north, south) are connected to the other primary lane and share one secondary lane. Table 6.4 shows the properties of the different 9 lanes versions of *R-NoC-SE*. *R-NoC-SE* routers can have a maximum of five primary lanes indicated by the *parallelism level* column in Table 6.4. A level of 5 means the router has maximum parallelism on the primary lanes (i.e. a lane is associated with each input port), while the secondary lanes are shared among packets from input ports. Such a router (e.g., *C1*) configuration in Table 6.4 behaves like a typical input buffered router at low traffic, while the secondary lanes are exploited at medium and high traffic. Another parameter denoted *Depth* relates to the lane connectivity: in a router of *Depth D* a packet can at most be routed on *D* lanes before leaving the router.

Scalability of *R-NoC-SE* routers

Area and performance of *R-NoC-SE* router configurations

Fig.6.8a depicts the performance of different router configurations for uniform traffic pattern. It shows that having a high-level of parallelism on primary lanes is rewarding, since it mitigates performance loss caused by temporarily blocked packets and avoids unnecessary contentions on the lanes. As displayed in Fig.6.8a, the zero-load packet latency is also improved for configurations with higher parallelism level.



(A) Performance of *R-NoC-SE* routers

(B) Area overhead of *R-NoC-SE* routers

FIGURE 6.8: Performance and area-overhead of *R-NoC-SE* routers (H: Hermes)

Router configurations with higher level of parallelism outperform those with less parallelism for similar or even lesser lane depth. Fig. 6.8b shows the corresponding area overhead for the router configurations. It is observed that the performance of the routers does not solely depend on the area, but on their topological parameters.

Scalability of *R-NoC-SE* routers

In order to assess the scalability of *R-NoC-SE*, the same versions with additional buffers are considered and their results are compared to that of their baseline (i.e. versions without additional buffers).

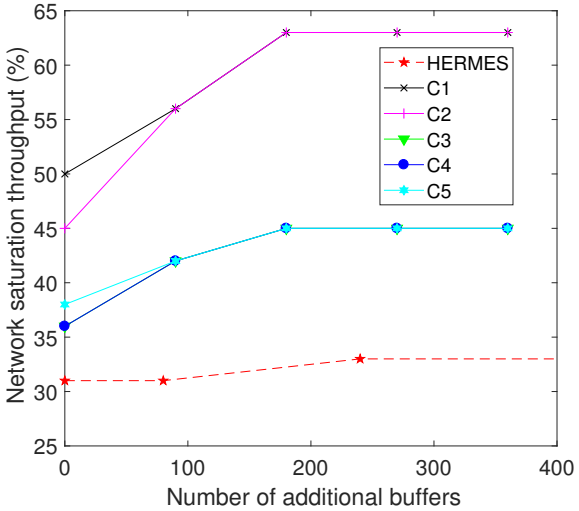


FIGURE 6.9: Comparison w.r.t. different buffer counts. Number represents percentage increase

The additional buffers are evenly distributed on the lanes as shown in Fig. 5.5 where the grey lines represent these buffers. Fig. 6.9 shows the performance of

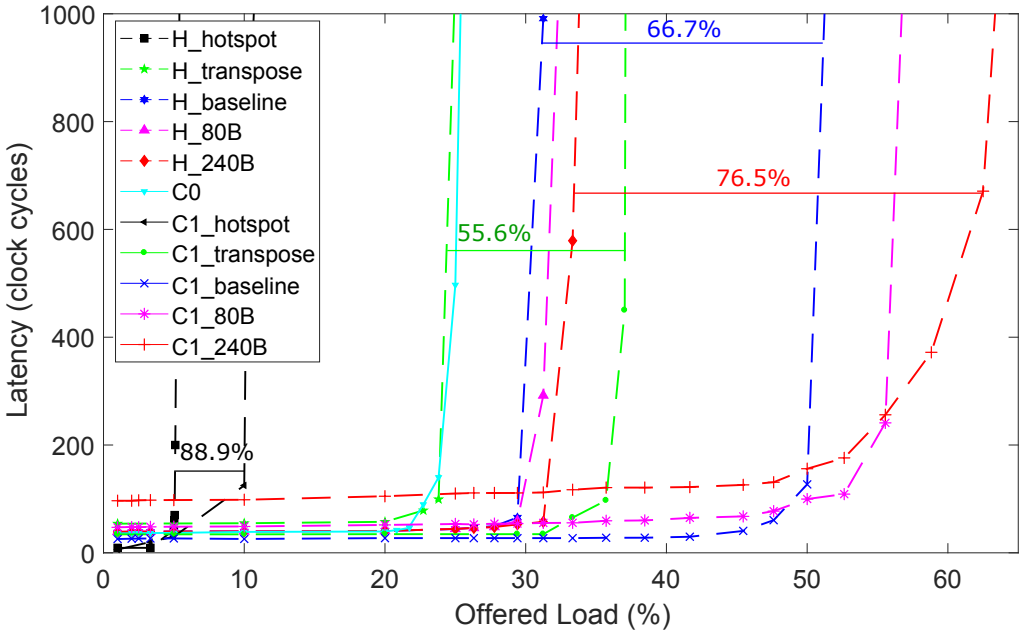


FIGURE 6.10: Comparison w.r.t. different traffics and buffer counts

the routers with additional buffers for uniform traffic pattern. In the plot, *zero-additional-buffers* represent the routers baseline configurations. The network saturation throughput is increased with additional buffers for *R-NoC-SE* routers. The figure also shows the corresponding performance of the Hermes router baseline and with additional buffers. A configuration with a total of 80 buffers for the Hermes baseline configuration, i.e., 16 slot buffer for each input port FIFO queue is considered. Similarly to *R-NoC-SE*, the Hermes router uses *wormhole* flow-control and *XY-routing*. It employs *credit-based* buffer management scheme. The baseline configurations for *C1, C2, C3, C4, C5* have a total of 80, 94, 68, 102, 86, and 88 buffers respectively. It is observed that *R-NoC-SE* offers better scalability than Hermes router [34].

Comparison of *R-NoC-SE (C1)* and Hermes

The *R-NoC-SE* configuration *C1* is selected for comparing with Hermes [74]. Both routers have a buffer count of 80 for a fair comparison.

TABLE 6.5: Comparison of Hermes and *R-NoC-SE*

Router	Area (mm^2)	Power (mW)	
Hermes	0.049	3.6	
R-NoC-SE (C1)	**0.052**	Min	Max
		2.6	**2.9**

Table 6.5 shows the area results for both routers. The Hermes router has an area-overhead that is *5.7%* smaller when compared to the *R-NoC-SE* router. *R-NoC-SE*

additional area is due to its multilanes/distributed design, requiring several mux and arbiters to control access to shared resources in the router. *R-NoC-SE* in turn consumes less power when compared to Hermes due to its input buffers [74]. The power results for both the shortest (i.e. min power) and the longest (i.e. max power) path on a given lane in *R-NoC-SE* is displayed. The power results were obtained for a single link operation when the routers were operated at similar frequency.

Fig. 6.10 shows the performance of both routers. The performance of the *4 lanes R-NoC-SE* is also displayed. As shown in the plot, the Hermes router outperforms the *R-NoC-SE C0* router, offering better network saturation throughput. As explained earlier, the Hermes router can always support up to five concurrent dataflows (for different source-destination pairs) regardless of the packet arrival time. Conversely, packets for different *source-destination* pair may compete for channel resources in the *R-NoC-SE C0* router. This observation motivated the *9 lanes versions of R-NoC-SE*. The 9 lanes *R-NoC-SE* (i.e. *C1_baseline*) provides a performance improvement of over 60% compared to the Hermes baseline router (i.e. *H_baseline*). In Fig. 6.10, *R-NoC-SE (C1)* with additional buffers significantly outperform the Hermes routers (with additional buffers) by up 77% [34].

The routers (Hermes and *R-NoC-SE C1*) were also simulated using *transpose and hotspot* traffic patterns. In transpose traffic pattern, each node communicates only with destination node with the upper and lower halves of its own address. In hotspot traffic pattern, all nodes communicate with a specific node i.e. *the hotspot* node. This creates a higher network contention when compared to the transpose and uniform traffic as stated in Chapter 2. It is observed that the network saturation throughput for *R-NoC-SE* is improved by 56% and 88% for transpose and hotspot traffic respectively when compared to the Hermes router. This confirms the intrinsic ability of *R-NoC-SE* to support specific traffic patterns by means of dynamically allocating buffer resources whenever needed.

Comparison of *R-NoC-SE* and Rotary

The Rotary [2] shares similar traffic-roundabout concept with the proposed router. The quantitative and qualitative comparison of both routers is presented here.

TABLE 6.6: Comparison of *R-NoC-SE (C1)* and Rotary [2]

Router	Head-flit-lat (cycles)	Saturation (%)	Flow-control	Network Topology
Rotary [2]	4	115	VCT/bubble	2D-Torus
C1	**2**	**50**	**Wormhole**	**Mesh**

Table 6.6 shows the comparison of Rotary [2] and *R-NoC-SE (C1)*. The head-flit travel time for *R-NoC-SE* is twice less than that of Rotary since it uses shorter router

pipeline requiring only two cycles. Table 6.6 also displays the network saturation for both routers for similar network size. The network saturation threshold for Rotary outperforms that of *R-NoC-SE*. However, this performance improvement comes at a significantly higher area and power cost associated with the use of large buffers. Besides, Rotary uses Torus network which provides shorter paths between network nodes. Packets in the Rotary router are allowed to make multiple turns before using any available output port. This leads to significant dynamic power consumption especially at high traffic injection rates, caused by an increase in router switching activities. Conversely, packets in *R-NoC-SE* can adaptively switch lanes to free up the router resource for use by other packets.

Comparison of *R-NoC-SE* and VC-based single cycle routers

TABLE 6.7: Network saturation throughput (%) of virtual-channel (VC) based routers and *R-NoC-SE (C1)* for uniform traffic pattern.

Packet length	RIVR [24]	IVR-SC [76]	FOVR-LS [22]	***C1***
4	70%	71%	61%	**57%**
8	54%	56%	53%	**53%**
12	48%	49%	51%	**50%**
16	45%	46%	47%	**47%**

Table 6.7 shows the performance comparison of *R-NoC-SE* with state of the art virtual channel (VC) based routers for similar mesh network topology and buffer count. The *BIVR* [24] router represents typical VC routers with 5 pipeline stages, while the *IVR-SC* [76] and *FOVR-LS* [22] are *single cycle routers* requiring only *one* clock cycle for a single flit to travel-through the router. In general, the network no-load latency for *FOVR-LS* is lower than that of *R-NoC-SE* for equal packet length, while the network no-load latency for *IVR-SC* is only marginally lower than that of *R-NoC-SE*. As shown in Table 6.7, the network-saturation threshold for *R-NoC-SE* is highly competitive to that of the VC routers. The *R-NoC-SE* is expected to provide improved performance for non-uniform traffic due to its shared-buffer and dynamic resource allocation features.

6.3 Asynchronous evaluation

This section evaluates the asynchronous implementation of *R-NoC* router named *R-NoC-A* and corresponding network. This section also explores further *R-NoC-A* topologies in terms of area and performance besides the base *R-NoC-A* router configuration. *R-NoC-A* is also compared to existing asynchronous routers. As stated in Chapter 5, *R-NoC-A* is implemented using delay-insensitive 4-phase dual-rail protocol.

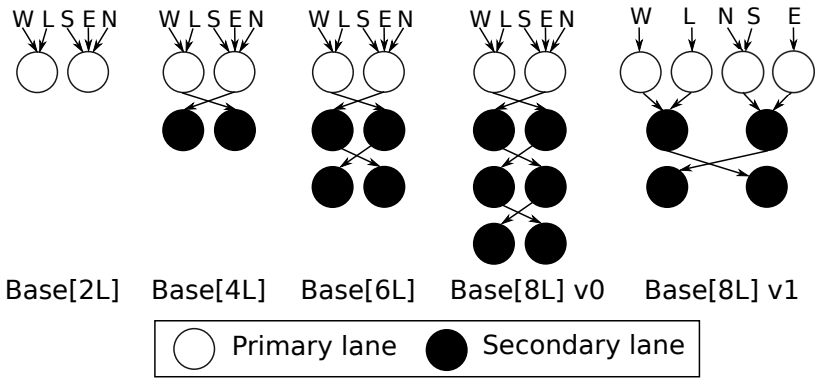


FIGURE 6.11: Considered *R-NoC-A* configurations

Fig. 6.11 shows the *R-NoC-A* configurations that are considered for exploration. Note that the *Base[4L]* configuration is similar to *R-NoC (C0)* explained earlier in this chapter.

6.3.1 Asynchronous router

Performance and Area Evaluation

TABLE 6.8: Performance of 34-bits ***R-NoC-A routers***. Base [xL] + yB denotes x lanes router with y additional buffers.

Router Version	Head flit latency ($nsec$)		Throughput ($Mflit/sec$)	Total area (mm^2)
	Min	Max		
Base [2L]	0.7	2.1	465	0.14
Base [4L]				0.18
Base [4L] + 20B	0.9	2.6		0.21
Base [4L] + 40B	1.0	2.9		0.25
Base [4L] + 80B	1.1	3.6		0.33

Table 6.8 shows the router versions considered for area and head-flit latency evaluations. The 2-lanes version i.e. *Base [2L]* consists of only the first 2-lanes as displayed in Fig. 6.11. The *Base [4L] + 20B*, *Base [4L] + 40B* and *Base [4L] + 80B* versions represent 4-lanes *R-NoC-A* routers with additional buffers. The additional buffers were uniformly distributed on the lanes as shown in Fig. 5.5, where the gray lines represent the additional buffers. The 4-lanes version is chosen to facilitate comparison with *R-NoC-SE* (i.e. the base synchronous elastic counterpart that also uses 4 lanes). For performance evaluation of the routers, gate-level simulations of the router blocks were back-annotated with delay values from the synthesis tool as stated earlier. It is observed that the area overhead grows with the number of buffers in the routers. At low traffic, latencies are lower because packets "cut-through" from

primary lanes to output ports, whereas they are increased at high traffic because secondary lanes are taken. The min and max latencies are displayed in Table 6.8. The router achieves a throughput of 465 Mflit/sec for the 45nm CMOS technology.

6.3.2 Baseline asynchronous network

For network-level performance exploration, a 4x4-mesh network of asynchronous *R-NoC-A* routers is considered. Similar to the synchronous evaluation, a packet length of 10 flits, with flit size of 34 bits is considered for simulations. The simulation model is similar to the one described in Section 6.2. Fig. 6.12 shows the network latency and offered load plot for the various *R-NoC-A* versions. It is observed that, the *Base [4L]* router achieves a better network saturation threshold ($49Gbps$) than the *Base [2L]* version ($27Gbps$) since it has more buffering resources and packets can switch lanes to avoid contentions [35].

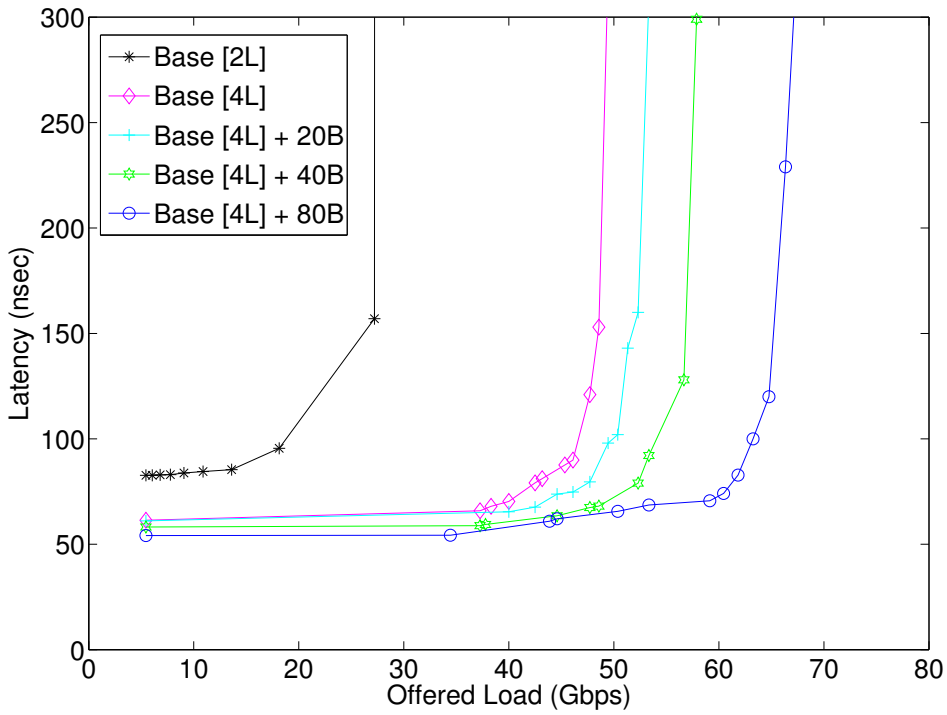


FIGURE 6.12: Performance of two and four lanes *R-NoC-A* NoCs. Base [xL] + yB denotes *R-NoC-A* version with x number of lanes + y additional buffers)

To further explore the proposed router performance, networks consisting of the three router versions with additional buffers shown in Table 6.8 i.e., *Base [4L] + 20B*, *Base [4L] + 40B* and *Base [4L] + 80B* is compared to a network consisting of the *Base [4L]* router with only 28 buffers in terms of performance. The uniform traffic pattern was considered for simulations. The performance of the router versions is shown

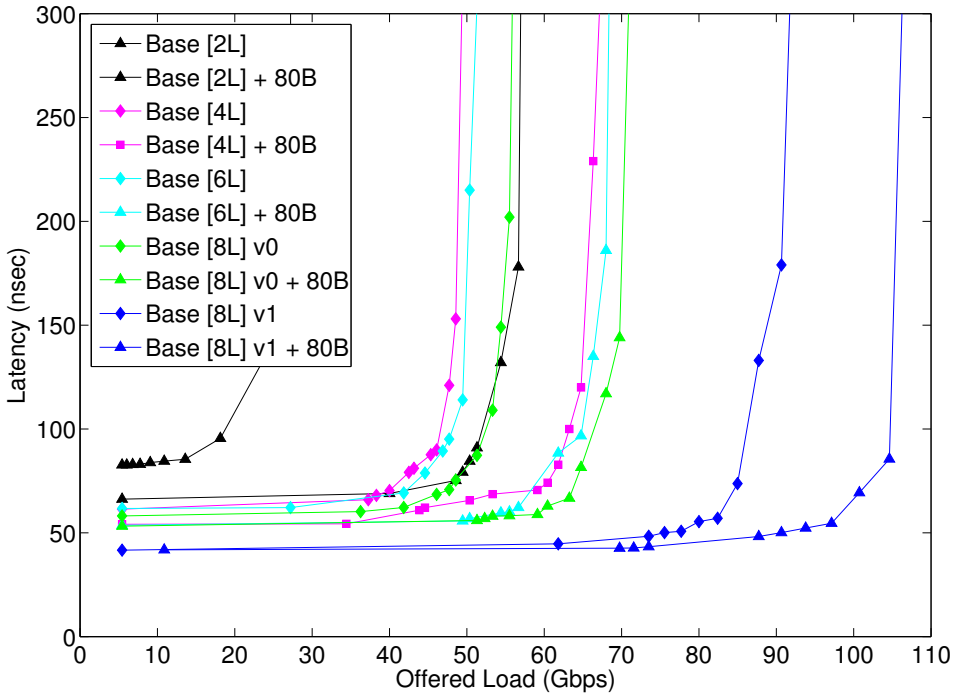


FIGURE 6.13: Performance exploration of *R-NoC-A* asynchronous NoCs

in Fig. 6.12. It is observed that the network saturation threshold is significantly improved for *Base [4L] + 80B* when compared the other versions since it has more buffering resources. This shows that adding *more buffers* is rewarding. A similar observation can be made for 6-lanes *R-NoC-A* (schematic is given in Fig. 6.11) with additional buffers shown in Fig. 6.13, which not only provide higher bandwidth but also benefits from additional buffers.

6.3.3 Exploring further asynchronous router topologies

As shown in Subsection 6.3.2, network performance can be significantly improved through the use of additional buffers. However, additional buffers can introduce additional area overhead. The goal of the subsection is to show that the asynchronous network performance can be significantly improved by exploring different *R-NoC-A* topologies and without corresponding area overhead. In order to realize this, two 8-lanes *R-NoC-A* routers i.e. *Base [8L] v0* and *Base [8L] v1* with a similar number of buffers are devised. The schematics are given in Fig. 6.11. *Base [8L] v0* has 2 *primary* lanes and 6 *secondary* lanes. *Base [8L] v1* has 4 *primary* lanes and 4 *secondary* lanes. Fig. 6.13 shows the performance of the two versions. It is observed that the network saturation threshold is significantly increased by up to 70% for *Base [8L] v1* when compared to *Base [8L] v0*. The reason is that *Base [8L] v1* allows for more concurrent

data-flows on its *primary* lanes when compared to *Base [8L] v0*. Increased concurrency on primary lanes results in fewer contentions, which improves throughput and the overall network performance. Note that the overall lower average latency at low traffic finds root in the overall *shorter path* taken in the routers.

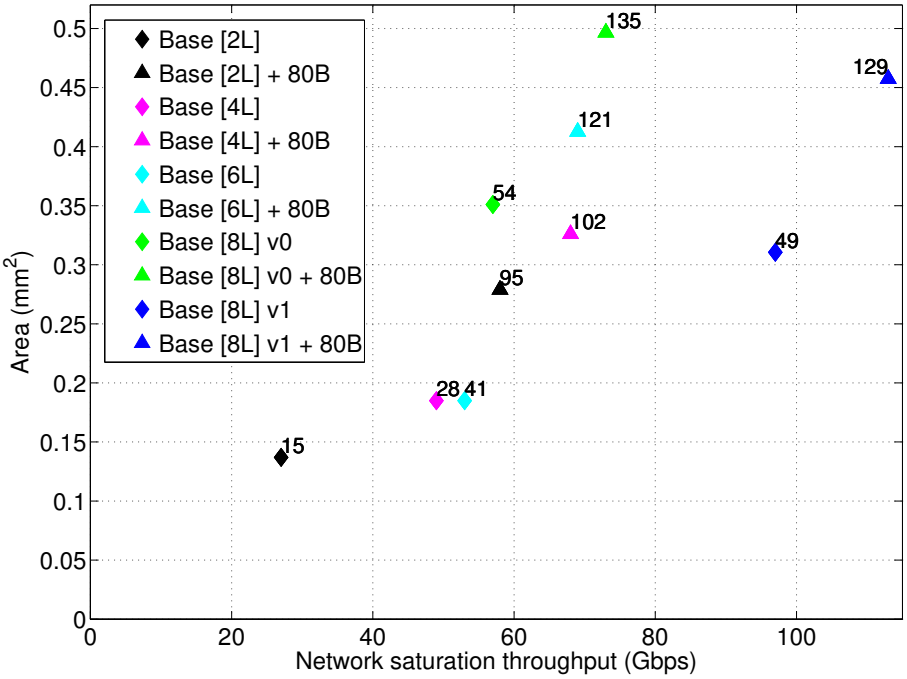


FIGURE 6.14: *R-NoC-A* performance/area trade-offs. Marker number: the no. of buffers available in the router version. Here, router versions with the same topology but different number of buffers same **colours** is compared.

Lastly, *R-NoC-A* performance/area trade-offs is explored. For this purpose, Fig. 6.14 shows the area against network saturation threshold for various router versions. In Fig. 6.14, similar *colours* represent routers with the same topology but differ in terms of number of buffers. It is observed that the network saturation threshold is significantly improved with additional buffers (*considering router versions with the same topology*). The *Base [8L] v1* version offers good performance and area trade-off. This plot further suggests the performance scalability of *R-NoC-A* in which both *buffers and lanes* can be tuned for matching the desired level of performance.

6.3.4 Comparison with existing solutions

Table 6.9 shows the performance and area comparison of *R-NoC-A* against existing routers [36]. The *R-NoC-A* results for both 45nm and 180nm CMOS technology is presented in order to facilitate our comparison with existing works based on old and more recent CMOS technology nodes. The *2 lanes* version of *R-NoC-A* with *10-bits*

TABLE 6.9: Performance and area comparison of *R-NoC-A* with existing solutions.

Router	Area (mm^2)	Head-flit latency ($nsec$)	Throughput ($Mflit/sec$)	Flit size ($bits$)	Tech (nm)	Router design
Rotary [2]	NA	10	400	64	NA	Synchronous
Bubble [87]	18.7	22.6	177	64	700	Synchronous
QNoC [30]	1.35	NA	NA	8	350	4-phase bundle data
	NA	10	208	NA	180	
HERMES-A [85]	0.33	NA	90	10	180	4-phase dual-rail
MANGO [11]	0.28	8.2	646	33	120	4-phase bundle data
ONIZAWA [7]	NA	2.7	526	34	130	2-phase dual-rail
ANoC [7]	0.25	2.7	250	34	130	4-phase dual-rail
FAUST [5]	NA	6	160	NA	130	
MAGARI [99]	0.17	2.3	550	32	65	
R-NoC-A[2-lanes]	**0.12**	**2.6**	**406**	**10**	**180**	
R-NoC-A[4-lanes]	**0.36**					
R-NoC-A[2-lanes]	**0.36**			**34**		
R-NoC-A[4-lanes]	**0.71**					
R-NoC-A[2-lanes]	**0.05**	**1.8**	**465**	**10**	**45**	
R-NoC-A[4-lanes]	**0.07**					
R-NoC-A[2-lanes]	**0.14**			**34**		
R-NoC-A[4-lanes]	**0.18**					

flit width achieves a smaller area overhead when compared to HERMES-A [85] for similar technology, asynchronous protocol and flit. *R-NoC-A* uses compact control circuits with minimal area overhead and does not use virtual-channels.

R-NoC-A based on the 45nm CMOS technology achieves one of the fastest head flit transmission time of 1.8ns (Here, the average for all possible path combinations is considered). *R-NoC-A* head-flit transmission time is significantly faster than most existing solutions as can be seen in Table 6.9. In terms of throughput, MANGO [11], ONIZAWA [7] and MAGARI [99] provide a higher throughput when compared to *R-NoC-A*. The MANGO [11] can be simply design since it is base on the bundle data protocol. However, this protocol is not delay-insensitive. ONIZAWA [7] is based on *2-phase delay-insensitive (DI)* protocol, which is generally faster than the *4-phase DI* protocol counterpart. This is because the *2-phase protocol* reduces the communication step by half compared to the 4-phase protocol. However, implementing circuits using the 2-phase protocol is complex [99]. The implementation in [7] support only a certain packet structure.

Compared to the synchronous routers, *R-NoC-A* achieves a higher throughput than the Rotary [2] and the Bubble [87] routers. Here, a 400MHz clock operating frequency is assumed for the Rotary router. For the Bubble router, the crossbar has the longest combinational path due to its arbitration complexity [87]. It imposes a cycle time of 5.65ns corresponding to a 177$Mflit/sec$ throughput. In summary, *R-NoC-A* performance and area results are competitive with existing synchronous and asynchronous solutions, thereby offering beneficial trade-offs.

6.4 Discussion: synchronous elastic vs. asynchronous

This section provides a quantitative comparison of the synchronous elastic (*R-NoC*) and asynchronous (*R-NoC-A*) versions of the router

TABLE 6.10: Comparison of *R-NoC-SE* and *R-NoC-A* routers using same 4-lanes *R-NoC* topology displayed in Fig. 6.2

Router	Area (mm^2)	Head-flit Lat. (*nsec*)	Power (mW)
Synchronous (*R-NoC-SE*)	0.03	6.6	1.88
Asynchronous (*R-NoC-A*)	0.18	0.7	0.89

Table 6.10 shows the comparison of both routers. It is observed that the synchronous routers leads to significant area overhead reduction compared to the asynchronous router. As explained in Chapter 5, delay-insensitive (DI) often lead to significant area overhead due to the its completion detector circuit. On the other hand, the asynchronous logic provides faster circuit compared to synchronous logic as explained in Chapter 5, since it is only limited by propagation delay of individual components instead of the worst case delay of a global clock [96]. As displayed in Table 6.10, *R-NoC-A* provides faster data transfer compared to the *R-NoC*, which is limited by global clock. Therefore, asynchronous logic provides a good performance/area trade-off.

One benefit of asynchronous logic as discussed in Chapter 5 is that it provides almost zero standby or static power consumption [96] which contributes to decreasing the total power consumption. Table 6.10 displays the total power consumed for a single link operation (i.e. packets from an input port to an output port in the router). The asynchronous circuit in this case provides lower dynamic power consumption.

Chapter 7

Evaluation of further network topologies

«"Success is walking from failure to failure with no loss of enthusiasm."»

Winston Churchill

Contents

7.1 Introduction

As stated in Chapter 4, the *R-NoC* concept provides a highly-adaptable architecture, which allows the router to be configured to meet numerous network topologies and applications demands. This chapter presents the evaluation of *R-NoC-DM* (i.e. router for the diagonally-linked mesh network topology described in Chapter 4). The *R-NoC-DM* evaluation flow is similar to that of *R-NoC-SE* discussed in Chapter 6.

The reminder of this chapter is as follows: section 7.2 presents the considered *R-NoC-DM* topologies. Section 7.3 evaluates the configurations, while section 7.3.2 provides comparison with existing solutions. Section 7.4 provides a summary of the chapter.

7.2 *R-NoC-DM* configurations

As stated in Chapter 4, *R-NoC-DM* has 4 additional ports (i.e. the diagonals) when compared to base *R-NoC-SE* (for mesh network topology). The diagonal links provide shorter network paths and reduce network congestions [50], which leads to enhanced network performance. On the other hand, routers for the diagonally-linked mesh require 4 additional ports which introduce additional arbitration complexity and area/power overheads. The goal of this section is to exploit *R-NoC* highly-adaptable architecture to realize significantly higher performance for diagonally-linked mesh network without corresponding area/power cost compared to typical input buffered router. In order to investigate performance/area trade-off, several *R-NoC-DM* router configurations are devised.

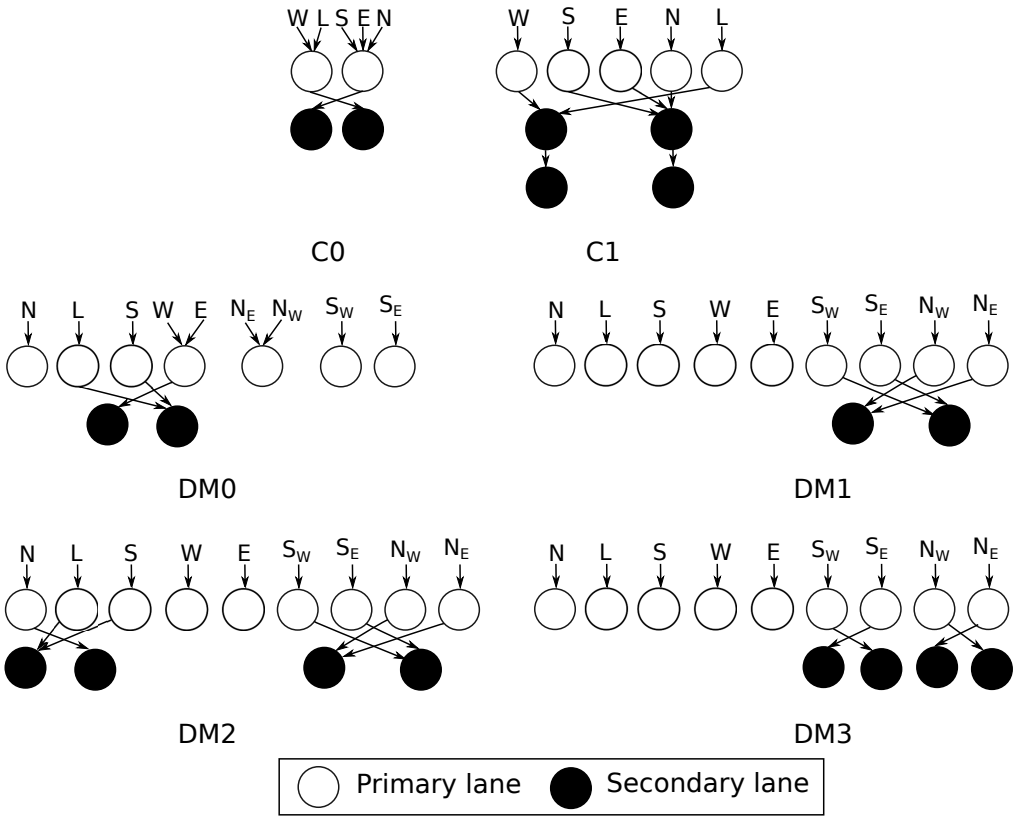


FIGURE 7.1: Considered *R-NoC-DM* configurations

Fig. 7.1 gives the schematics of the considered *R-NoC* router configurations. Note that Configurations *C0 and C1* are *R-NoC-SE* configurations for mesh network topology. They have been discussed earlier in Chapter 6. Table 7.1 shows the properties of the considered *R-NoC-DM* router configures.

TABLE 7.1: *R-NoC* configurations. DLM: diagonally-linked mesh

Config.	No. of ports	Parallelism level	No. of lanes
C0	mesh (5)	2	4
C1		5	9
DM0	DLM (9)	5	9
DM1		9	11
DM2		9	13
DM3		**9**	**13**

Router configurations with an equal number of ports and parallelism level have maximum parallelism on the primary lanes (i.e. a lane is associated with each input port), while the secondary lanes are shared among packets from input ports. Such routers behave like a typical input buffered router at low traffic, while the secondary lanes are exploited at medium/high traffic. The main difference between *DM2* and *DM3* is in their primary lanes. In *DM3*, a primary lane (with diagonal input port) is reserved only diagonal output port, while non-diagonal output ports are also attached to primary lanes (with diagonal input ports) in *DM2*. Router configuration

DM3 is expected to provide better performance since the additional delay for packets utilizing the diagonal output ports is removed.

7.3 Evaluation of *R-NoC-DM* configurations

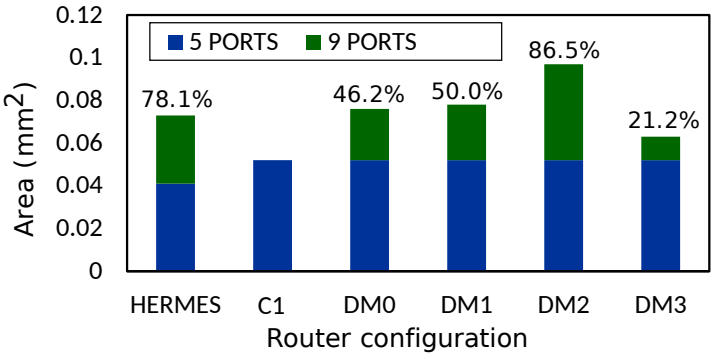


FIGURE 7.2: Router area. (Numbers: overheads introduced by additional ports)

Fig. 7.2 shows the corresponding area overhead for the routers. The cost incurred when additional ports are added to the router configuration is displayed. It is observed that adding additional ports to Hermes incurs massive cost compared to most *R-NoC-DM* router configurations.

7.3.1 Exploring *R-NoC-DM* NoCs

Performance of *R-NoC-DM*

Fig. 7.3 depicts the performance of different router configurations for uniform and transpose traffic patterns.

It shows that the diagonal links significantly leads to improved network performance for the considered traffic patterns since they provide shorter paths between network nodes and more communication links. As displayed in Fig. 7.3, the zero-load packet latency is also improved for the diagonal links. As shown in Fig. 7.3, assigning only the diagonal input port and corresponding output port on a lane further improves network performance. This is the case with router config *DM3* that further achieves excellent performance for transpose traffics (similar to that of uniform), outperforming all other configurations. In addition, it is observed that the performance of the routers does not solely depend on the area, but on their topological parameters.

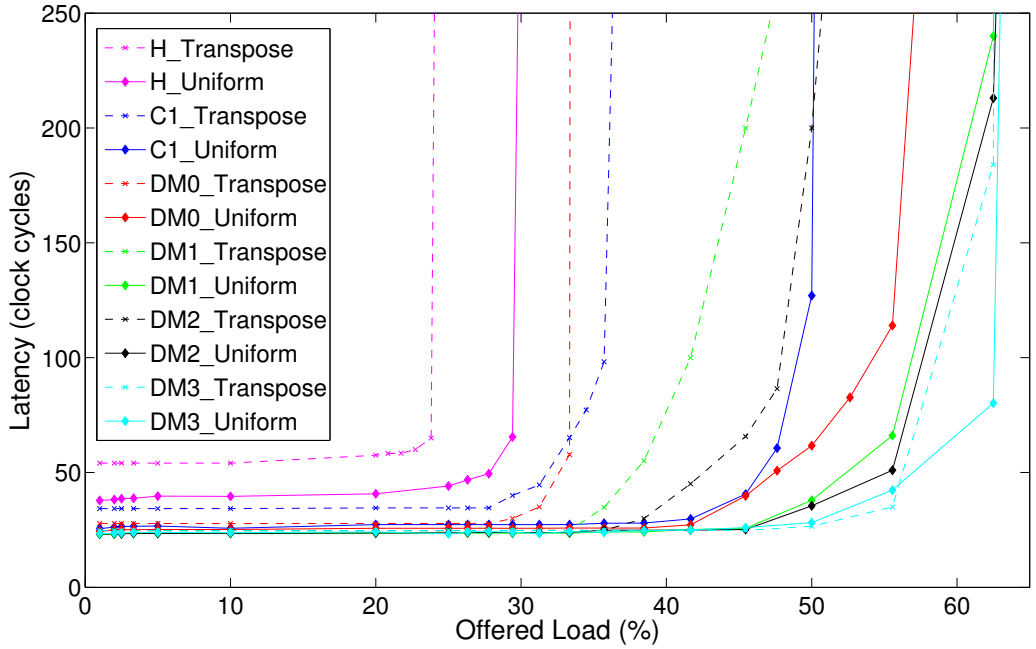


FIGURE 7.3: Performance of *R-NoC* routers. (H: Hermes)

Performance scaling through buffer insertion

In order to assess the scalability of *R-NoC-DM*, the router versions with additional buffers are compared against their baseline (i.e. without additional buffers). The additional buffers are evenly distributed on the lanes as shown in Fig. 5.5 where the grey lines represent these buffers.

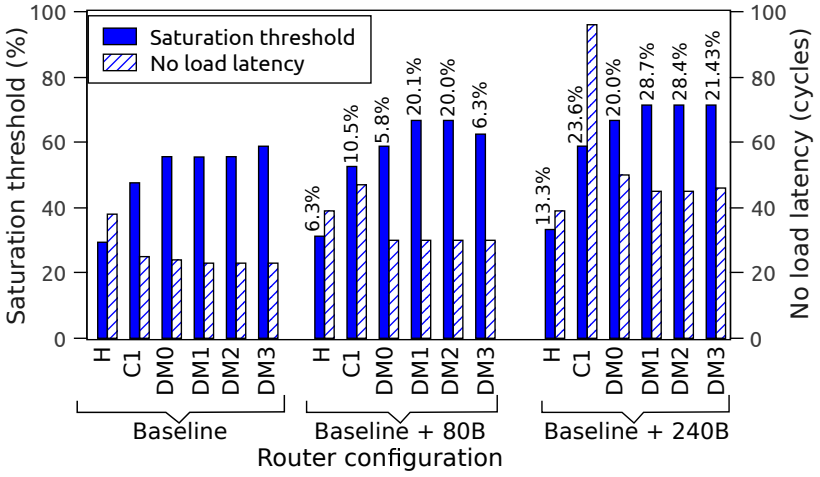


FIGURE 7.4: Performance comparison for *R-NoC-DM* configurations and Hermes (H) routers for uniform traffic.

Fig. 7.4 shows the performance of the routers with additional buffers for uniform traffic pattern, while Fig. 7.5 shows the performance for transpose traffic pattern.

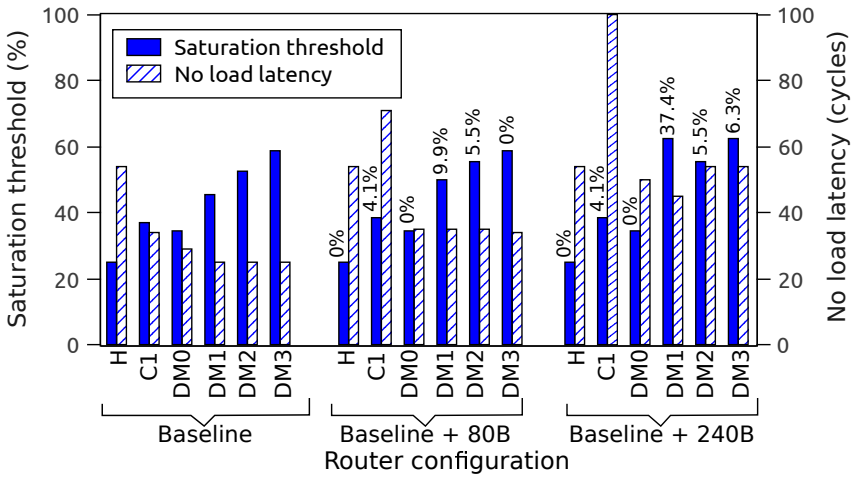
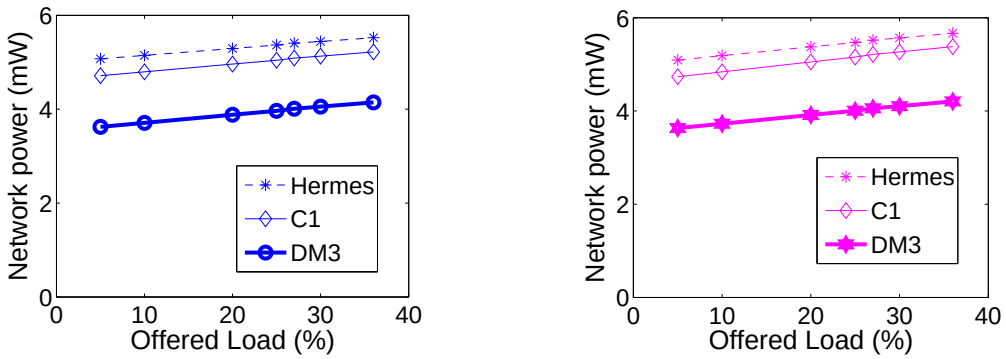


FIGURE 7.5: Performance comparison for *R-NoC-DM* configurations and Hermes (H) routers for transpose traffic.

The network saturation throughput is increased with additional buffers for *R-NoC-DM* routers. As displayed in Fig. 6.8, the router configurations with diagonal links provide better scalability. The figures also shows the corresponding performance of the Hermes router baseline and with additional buffers.

A configuration with a total of 80 buffers was considered for the Hermes baseline configuration, i.e., 16 slot buffer for each input port FIFO queue for a fair comparison with *C1* which also has a total of 80 buffers. The Hermes router uses *wormhole* flow-control and *XY-routing*. It employs *credit-based* buffer management scheme for buffer allocation. As discussed earlier in Chapter 6, *C1* offers an overall improved performance over Hermes for uniform traffic and provides better scalability than Hermes.

Network-level power consumption



(A) Uniform traffic (B) Transpose traffic

FIGURE 7.6: Network-level power consumption comparison

Fig. 7.6 shows the network-level power comparison of *R-NoC-SE, R-NoC-DM* and Hermes. Power estimation was carried out at Gate level using *Synopsys Prime-Time PX* and the 45nm CMOS cell library as discussed in Chapter 6. A clock frequency of 300MHz was considered for simulation. A network size of 2x2 (for simplicity reason) was considered for uniform and transpose traffic. It is visible from the plot that the performance improvements of both *R-NoC-SE and R-NoC-DM* routers over Hermes are not achieved at the expense of higher power consumption. Both *R-NoC-SE and R-NoC-DM* achieve lesser power consumption than Hermes since they can transmit data faster than Hermes, which leads to reduced network switching activities in the routers. Note that *R-NoC-DM* provides overall reduced power consumption because of the shorter network diameter and therefore lesser number of hops between routers.

Application performance

TABLE 7.2: Real application characteristics

Applications	Number of tasks
Video object plan encoder (VOPD) [100]	16
Multimedia system (MMS) [81]	25
WiFi application (802.11) [103]	24
Multi-window display (MWD) [10]	12

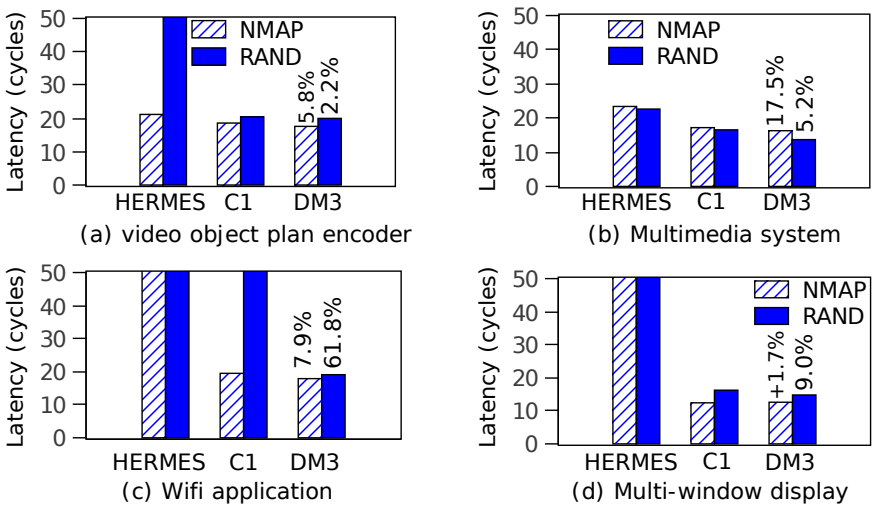


FIGURE 7.7: Routers performance for application traffic. Number on bars: % decrease in average package latency. (+ve value shows % increase)

The network performance for real-world applications traffics was estimated. Table 7.2 gives the properties of the considered applications. The applications are represented as tasks graphs, where the inter-tasks communications require specific communication bandwidth for meeting application requirements. The method proposed in [101] was followed in order to generate the experimental traffic for the

applications. The random (RAND) and near (NMAP) mappings were considered and mapped on a 4x4 network. In NMAP, communicating tasks are mapped close to each other in order to reduce average packet latency. Multiple tasks are mapped on a single core for MMS and 802.11 application since they have more tasks than network cores. Fig. 7.7 shows the result of the application mappings in which the average communication latency is reported. Generally, NMAP intuitively provides lower latency than RMAP. *C1* provides improved performance over HERMES due to its resource sharing feature. Compared to *C1* and Hermes, *DM3* provides even lower latency for both NMAP and RMAP for most applications due to the diagonal links that create shorter distances between communicating nodes, as well as increased total NoC bandwidth.

7.3.2 Comparison with existing solutions

The *R-NoC-DM* configuration *DM3* is selected for comparing with Hermes [74].

TABLE 7.3: Comparison of Hermes and *R-NoC-DM* router

Router	Area (mm^2)	Power (mW)	
Hermes (9 ports)	0.073	3.6	
R-NoC-DM (DM3)	**0.063**	Min	Max
		3.3	**4.0**

Table 7.3 shows the area results for both routers. It is observed that the area overhead *R-NoC-DM* is 13.7% less than that of Hermes for a similar number of ports. Hence, *R-NoC* can provide significant performance improvement at a feasible cost due to its highly adaptable architecture. Power estimation of the routers at Gate level was carried out using *Synopsys PrimeTime PX* and the 45nm CMOS cell library. The power results for both the shortest (i.e. min power) and the longest (i.e. max power) path on a given lane in *R-NoC-DM* is displayed in Table 7.3. The power results were obtained for a single link operation when the routers were operated at a similar frequency. *R-NoC-DM* provides power reduced of 9% over Hermes for the short path.

TABLE 7.4: Comparison of *R-NoC-DM* and DMesh router [50] in terms of cost of adapting to DMesh network topology

Router	Power overhead	Area overhead
DMesh Router [50]	65%	N/A
DM3	**37.9%**	**21.2%**

Authors [50] proposed a DMesh router and compared it to that of their mesh router known as NEPA [50]. Table 7.4 shows the cost of adapting to DMesh network topology in terms of power and area. As displayed in Table 7.4, the DMesh router incurs a significant cost in terms of power consumption compared to *R-NoC-DM*

since it uses crossbars and input buffers. Thus, adapting to DMesh network topology requires larger crossbars which consumes additional power. The result further supports the claim that *R-NoC* architecture can be readily adapted for significant performance improvement without corresponding power overhead.

7.4 Summary

This chapter extended *R-NoC* to provide support for diagonally linked mesh network topology (DMesh)[50] for performance improvement. As stated in Chapter 4, routers for the DMesh network have 4 additional ports (i.e. the diagonals) compared to routers for the mesh network topology. The additional ports introduces additional router area and power overheads. As shown in the evaluation, extending *R-NoC* to support DMesh incurs a small area and power overhead compared to typical input buffered router. As explained in Chapter 2, *R-NoC* do not use explicit crossbars due to its distributed architecture. Therefore, introducing additional ports does not incur a quadratic cost as is the case with typical input buffered routers that are based on crossbars. This provides opportunities for extending *R-NoC* to support concentration networks (discussed in Chapter 2), where manycores shares a single router, at a minimum cost and other network topologies such as three-dimensional NoCs.

Chapter 8

Conclusion

The demand for more power-efficient and higher performance computing systems has ushered in the manycore era, where many intellectual property cores (IP cores) can be integrated on a single chip. This new trend has provided a higher level of performance for meeting various applications requirements. However, as the number of cores grows, there is need for scalable on-chip interconnect networks that can deliver high speed data transfer among the many IP cores. Studies show that traditional bus and crossbar interconnects do not scale with increasing number of cores. Conversely, Networks-on-Chip (NoCs) has emerged as an alternative and scalable interconnect for manycore systems. However, most existing NoCs suffer performance degradation due to underutilization of NoC resources.

In order to remove the limitation of existing approach, this thesis was set out with the goal of designing a scalable, high performance and power efficient NoC for future manycore systems.

A survey of state-of-the-art NoC revealed that *roundabout-inspired/ring-like* NoC router architectures are attractive as they provide inherent resource utilization for improving performance. However, such architectures are susceptible to deadlock due to their ring-like architectures. In other to avoid deadlock, the Rotary [2], a roundabout-inspired router, utilized combined virtual-cut-through (VCT) and bubble flow-control known as local bubble scheme (LBS). While LBS can avoid deadlock, it introduces higher area/power constraints. Therefore, it is not suitable for systems with tight area/power overheads. In order to remove the limitations of existing solutions, this thesis first proposes an algorithm capable of generating deadlock-free roundabout-inspired router architectures that use wormhole flow-control. Thus, drastically reducing the area/power overhead associated with LBS. The algorithm was used to generate *R-NoC* router for various network topologies. The router concept consists of lanes shared by multiple input/output ports in order to improve resource utilization. *R-NoC* provides dynamic resource allocation where a subset of the total number of lanes is used at low traffic, while the others are exploited with network contention increases.

The initial evaluation of the base *R-NoC* router, which consists of 4-lanes shared by multiple inputs/output ports, shows that the router achieves a network zero-load latency that is 32% less than that of Hermes [74] (a typical input buffered router) and also provides a smaller area overhead. However, the network saturation throughput for Hermes [74] is improved by 55% compared to the base *R-NoC*. The reason being that the parallelism level in *R-NoC* is limited, which arises because multiple input ports are connected to the same lane. This leads to cases where packets with different arrival times are temporarily blocked on the lanes. Therefore, *R-NoC* cannot always support five concurrent data-flows supported in typical router architectures such as Hermes [74].

In order to improve network throughput, additional buffers can be added to the router. However, this will incur additional area and power overheads associated with buffers. Alternatively, this thesis explores varying *R-NoC* configurations with different level of parallelism and lane counts. The evaluation shows that *R-NoC* configuration with *maximum parallelism* (meaning that each input port is attached to a lane, while the other lanes are shared by multiple input ports) provides improved network performance. A performance improvement of 66% and 88% over Hermes [74] for uniform and non-uniform traffic patterns respectively is reported. This confirms the intrinsic ability of *R-NoC* to support specific traffic patterns by means of dynamically allocating buffering resources whenever needed. Also, *R-NoC* provides a network power that is less than that of Hermes [74]. Therefore, the performance improvement of *R-NoC* is not at the expense of area and power. *R-NoC* is shown to provide competitive results for virtual-channels single cycle routers.

One of the objectives of this thesis was to explore other network topologies for performance improvement without incurring significant area/power overheads. For this reason, *R-NoC-D* was proposed for the diagonally-linked mesh network topology (DMesh) [50]. *R-NoC-D* provides significant performance improvements over Hermes [74] and its mesh counterpart (i.e. *R-NoC*) and reduced network power consumption. DMesh requires additional ports which introduce additional area and power overheads. The impact of introducing additional ports on area/power was studied. The evaluation shows that introducing additional ports in Hermes [74] incurs far more area overhead than in *R-NoC*.

8.1 Future works

1. **In-order packet delivery in *R-NoC*.** Packets in *R-NoC* may be delivered in *out-of-order* fashion since packets (not flits) belonging to a particular message may follow different routes in the router. The current *R-NoC* implementations are suitable for connecting high-performance cores with a network stack at the receiving end for taking care of reordering packets. However, the current implementations are not suitable for universal asynchronous receiver-transmitter (UART) devices, where data is transmitted sequentially. Another alternative is to embed all the information to be transmitted in one single packet. However, this will incur reduced network throughput.

2. **Support for virtual-channels (VCs) in *R-NoC*.** Modern high-level protocols require the use of VCs in the NoC in order to ensure functional correctness. An example is in coherence protocols that require isolation of request/reply messages in order to avoid protocol level deadlocks [92]. In cache coherence NoCs, VCs can be dedicated to each message class. Studies have shown that

the MOESI directory-based cache coherence protocol requires at least three VCs in order to avoid protocol level deadlocks [67]. Elastic buffer architecture have been extended to support VCs. The proposed method will be studied in order to understand the cost of a similar implementation in *R-NoC*.

3. **Extending *R-NoC* to support other network topologies**. In order to improve network performance *R-NoC* has been extended to support the diagonally-linked mesh topology. However, there is room for exploring other network topologies. Further works will extend to *three-dimensional* (3D) networks such as 3D mesh network [33]. 3D can exploit vertical interconnect technology such as TSVs to provide faster and low power vertical interconnect [40, 25]. *R-NoC* will be extended to support the so-called concentration networks [58, 93], where multiple cores share a common NoC router for area reduction and performance benefits. Compared to typical input buffered router, extending *R-NoC* to support concentration is expected to incur less area and power overhead due to its distributed architecture.

8.2 Publications

The list of publications includes:

International Conferences

1. **Charles Effiong**, Gilles Sassatelli, Abdoulaye Gamatie. Distributed and Dynamic Shared-Buffer Router for High-Performance Interconnect. In *The International Symposium on Networks-on-Chip (NOCS)*, Seoul, South Korea, October 2017.

2. **Charles Effiong**, Gilles Sassatelli, Abdoulaye Gamatie. Scalable and Power-Efficient Implementation of an Asynchronous Router with Buffer Sharing. In *Euromicro Conference on Digital System Design (DSD 2017), Vienna, Austria*, September 2017.

3. **Charles Effiong**, Gilles Sassatelli, Abdoulaye Gamatie. Roundabout: a Network-on-Chip Router with Adaptive Buffer Sharing. In *IEEE International NEW Circuits And Systems (NEWCAS 2017)*, Strasbourg, France, June 2017.

Under submission

1. **Charles Effiong**, Gilles Sassatelli, Abdoulaye Gamatie. High-Performance Network-on-Chip through Shared and Dynamic Buffer Allocation. [Under submission]

Other publications

I was also involved in other works which are not directly linked to my PhD. These works deals with fast and accurate evaluation of manycore systems using high-level simulation framework [62][63] and process variation impact on 3D-NoC [33]. Related publications are listed below:

1. Khalid Latif, Manuel Selva, **Charles Effiong**, Roman Ursu, Abdoulaye Gamatie, Gilles Sassatelli, Leonardo Zordan, Luciano Ost, Piotr Dziurzanski, and Leandro Soares Indrusiak. 2016. Design space exploration for complex automotive applications: an engine control system case study. In Proceedings of the 2016 Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools (RAPIDO '16).

2. Khalid Latif, **Charles Effiong**, Abdoulaye Gamatié, Gilles Sassatelli, Leonardo Zordan, Luciano Ost, Piotr Dziurzanski and Leandro Indrusiak. An Integrated Framework for Model-Based Design and Analysis of Automotive Multi-Core Systems, Forum on specification & Design Languages (FDL '15), Work-in-Progress Session, Barcelona - Spain, September 2015.

3. **Charles Effiong**, V. Lapotre, A. Gamatie, G. Sassatelli, A. Todri-Sanial and K. Latif, "On the Performance Exploration of 3D NoCs with Resistive-Open TSVs," 2015 IEEE Computer Society Annual Symposium on VLSI, Montpellier, 2015, pp. 579-584.

Bibliography

[1] Arteris S A. "From "Bus" and "Crossbar" to "Network-On-Chip"". In: 2009.

[2] Pablo Abad et al. "Rotary Router: An Efficient Architecture for CMP Interconnection Networks". In: *SIGARCH Comput. Archit. News* 35.2 (June 2007), pp. 116–125. ISSN: 0163-5964.

[3] N. R. Adiga et al. "Blue Gene/L torus interconnection network". In: *IBM Journal of Res. and Dev.* 49.2.3 (2005), pp. 265–276. ISSN: 0018-8646. DOI: 10.1147/rd.492.0265.

[4] Adrijean Adriahantenaina et al. "SPIN: a scalable, packet switched, on-chip micro-network". In: *Proceedings of the conference on Design, Automation and Test in Europe: Designers' Forum-Volume 2*. IEEE Computer Society. 2003, p. 20070.

[5] A. Alhussien, C. Wang, and N. Bagherzadeh. "A scalable delay insensitive asynchronous NoC with adaptive routing". In: *2010 17th Int'l Conf. on Telecom.* 2010, pp. 995–1002.

[6] William John Bainbridge. "Asynchronous Systems on Chip Interconnect". PhD thesis. Department of Computer Science, The University of Manchester, 2000.

[7] E. Beigne et al. "An asynchronous NOC architecture providing low latency service and its multi-level design framework". In: *11th IEEE International Symposium on Asynchronous Circuits and Systems*. 2005, pp. 54–63.

[8] S. Bell et al. "TILE64 - Processor: A 64-Core SoC with Mesh Interconnect". In: *2008 IEEE International Solid-State Circuits Conference - Digest of Technical Papers*. 2008, pp. 88–598.

[9] L. Benini and G. De Micheli. "Networks on chips: a new SoC paradigm". In: *Computer* 35.1 (2002), pp. 70–78.

[10] D. Bertozzi et al. "NoC synthesis flow for customized domain specific multiprocessor systems-on-chip". In: *IEEE Transactions on Parallel and Distributed Systems* 16.2 (2005), pp. 113–129.

[11] T. Bjerregaard and J. Sparso. "Implementation of guaranteed services in the MANGO clockless network-on-chip". In: *IEE Proceedings - Computers and Digital Techniques* 153.4 (2006), pp. 217–229.

[12] Tobias Bjerregaard and Shankar Mahadevan. "A Survey of Research and Practices of Network-on-chip". In: *ACM Comput. Surv.* 38.1 (June 2006). ISSN: 0360-0300.

[13] Tobias Bjerregaard and Jens Sparso. "A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip". In: *Design, Automation and Test in Europe, 2005. Proceedings*. IEEE. 2005, pp. 1226–1231.

[14] S. Borkar. "NoCs: What's the point?" In: *NSF Workshop on Emerging Tech. for Interconnects (WETI)*. 2012.

[15] Shekhar Borkar. "Thousand Core Chips: A Technology Perspective". In: *Proceedings of the 44th Annual Design Automation Conference*. DAC '07. San Diego, California: ACM, 2007, pp. 746–749. ISBN: 978-1-59593-627-1. DOI: 10.1145/1278480.1278667. URL: http://doi.acm.org/10.1145/1278480.1278667.

[16] Cadence. *Cadence RTL Compiler*. URL: https://www.cadence.com/.

[17] J. Carmona et al. "Elastic Circuits". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 28.10 (2009), pp. 1437–1455. ISSN: 0278-0070.

[18] C. Carrion et al. "A flow control mechanism to avoid message deadlock in k-ary n-cube networks". In: *Proceedings 4th HiPC*. 1997, pp. 322–329.

[19] L. Chen and T. M. Pinkston. "Worm-Bubble Flow Control". In: *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*. 2013, pp. 366–377.

[20] L. Chen, R. Wang, and T. M. Pinkston. "Critical Bubble Scheme: An Efficient Implementation of Globally Aware Network Flow Control". In: *IEEE Int'l Parallel Distributed Processing Symp.* 2011, pp. 592–603.

[21] Yancang Chen et al. "A single-cycle output buffered router with layered switching for Networks-on-Chips". In: *Computers & electrical engineering* 38.4 (2012), pp. 906–916.

[22] Yancang Chen et al. "A single-cycle output buffered router with layered switching for Networks-on-Chips". In: *Computers and Electrical Engineering* 38 (2012), pp. 906–916.

[23] W. J. Dally. "Virtual-channel flow control". In: *IEEE Transactions on Parallel and Distributed Systems* 3.2 (1992), pp. 194–205. ISSN: 1045-9219.

[24] William Dally and Brian Towles. *Principles and Practices of Interconnection Networks*. San Francisco, CA, USA: Morgan Kaufmann, 2003. ISBN: 0122007514.

[25] Florian Darve et al. "Physical implementation of an asynchronous 3D-NoC router using serial vertical links". In: *VLSI (ISVLSI), 2011 IEEE Computer Society Annual Symposium on*. IEEE. 2011, pp. 25–30.

[26] Mark E. Dean, Ted E. Williams, and David L. Dill. "Efficient Self-timing with Level-encoded 2-phase Dual-rail (LEDR)". In: *Proceedings of the 1991 University of California/Santa Cruz Conference on Advanced Research in VLSI*. Cambridge, MA, USA: MIT Press, 1991, pp. 55–70. ISBN: 0-262-19308-6.

[27] R. Dick. *Embedded System Synthesis Benchmark Suites (e3s)*. URL: http://ziyang.eecs.umich.edu/dickrp/e3s/.

[28] John Dielissen et al. "Concepts and implementation of the Philips network-on-chip". In: *IP-Based SoC Design*. 2003, pp. 1–6.

[29] Giorgos Dimitrakopoulos, Anastasios Psarras, and Ioannis Seitanidis. *Microarchitecture of Network-on-Chip Routers: A Designer's Perspective*. Springer Publishing Company, Incorporated, 2014. ISBN: 1461443008, 9781461443001.

[30] Rostislav (Reuven) Dobkin, Ran Ginosar, and Avinoam Kolodny. "QNoC asynchronous router". In: *Integration, the VLSI Journal* 42.2 (2009), pp. 103 – 115.

[31] José Duato. "A Necessary and Sufficient Condition for Deadlock-Free Adaptive Routing in Wormhole Networks". In: *IEEE Trans. Parallel Distrib. Syst.* 6.10 (Oct. 1995), pp. 1055–1067. ISSN: 1045-9219.

[32] Jose Duato, Sudhakar Yalamanchili, and Lionel M Ni. *Interconnection networks: an engineering approach*. Morgan Kaufmann, 2003.

[33] C. Effiong et al. "On the Performance Exploration of 3D NoCs with Resistive-Open TSVs". In: *2015 IEEE Computer Society Annual Symposium on VLSI*. 2015, pp. 579–584.

[34] Charles Effiong, Gilles Sassatelli, and Abdoulaye Gamatie. "Distributed and Dynamic Shared-Buffer Router for High-Performance Interconnect". In: *International Symposium on Networks-on-Chip (NOCS)*. NOCS'17. Seoul, South Korea, 2017.

[35] Charles Effiong, Gilles Sassatelli, and Abdoulaye Gamatie. "Roundabout: a Network-on-Chip Router with Adaptive Buffer Sharing". In: *IEEE International New Circuits and Systems Conference*. Newcas'17. Strasbourg, France, 2017.

[36] Charles Effiong, Gilles Sassatelli, and Abdoulaye Gamatie. "Scalable and Power-Efficient Implementation of an Asynchronous Router with Buffer Sharing". In: *Euromicro Conference on Digital System Design*. DSD'17. Vienna, Austria, 2017.

[37] H. Elmiligi et al. “A delay-aware topology-based design for Networks-on-Chip applications”. In: *2009 4th International Design and Test Workshop (IDT)*. 2009, pp. 1–5.

[38] C. Fallin et al. “MinBD: Minimally-Buffered Deflection Routing for Energy-Efficient Interconnect”. In: *2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip*. 2012, pp. 1–10.

[39] Chris Fallin, Chris Craik, and Onur Mutlu. “CHIPPER: A Low-complexity Bufferless Deflection Router”. In: *Proceedings of the 2011 IEEE 17th International Symposium on High Performance Computer Architecture*. HPCA ’11. 2011, pp. 144–155. ISBN: 978-1-4244-9432-3.

[40] Brett Stanley Feero and Partha Pratim Pande. “Networks-on-chip in a three-dimensional environment: A performance evaluation”. In: *IEEE Transactions on computers* 58.1 (2009), pp. 32–45.

[41] Marc Galceran-Oms. “Automatic Pipelining of Elastic Systems”. PhD thesis. universitat politècnica de catalunya, 2011.

[42] M. Galles. “Spider: a high-speed network interconnect”. In: *IEEE Micro* 17.1 (1997), pp. 34–39. ISSN: 0272-1732.

[43] Mentor Graphics. *ModemSIM*. URL: https://www.mentor.com/products/fv/modelsim.

[44] H. Hassan, A. Shalaby, and H. Kim. “DPSB: Dual port shared buffer mechanism for efficient buffer utilization in Network on Chip routers”. In: *2015 International SoC Design Conference (ISOCC)*. 2015, pp. 135–136.

[45] F. O. Hatem and T. N. Kumar. “A low-area asynchronous router for clock-less network-on-chip on a FPGA”. In: *2013 IEEE Symposium on Computers Informatics (ISCI)*. 2013, pp. 152–158.

[46] M. Hayenga, N. Enright Jerger, and M. Lipasti. “SCARAB: A single cycle adaptive routing and bufferless network”. In: *2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 2009, pp. 244–254.

[47] J. Henkel, W. Wolf, and S. Chakradhar. “On-chip networks: a scalable, communication-centric embedded system design paradigm”. In: *17th International Conference on VLSI Design. Proceedings.* 2004, pp. 845–851.

[48] Y. Hoskote et al. “A 5-GHz Mesh Interconnect for a Teraflops Processor”. In: *IEEE Micro* 27.5 (2007), pp. 51–61. ISSN: 0272-1732.

[49] Jingcao Hu and R. Marculescu. “DyAD - smart routing for networks-on-chip”. In: *Proceedings. 41st Design Automation Conference, 2004.* 2004, pp. 260–263.

[50] Wen-Hsiang Hu, Seung Eun Lee, and Nader Bagherzadeh. "DMesh: a diagonally-linked mesh network-on-chip architecture". In: *Network on Chip Architectures* (2008), p. 14.

[51] intel. *Intel® Xeon Phi™*. 2016. URL: https://www.intel.com/content/www/us/en/products/processors/xeon-phi/xeon-phi-processors/7230f.html.

[52] ITRS. "The International Technology Roadmap for Semiconductors (ITRS)". In: (Jan. 2011).

[53] H. M. Jacobson et al. "Synchronous interlocked pipelines". In: *Proceedings Eighth International Symposium on Asynchronous Circuits and Systems*. 2002, pp. 3–12.

[54] S. A. R. Jafri et al. "Adaptive Flow Control for Robust Performance and Energy". In: *2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*. 2010, pp. 433–444.

[55] J. Jose et al. "DeBAR: Deflection based adaptive router with minimal buffering". In: *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*. 2013, pp. 1583–1588.

[56] Daewook Kim, Manho Kim, and Gerald E Sobelman. "Asynchronous FIFO interfaces for GALS on-chip switched networks". In: *ISOCC* (2005), pp. 186–189.

[57] Ho Won Kim et al. "Adaptive Virtual Cut-Through as a Viable Routing Method". In: *Journal of Parallel and Distributed Computing* 52.1 (1998), pp. 82 –95. ISSN: 0743-7315.

[58] J. Kim, J. Balfour, and W. Dally. "Flattened Butterfly Topology for On-Chip Networks". In: *40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007)*. 2007, pp. 172–182.

[59] P. Kongetira, K. Aingaran, and K. Olukotun. "Niagara: a 32-way multithreaded Sparc processor". In: *IEEE Micro* 25.2 (2005), pp. 21–29. ISSN: 0272-1732.

[60] P. Kumar et al. "Exploring concentration and channel slicing in on-chip network router". In: *2009 3rd ACM/IEEE International Symposium on Networks-on-Chip*. 2009, pp. 276–285.

[61] K. Latif, T. Seceleanu, and H. Tenhunen. "Power and Area Efficient Design of Network-on-Chip Router through Utilization of Idle Buffers". In: *2010 17th IEEE International Conference and Workshops on Engineering of Computer Based Systems*. 2010, pp. 131–138.

[62] Khalid Latif et al. "An integrated framework for model-based design and analysis of automotive multi-core system". In: *Forum on specification & Design Languages (FDL '15), Work-in-Progress Session*. Barcelona, Spain, 2015.

[63] Khalid Latif et al. "Design Space Exploration for Complex Automotive Applications: An Engine Control System Case Study". In: *Proceedings of the 2016 Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools*. RAPIDO '16. Prague, Czech Republic: ACM, 2016, 2:1–2:7. ISBN: 978-1-4503-4072-4.

[64] Jian Liang, S. Swaminathan, and R. Tessier. "ASOC: a scalable, single-chip communications architecture". In: *Proceedings 2000 International Conference on Parallel Architectures and Compilation Techniques (Cat. No.PR00622)*. 2000, pp. 37–46.

[65] ARM Ltd. *Advanced Microcontroller Bus Architecture*. 1996. URL: https://www.arm.com/products/system-ip/amba-specifications.

[66] S. Ma et al. "Leaving One Slot Empty: Flit Bubble Flow Control for Torus Cache-Coherent NoCs". In: *IEEE Trans. on Comp.* 64.3 (2015), pp. 763–777. ISSN: 0018-9340.

[67] Milo M. K. Martin et al. "Multifacet's General Execution-driven Multiprocessor Simulator (GEMS) Toolset". In: *SIGARCH Comput. Archit. News* 33.4 (Nov. 2005), pp. 92–99. ISSN: 0163-5964.

[68] Peggy B McGee et al. "A level-encoded transition signaling protocol for high-throughput asynchronous global communication". In: *Asynchronous Circuits and Systems, 2008. ASYNC'08. 14th IEEE International Symposium on*. IEEE. 2008, pp. 116–127.

[69] A. Mello et al. "Evaluation of current QoS Mechanisms in Networks on Chip". In: *2006 International Symposium on System-on-Chip*. 2006, pp. 1–4.

[70] A. Mello et al. "Virtual Channels in Networks on Chip: Implementation and Evaluation on Hermes NoC". In: *2005 18th Symposium on Integrated Circuits and Systems Design*. 2005, pp. 178–183.

[71] L. Mhamdi, K. Goossens, and I. V. Senin. "Buffered Crossbar Fabrics Based on Networks on Chip". In: *2010 8th Annual Communication Networks and Services Research Conference*. 2010, pp. 74–79.

[72] G. Michelogiannakis and W. J. Dally. "Elastic Buffer Flow Control for On-Chip Networks". In: *IEEE Transactions on Computers* 62.2 (2013), pp. 295–309. ISSN: 0018-9340.

[73] G. Michelogiannakis and W. J. Dally. "Router designs for elastic buffer on-chip networks". In: *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*. 2009, pp. 1–10.

[74] Fernando Moraes et al. "HERMES: An Infrastructure for Low Area Overhead Packet-switching Networks on Chip". In: *Integr. VLSI J.* 38.1 (Oct. 2004), pp. 69–93. ISSN: 0167-9260.

[75] Thomas Moscibroda and Onur Mutlu. "A Case for Bufferless Routing in On-chip Networks". In: *SIGARCH Comput. Archit. News* 37.3 (June 2009), pp. 196–207. ISSN: 0163-5964.

[76] R. Mullins, A. West, and S. Moore. "Low-Latency Virtual-Channel Routers for On-Chip Networks". In: *Sigarch Comp. Archit. News* (Mar. 2004). ISSN: 0163-5964.

[77] S. M. Nowick and M. Singh. "High-Performance Asynchronous Pipelines: An Overview". In: *IEEE Design Test of Computers* 28.5 (2011), pp. 8–22. ISSN: 0740-7475.

[78] N. Onizawa et al. "High-Throughput Compact Delay-Insensitive Asynchronous NoC Router". In: *IEEE Trans on Comp* 63.3 (2014), pp. 637–649.

[79] J. D. Owens et al. "Research Challenges for On-Chip Interconnection Networks". In: *IEEE Micro* 27.5 (2007), pp. 96–108. ISSN: 0272-1732.

[80] G. Oxman and S. Weiss. "Simple method to reduce congestion in bufferless network-on-chip". In: *Electronics Letters* 50.8 (2014), pp. 581–583. ISSN: 0013-5194.

[81] M. Palesi et al. "Application Specific Routing Algorithms for Networks on Chip". In: *IEEE Transactions on Parallel and Distributed Systems* 20.3 (2009), pp. 316–330.

[82] Ivan Miro Panades and Alain Greiner. "Bi-synchronous fifo for synchronous circuit communication well suited for network-on-chip in gals architectures". In: *Networks-on-Chip, 2007. NOCS 2007. First International Symposium on*. IEEE. 2007, pp. 83–94.

[83] J. Parkhurst, J. Darringer, and B. Grundmann. "From Single Core to Multi-Core: Preparing for a new exponential". In: *2006 IEEE/ACM International Conference on Computer Aided Design*. 2006, pp. 67–72.

[84] Li-Shiuan Peh, Stephen W. Keckler, and Sriram Vangal. "On-Chip Networks for Multicore Systems". In: *Multicore Processors and Systems*. Ed. by Stephen W. Keckler, Kunle Olukotun, and H. Peter Hofstee. Boston, MA: Springer US, 2009, pp. 35–71.

[85] Julian J. H. Pontes et al. "Hermes-A - An Asynchronous NoC Router with Distributed Routing". In: *PATMOS*. Vol. 6448. LNCS. Springer, 2010, pp. 150–159.

[86] JULIAN JOSÉ HILGEMBERG PONTES. "Soft Error Mitigation in Asynchronous Networks-on-Chip". PhD thesis. Pontifícia Universidade Católica do Rio Grande do Sul, 2012.

[87] V. Puente et al. "The Adaptive Bubble Router". In: *J. Parallel Distrib. Comput.* 61.9 (Sept. 2001), pp. 1180–1208. ISSN: 0743-7315.

[88] Rezaur Rahman. "Introduction to Xeon Phi Architecture". In: *Intel® Xeon Phi™ Coprocessor Architecture and Tools: The Guide for Application Developers*. Berkeley, CA: Apress, 2013, pp. 3–14.

[89] Ville Rantala, Teijo Lehtonen, Juha Plosila, et al. *Network on chip routing algorithms*. 2006.

[90] E. Rijpkema et al. "Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip". In: *2003 Design, Automation and Test in Europe Conference and Exhibition*. 2003, pp. 350–355.

[91] Mostafa S. Sayed et al. "Flexible Router Architecture for Network-on-chip". In: *Comput. Math. Appl.* 64.5 (Sept. 2012), pp. 1301–1310. ISSN: 0898-1221.

[92] I. Seitanidis et al. "ElastiStore: An elastic buffer architecture for Network-on-Chip routers". In: *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*. 2014, pp. 1–6.

[93] F. N. Sibai. "Resource Sharing in Networks-on-Chip of Large Many-core Embedded Systems". In: *2009 International Conference on Parallel Processing Workshops*. 2009, pp. 513–519.

[94] S. Z. Sleeba, J. Jose, and M. M. G. "HiPAD: High Performance Adaptive Deflection Router for On-Chip Mesh Networks". In: *2015 Fifth International Conference on Advances in Computing and Communications (ICACC)*. 2015, pp. 16–19.

[95] V. Soteriou et al. "A High-Throughput Distributed Shared-Buffer NoC Router". In: *IEEE Computer Architecture Letters* 8.1 (2009), pp. 21–24. ISSN: 1556-6056.

[96] Jens Spars and Steve Furber. *Principles of Asynchronous Circuit Design: A Systems Perspective*. 1st. Springer Pub. Company, Inc., 2010. ISBN: 1441949364, 9781441949363.

[97] Synopsys. *Synopsys design compiler*. URL: https://www.synopsys.com/support/training/rtl-synthesis/design-compiler-rtl-synthesis.html.

[98] M. B. Taylor et al. "The Raw microprocessor: a computational fabric for software circuits and general-purpose programs". In: *IEEE Micro* 22.2 (2002), pp. 25–35. ISSN: 0272-1732.

[99] Y. Thonnart, P. Vivet, and F. Clermidy. "A fully-asynchronous low-power framework for GALS NoC integration". In: *DATE 2010*. 2010, pp. 33–38.

[100] Erik B. van der Tol and Egbert G. Jaspers. "Mapping of MPEG-4 decoding on a flexible architecture platform". In: vol. 4674. 2001, pp. 1–13.

[101] A. T. Tran and B. M. Baas. "Achieving High-Performance On-Chip Networks With Shared-Buffer Routers". In: *IEEE Trans on Very Large Scale Integration (VLSI) Systems* 22.6 (2014), pp. 1391–1403. ISSN: 1063-8210.

[102] A. T. Tran and B. M. Baas. "DLABS: A dual-lane buffer-sharing router architecture for networks on chip". In: *2010 IEEE Workshop On Signal Processing Systems*. 2010, pp. 327–332.

[103] A. T. Tran, D. N. Truong, and B. M. Baas. "A complete real-time 802.11a baseband receiver implemented on an array of programmable processors". In: *2008 42nd Asilomar Conference on Signals, Systems and Computers*. 2008, pp. 165–170.

[104] Balaji Venu. "Multi-core processors-an overview". In: *arXiv preprint arXiv:1110.3535* (2011).

[105] Xiaofang Maggie Wang and Leeladhar Bandi. "X-Network: An area-efficient and high-performance on-chip wormhole interconnect network". In: *Microprocessors and Microsystems* 37.8 (2013), pp. 1208–1218.

[106] X. Y. Xiang and N. F. Tzeng. "Deflection Containment for Bufferless Network-on-Chips". In: *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 2016, pp. 113–122.

[107] M. K. Yadav, M. R. Casu, and M. Zamboni. "DVFS Based on Voltage Dithering and Clock Scheduling for GALS Systems". In: *2012 IEEE 18th International Symposium on Asynchronous Circuits and Systems*. 2012, pp. 118–125.