Technische Universität München

Lehrstuhl für Integrierte Systeme

A Selective Packet Discard Technique for Efficient Deadlock Recovery in Networks-on-Chip

Andreas Johann Lankes

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. Ir. Majid Zamani

Prüfer der Dissertation:

1. Univ.-Prof. Dr. sc.techn. Andreas Herkersdorf
2. Univ.-Prof. Dr.-Ing. Dr. h. c. Jürgen Becker,
   Karlsruher Institut für Technologie

Die Dissertation wurde am 30.09.2014 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 09.03.2015 angenommen.

TECHNISCHE UNIVERSITÄT MÜNCHEN

Abstract

Department of Electrical Engineering

Doctor of Engineering

A Selective Packet Discard Technique for Efficient Deadlock Recovery in Networks-on-Chip

by Andreas Johann Lankes

The ever increasing complexity of Systems-on-Chip (SoCs) and the hereby also increasing communication requirements lead to the necessity of new on-chip communication concepts. The concept of NoCs was introduced to provide scalable, adaptable and distributed on-chip interconnection infrastructures. The research field of NoCs poses many research challenges. This dissertation focuses on the problem of message dependent deadlocks, a special kind of deadlocks, in the context of fault tolerance. Furthermore, hierarchical network architectures were studied and optimized for shared resource access.

As state of the art NoCs principally assume lossless communication, deadlock avoidance techniques are the first choice to solve message dependent deadlocks. This leads to high costs for actually rarely occurring events. As an alternative approach, a progressive deadlock recovery concept is proposed and evaluated in the thesis. However, as sensitivity of future CMOS technologies to all forms of manufacturing and environmental variations increases, lossless communication can no longer be simply assumed. Thus, measures to cope with data corruption or packet loss will become necessary in the future anyhow. This change calls for a completely different approach, that is able to efficiently solve the problem of message dependent deadlock and to provide fault tolerance, in combination.

This thesis proposes sdNoC, a NoC deliberately employing the perceived deficiency of packet loss as a feature. The selective discard of stuck packets in the NoC, a proven practice in computer networks, solves all types of deadlocks. Especially counter measures to message-dependent deadlocks otherwise lead to high costs either in terms of throughput or chip area. The extensive evaluation of sdNoC in terms of chip area requirements and performance shows that the selective discard of temporarily congested packets can even help to improve the effective throughput. The end-to-end retransmission mechanism required for the reliable communication, then also provides lossless communication for the cores.

TECHNISCHE UNIVERSITÄT MÜNCHEN

Zusammenfassung

Department of Electrical Engineering

Doctor of Engineering

A Selective Packet Discard Technique for Efficient Deadlock Recovery in Networks-on-Chip

by Andreas Johann Lankes

Die ständig steigende Komplexität von Systems-on-Chip (SoCs) und die dadurch ebenso steigenden Kommunikationsanforderungen machen neue On-Chip Kommunikationskonzepte notwendig. Daher wurde das Konzept von Networks-on-Chip entwickelt, das anpassbare, verteilte und skalierbare Kommunikationsinfrastrukturen ermöglicht. Das Themenfeld Networks-on-Chip (NoCs) bietet ein sehr weites Feld von Forschungsmöglichkeiten. Schwerpunkt dieser Arbeit ist das Problem der nachrichtenabhängigen Deadlocks, eine spezielle Art von Deadlocks, im Zusammenhang mit Fehlertoleranz. Daneben wurden auch hierarchische Netzwerktopologien untersucht und für den Zugriff auf gemeinsame Ressourcen optimiert.

Aktuelle On-Chip Netzwerke basieren zumeist auf der Annahme von verlustloser Kommunikation. Bei der Deadlockproblematik sind daher auch Techniken zur Vermeidung die erste Wahl. Die Verwendung dieser resultiert jedoch in hohen Kosten. Als Alternative schlägt diese Arbeit ein progressives Deadlock Recovery Konzept vor, und analysiert und bewertet es. Da jedoch die Anfälligkeit zukünftiger CMOS Technologien gegenüber einer Vielzahl von Herstellungs- und Umgebungseinflüssen steigt, wird die Annahme verlustloser Kommunikation in Zukunft nicht mehr möglich sein. Die Einführung von Maßnahmen zum Umgang mit Datenkorruption oder Paketverlust im Netzwerk wird dann unumgänglich sein. Dieser Wandel erfordert einen vollständig neuen Ansatz, der sowohl das Problem der nachrichtenabhängigen Deadlocks effizient löst, als auch Fehlertoleranz bietet.

Als Ansatz schlägt diese Dissertation sdNoC vor, ein NoC das Paketverluste, eigentlich eine Unzulänglichkeit, bewusst einsetzt. Das Verwerfen einzelner, blockierter Pakete, eine gängige Praxis in Computernetzwerken, löst Deadlocks jeglicher Art. Insbesondere Maßnahmen zur Lösung nachrichtenabhängiger Deadlocks führen ansonsten zu hohen Kosten. Eine umfangreiche Untersuchung von sdNoC hinsichtlich der benötigten Chipfläche und der Performanz zeigt, dass das Verwerfen einzelner, temporär blockierter Pakete sogar den effektiven Durchsatz des Netzes verbessern. Die Funktionalität

zum erneuten Übertragen von Paketen, bietet sowohl Fehlertoleranz als auch verlustlose Kommunikation zwischen den Kommunikationspartnern.

Acknowledgements

I would like to express my appreciation and gratitude to my doctoral advisor Prof. Dr. Andreas Herkersdorf for having given me the opportunity to work on such an interesting research project and the guidance and mentorship he provided to me. Thereby, i was not only able to broaden my knowledge, but also gained valuable experience in my research work as well as in teaching. I am especially thankful for the experience i gained during the teaching at the Nanyang Technological University in Singapore. I would also like to thank Prof. Dr. Dr. Jürgen Becker for acting as my secondary examiner on the doctoral examination commission.

I also want to thank Dr. Thomas Wild who has helped a lot in the creative process of this dissertation. He was always available for discussions whether regarding research direction or technical aspects. I also appreciate his detailed feedback he gave me during the work on on my publications and his feedback on this dissertation.

I am also very grateful to Stefan Wallentowitz for all the valuable discussions on a wide variety of topics in the field of NoCs. He greatly contributed to the development of the sdNoC hardware model. I also want to thank him for the fast and straightforward IT support.

I want to thank Roman Plyaskin, with whom i shared the office for 6 years, for the long and helpful discussions and the good office atmosphere. I am also grateful to all other colleagues at the institute for the good cooperation and the positive working atmosphere.

Last but no least, i am grateful to my family for their support. I am deeply grateful to my fiancée Astrid who supported and motivated me. Especially during the end of the writing process she shouldered far more than her fair share of the household burdens to enable my work on this thesis.

Contents

List of Figures

List of Tables

Abbreviations

BE	**B**est **E**ffort
BW_{link}	**B**and**w**idth of link
CDG	**C**hannel **D**ependency **G**raph
CMOS	**C**omplementary **M**etal **O**xide **S**emiconductor
CPU	**C**entral **P**rocessing **U**nit
CRC	**C**yclic **R**edundancy **C**heck
DC	**D**eadlock **C**hannel
DRU	**D**eadlock **R**ecovery **U**nit
DMA	**D**irect **M**emory **A**ccess
DOR	**D**imension **O**rdered **R**outing
ECC	**E**rror **C**orrection **C**ode
FIFO	**F**irst **I**n **F**irst **O**ut
GALS	**G**lobally **A**synchronous **L**ocally **S**ynchronous
IPT	**I**nter **P**rocessor **T**raffic
GS	**G**uaranteed **S**ervice
IO	**I**nput **O**utput
MEM	**Mem**ory
NA	**N**etwork **A**dapter also called network interface
NI	**N**etwork **I**interface also called network adapter
NoC	**N**etwork-**o**n-**C**hip
OMNeT++	**O**bjective **M**odular **Ne**twork **T**estbed in C++
Pdr	**P**rogressive **d**eadlock **r**recovery
RED	**R**andom **E**arly **D**iscard
sdNoC	**s**elective packet **d**iscard **N**etwork-**o**n-**C**hip
QoS	**Q**uality **o**f **S**ervice

SAF	**S**tore **a**nd **F**orward
So	**S**trict **o**rdering
SoC	**S**ystem **o**n **C**hip
SCC	**S**ingle-chip **C**loud **C**omputer
SDM	**S**pace **D**ivision **M**ultiplexing
St	**St**andard network
TCP	**T**ransmission **C**ontrol **P**rotocol
TDMA	**T**ime **D**ivision **M**ultiple **A**ccess
TDRN	**T**oken **D**istribution **R**ing **N**etwork
TMR	**T**riple **M**odular **R**edundancy
TSMC	**T**aiwan **S**emiconductor **M**anufacturing **C**ompany
VC	**V**irtual **C**hannel
VDC	**V**irtual **D**eadlock **C**hannel

Symbols

BW_{link}	Bandwidth of a link	bits/sec
BW_{aggr}	Aggregated link bandwidth of network	BW_{link}
d	Network distance	hops
N	Number of tiles of a network	
w	Link width	

Chapter 1

Introduction

The complexity of integrated circuits increases steadily, due to the constant shrinking of feature sizes in semi-conductor fabrication processes. This trend is described by Moore's law [2], which predicts a doubling of complexity approximately every 20 months. In the past, this development has led to the emerge of Systems-on-Chip (SoCs) and Multi-processor SoCs [3], which unite complete systems consisting of many modules such as processors, hardware accelerators, memories, etc., on one chip. As Moore's law still holds true, the number of modules or cores on SoCs is still constantly increasing. This leads us from SoCs consisting of a few cores to many-core processors or systems with up to a thousand cores [4, 5].

However, traditional on-chip interconnects, such as buses or crossbars, can no longer provide the correspondingly also growing requirements in communication bandwidth. Both, buses as well as crossbars, do not scale. Since a bus is a shared medium, the bandwidth per core decreases with every core that is connected to it. To make matters worse, the maximum clock frequency that a bus can be operated at, reduces with its increasing geometrical expansion due to the wiring delay [6]. While crossbars can provide higher bandwidths up to a limited number of cores, their chip area requirements explode with rising numbers of cores.

In recent years, the paradigm of Networks-on-Chip (NoCs) [6–11] has been developed with the intention of solving the shortcomings of the traditional on-chip interconnects

and providing a scalable communication concept for the future. NoCs are not a completely new concept of on-chip communication. Instead, it rather unifies existing techniques. NoCs apply the segmentation of wires or an interconnection infrastructure, as it was already done in bus bridges. First of all, this keeps wires short and thus avoids the decrease of the clock frequency. In contrast, the short point-to-point links employed in NoCs enable high clocks. Second, the segmentation of an interconnect leads to a distributed communication infrastructure, which enables (to some extent) parallel transfers for all components on a chip.

An alternative way to consider the concept of NoCs, is to view them as downscaled computer networks. Due to significantly different boundary conditions in the on-chip world, this downscaling process cannot be limited to just copying the concepts. Concepts and techniques employed in NoC routers must become simpler and more compact than in off-chip routers, in order to limit chip area requirements, allow multiple replications on a chip together with the computation cores and to reduce the transfer latencies to a minimum. In addition, the buffer space requirements of the employed protocols must be significantly reduced in comparison to computer network routers.

NoCs are already used in SoCs with high communication requirements or geometrically large SoCs that require a highly distributed communication architecture in order to still provide the required throughput. Up to this date, a multitude of NoC architectures have been proposed, with many different techniques in terms of forwarding, switching, etc. Nevertheless, some of these techniques are more popular than others. The most widespread NoCs [12] are based on packet switching and wormhole forwarding [8, 13]. Such NoCs are the focus of this work.

1.1 Motivation

Networks-on-Chip still pose many research challenges [14]. Especially reducing the transfer latencies and increasing the throughput, while reducing the chip area requirements and the power consumption are central targets. This becomes obvious when analyzing Intel's Teraflops processor [15, 16] as an example. The NoC of this 80 core processor requires 25% of the total chip area and up to 30% of the total power consumption.

The performance and cost of a NoC depends significantly on its network topology. The topology determines the interconnection of the network's routers. The higher the degree of connectivity, this means the more interconnections between the routers in a NoC, the higher the throughput (path diversity, see Section 2.3) and the lower the latency. For example, a ring network provides only a low degree of connectivity, whereas a mesh network has a higher degree of connectivity. With an increasing number of links the number of packets or communication flows that have to share the same links is reduced. However, due to the increasing number of ports per router the complexity of the router architecture also rises. This means more complex arbitration and a more complex internal switch, which interconnects the input and output channels, and additional FIFO queues for each port. Since buffers dominate the chip area requirements of a NoC [17], the chip area requirements rise. In addition, buffers also have a major impact on the power consumption [15] of a NoC. Thus, saving buffer space is a key research problem in NoCs [18]. To make matters worse, not only the topology determines the number of buffers but also measures to avoid or handle deadlocks affect the number of buffers (virtual channels). Thus, these issues also have a significant impact on cost and performance of a NoC.

Deadlocks are a key research problem [18] in Networks-on-Chip, much research has been done on this field. Many deadlock free routing algorithms have been developed: XY-routing [19], turn model based routing [20], odd even routing [21, 22], etc. However, all these routing algorithms just consider the network itself and thus depend on the pre-condition that the network adapters are able to completely receive all packets from the network [23]. However, with transport level protocols used in the communication between cores, this assumption may not always be valid. As a consequence, message dependent deadlocks can occur. These pose an additional problem that has to be solved on top of routing dependent deadlocks of the network itself and that again leads to additional costs. However, many NoCs do not consider message dependent deadlocks (Hermes [24], SPIN, Xpipes [25, 26], MANGO [27]) at all. Recent NoC based many-core processors, like the Intel Single-chip Cloud Computer [28] that consider this type of deadlock, all apply deadlock avoidance.

Up to this date, on-chip communication is generally designed strictly lossless. However, future CMOS technologies will get increasingly sensitive to all forms of manufacturing and environmental variations. Under these circumstances, simply assuming error free

communication will no longer work and keeping a lossless approach will become more and more costly.

Considering all the above described issues and problems already makes designing NoCs a challenge, as many of the mentioned metrics and techniques form a trade-off. Last but not least, the NoC architecture has to be adapted to the application's requirements in terms of performance. Thus, a one-fits-all approach is impossible. Instead, the topology, the strategy against deadlocks, all the other architectural parameters have to be carefully selected and investigated. Hereby, fault tolerance [29, 30] aspects should not be omitted.

1.2 Thesis Contribution

This dissertation contributes to two topics in the field of on-chip networks: The development and evaluation of deadlock recovery for message dependent deadlocks and the introduction and evaluation of hierarchical, multi-topology NoCs.

The first topic is the development and evaluation of anti-deadlock strategies, especially for message dependent deadlocks, that are scalable and adaptable to the application's requirements. Hereby, this thesis focuses on deadlock recovery techniques. Both, progressive [31] and regressive [32] deadlock recovery techniques for NoCs have been investigated and evaluated in depth. Both deadlock recovery schemes solve all types of deadlocks with minimal additional buffer overhead, which is even less than that of the compared standard technique for handling message dependent deadlocks. The concepts also enable an adaptation in terms of cost and performance to the application's requirements.

Especially my proposed regressive deadlock recovery concept, called sdNoC, can provide equal throughput compared to alternative techniques that require double buffer space. In addition, sdNoC resolves local congestions around hot-spots and is even able to provide fault tolerance against all kinds of soft error in the network.

The second field of contribution addresses the topic of network topologies. This work proposes hierarchical, multi-topology NoCs [1] that allow a more profound and detailed adaptation to an application's requirements. This can be achieved by individually adapting the different sub-networks to the corresponding requirements of the sub partitions

of the system. Thus, over-provisioning of the interconnect for sub-partitions of a system can be avoided. In addition, hierarchical, multi topology NoCs enable an optimized connection of shared resources in the logical center of the network.

1.3 Structure of this Thesis

The following list gives an overview of the structure of this thesis.

- Chapter 2 introduces the basics of NoCs and gives a deeper insight on the state of the art in the fields which are relevant for this thesis. The basic components of NoCs, i.e. routers and network adapters, are presented and explained in more detail. Based on this, the communication protocols applied in NoCs as well as their topologies are introduced. Then, the state of the art in the fields which are in the focus of this thesis is provided. The problem of deadlocks, the different types of deadlocks and the various ways of solving this problem are elaborately discussed. This chapter concludes with an overview of bufferless NoCs and fault tolerance in on-chip networks.

- Chapter 3 presents the main contribution of this thesis, the progressive and the regressive deadlock recovery concept (sdNoC). Both concepts are introduced, discussed and evaluated in detail. The system level performance evaluation, in which many different parameters are investigated, is performed with the help of simulation models. In addition, the concepts are also evaluated in terms of buffer space requirements. Hereby, the improvements of sdNoC are demonstrated by comparing it against the concept, which is the state of the art for solving message dependent deadlocks in on-chip networks.

- Chapter 4 covers the development of Verilog models realizing the sdNoC concept. A router and network adapter implementing all necessary features of my proposed regressive deadlock recovery concept are presented. With the help of these hardware models the chip area requirements and low level performance values of sdNoC are estimated.

- Chapter 5 presents the contributions of this thesis in the field of hierarchical NoC topologies. This includes the approach of employing different types of topologies

in the hierarchical NoCs and optimizing the access to shared resources, like I/Os or memories. The concepts are discussed and an evaluation is made based on simulation results.

- Chapter 6 concludes this thesis summarizing the scientific contributions and giving an outlook of possible future research directions.

The work presented in this dissertation is based on the RapidMPSoC project. This project was funded by the German Federal Ministry of Education and Research (Bundesministerium für Bildung und Forschung, BMBF) under grant reference 01M3085 within the Research Programme ICT 2020.

Chapter 2

State of the Art in Networks-on-Chip

Networks-on-Chip (NoCs) are distributed communication infrastructures, consisting of routers, that are interconnected by point-to-point links. The computation cores, memories and I/O interfaces of system are connected to the routers via network interfaces (also called network adapters). Usually, one core is connected to a router, but there are also network topologies or architectures in which multiple cores are connected to one router.

Another form of connecting cores to a NoC is to interconnect a couple of cores (generally two to four) locally by a bus or a crossbar. This computation island is then connected via a network adapter to the NoC. Such an architecture is referred to as tightly coupled tiles.

According to [7], NoCs can also be regarded from a hierarchy of layers, analogous to the OSI layer model of computer networks:

- The physical layer deals with the implementation of the physical links between switches. It can comprise advanced link implementation techniques such as low swing or multi-level signaling. The physical layer forms the basis for moving data words between two connected routers.
- On the data link layer reliable communication is provided. That means controlling of medium access, e.g. resolution of contention. Other tasks on this layer are

synchronization of the communication, flow control between two routers, and error detection or correction.

- Network layer describes the topology and the protocol (routing and forwarding scheme) used in the network, as well as the router architecture (e.g. buffering strategies, switch, ...).

- The transport layer mainly affects the network adapters. Basic tasks like segmentation and reassembly of packets are part of this layer. Where necessary, this layer can also comprise end-to-end services, such as flow control or bandwidth regulations between the source and the destination of a transfer.

Designing and adapting a NoC to a system and its application is a challenge, due to the huge design space. It is made up by a multitude of parameters such as topology, switching method, forwarding technique, routing algorithm, flow control, router architecture, buffering, etc. In the following, an overview of the most important techniques, methods and architectural parameters is given.

2.1 NoC Building Blocks

A NoC generally consists of three basic components [11], listed in the following:

- Routers (also called network nodes) build the network and are responsible for the correct transfer of the data to its destination. Routers realize the buffering of data, its routing, forwarding, flow control etc. A router implements network interfaces to enable its interconnection.

- Point-to-point links interconnect the routers and provide the bandwidth for the data transport. The number of links between routers and the kind of connectivity is determined by the topology of the network.

- A Network adapter connects a core or complete computation island to the network, i.e. to a router. Thus, a network adapter requires a core and a network interface.

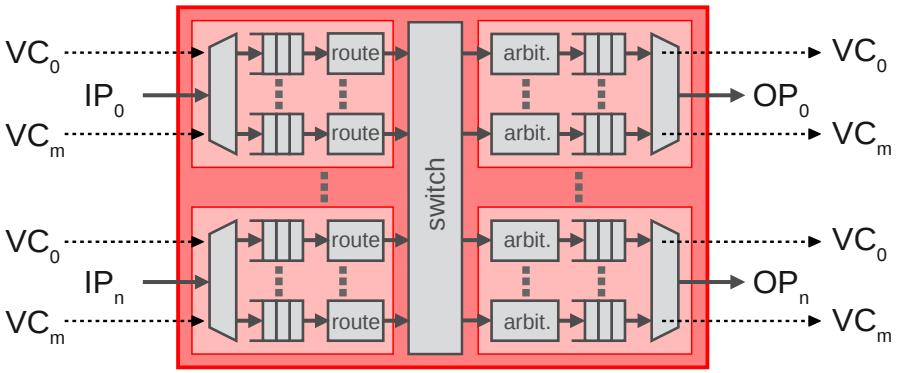

FIGURE 2.1: General router architecture with n ports and m virtual channels per port

2.1.1 Router Architecture

An example of a router architecture implementing buffered forwarding is shown in Figure 2.1. Basically, a router has multiple input ports at which it receives data that it forwards to its output ports, according to the implemented protocols. In the depicted router, the modules and functionality of an input port is combined in an input module (lighter area in figure). The same is done at the output side in the output module. The data path of the input module generally contains a buffer queue and a routing or decode unit that determines the output port for the incoming data. In case the router implements virtual channels (more details in next section), the input module requires a de-multiplexer that enables the distribution of the incoming data on parallel buffer queues. While the depicted router allocates a routing unit for each virtual channel, a shared routing unit would also be possible.

The next element of the router in the data path is the switching unit that interconnects all input and output modules. The switch unit transfers data from the input channels to the correct output channel (one of the buffer queues of the output module). The switch can for example be implemented by multiplexers or a crossbar.

The output module contains again one buffer queue and one arbitration unit per virtual channel. The arbitration unit solves contentions, in case two packets from different input channels want to be switched to the same output channel. In such a case, the arbitration unit determines which one will be first and which has to wait. Packets accepted by the arbitration unit are transfered over the switch unit and put into the output buffer queue. From there, the data is sent on the link, if the flow control information from

the downstream router allows this. If the router implements virtual channels, and thus has multiple parallel buffer queues in its output module, a multiplexer with arbitration functionality is required at the output that handles the access of the data from the different queues to the link.

The router shown in Figure 2.1 employs a basic, input- and output buffered architecture. In practice, a multitude of router architectures has been implemented or proposed, which are however quite similar to the basic router architecture described here or at least can be derived from it. For example, the output buffer can be omitted to save chip area. In case, the major design target is to achieve high clock rates, the output buffer would not be removed, instead even additional pipeline stages would be introduced. Figure 2.5 shows a router which has been extended by an additional pipeline stages between the routing and the switching unit. Thus, the routing and the arbitration functionality is decoupled, and critical paths are shortened. Intel's NoC router of the Teraflops processor [16] implements even 5 pipeline stages [15].

Virtual channels [33, 34] (VCs) allow packets to share the same link (also called physical channel) among multiple logical channels. For each virtual channel a buffer (input and/or output) is instantiated, from which packets are de-/ multiplexed from/to the link (see Figure 2.1). Since buffers are the major contributor to power and chip area requirements of a NoC router, virtual channels are very costly. These costs have to be weighed in against the advantages of virtual channels. The utilization of the network's links can be significantly increased by their introduction, especially if the network employs wormhole forwarding (see Section 2.2.2.3). Accordingly, virtual channels can increase the throughput of the network.

Virtual channels can either be freely used by the packets in the network in order to increase performance, to realize Quality of Service (QoS) or to handle the problem of deadlocks in a NoC. For QoS, packets can be split up into different priority classes, which are then statically assigned to specific virtual channels. Virtual channels serving higher priority packets can then be preferred in the arbitration and switching of the NoC router. More advanced QoS schemes can even provide guaranteed bandwidth for individual connections by reserving a complete logical channel, based on virtual channels, between source and sink.

Virtual channels are commonly implemented by simple, independent FIFO queues, which makes sharing of buffer space between the virtual channels impossible. Nevertheless, this implementation is preferred as it avoids complex memory management and thus enables higher router clocks [35].

2.1.2 Network Adapter

Network adapters are the connectors between computation cores of the tile and the network, i.e. they decouple computation from communication. Therefore, an adapter requires a core interface as well as a network interface. The basic task of the network adapter is to split messages into packets and packets into flits that are transferred over the network. This means the network adapter receives data on the core interface, packages it into the according data units for the network (generally flits), and then sends these data units over the network interface into the network. On the receiving side, the network adapter has to perform the according tasks to receive the data units from the network, reassemble them and hand it over to the connected core.

Many other features that can be realized in a network adapter are optional, such as end-to-end flow control. This control mechanism tries to prevent network contention around hotspot modules, which would otherwise also hinder packets not destined to the hotspot. That means, the bottleneck at a hotspot would then lead to a network-wide congestion problem. For example, fair hotspot access [36] is a concept solving this problem in the network adapters.

More advanced network adapters can also comprise DMA or message passing functionality [28]. Such functionality is mostly implemented together with a tile local memory, which the network adapter then works on autonomously. A network adapter capable of DMA just requires the specification of a transaction from the tile's processor. It then transfer the specified data autonomously, splitting it up in packets and finally in flits to send it into the network. Message passing is a protocol operating on top of DMA functionality. It manages complete transfers of data from one tile to another one, which comprises setting up data structures, reserving memory in the destination, etc. For example, message passing is implemented in Intel's SCC [28].

Quality of service (QoS) is also handled by the network adapter together with the routers of the NoC. Depending on the QoS classes, traffic is separated into best effort (BE) and guaranteed service (GS) traffic. For GS traffic the network guarantees a minimum bandwidth or a maximum latency or even both. Some applications simply require such guarantees from the used on-chip network, as these guarantees were provided by the replaced communication structures such as dedicated point-to-point links. For example, interrupts in a real time system with hard time constraints will require a guaranteed maximum latency, whereas streaming applications will rather need a guaranteed bandwidth [11].

The implementation of a network adapter's functionality can either be done in hardware or in software. Usually the hardware implementation is preferred in order to keep the network latency, here especially the network access latency as low as possible and to unload simple tasks from the processor core. For this purpose, network adapters or the network access can also be tightly integrated into the processing cores [37]. Depending on the clocking scheme that is used for the network and the computation resources, the network adapter also has to synchronize between the network and the connected core. In a system designed according to the GALS (globally asynchronous locally synchronous) principle, the network adapter has to synchronize the differently clocked islands of the system.

2.2 Communication Protocols

The communication protocols of a NoC determine how the data travels through the network. This comprises routing, forwarding and flow control. The routing function determines the path that packets take in the network from the sender to their receiver. Forwarding defines the data units and the strategy that defines how they are forwarded between the routers.

A basic categorization of communication protocols can be made depending on whether they are lossless or lossy. In computer networks lossy communication is pervasive. Due to this, protocols like TCP secure higher communication layers against packet or data loss. In contrast, on-chip communication is generally lossless, in order to simplify the overlying protocols or the complete protocol stack. While assuming lossless communication in the

on-chip interconnect is feasible up to now, future semi-conductor technologies will make this more and more costly. With shrinking feature sizes, information is represented by steadily decreasing charges, which makes the information more and more sensitive to different kinds of disturbances. To keep a strictly lossless approach, additional efforts will be required to detect and correct bit errors in the transferred flits but also in router logic.

Communication in a network can be further classified according to its switching technique. Two basic techniques exist: circuit switching and packet switching. In circuit switching a physical or logical path from the source to the destination is set up. Then the whole message is transferred through this path, which either behaves like a single or a pipelined wire. This method guarantees a defined bandwidth (QoS) once the path is set up (high latency), however this has to be paid with low link utilization, as all the links along this path are blocked until the circuit is closed again. The capabilities of circuit switching to share the links of a network are relatively bad. Such switching should be used if the messages to be transported are long and infrequent, i.e. the time required to transfer the message is long compared to the time for setting up the path. In case logical connections are set up, virtual channels together with TDMA can be used to increase the link utilization.

Due to these disadvantages of circuit switching, most NoCs employ packet switching. Hereby, links are only reserved and blocked for the time that is required to transfer the packet. Accordingly, packet switching is better at sharing the network's link bandwidth to all connected cores. In general, this results in better link utilization and higher throughput, as long as the communication streams between cores are not limited in numbers and very static.

2.2.1 Routing

The process of determining the path of the data in the network is called routing. Two basic categories of routing algorithms exist [19]:

- Oblivious routing does not consider the status or load of the network for determining the path of the data through the network. A sub-category of oblivious routing

is deterministic routing, where the path of data is already specified by the source and sink of the data.

- Adaptive routing considers the status of the network in the determination of the route, that the data will take. Routing schemes of this type enable dynamic load balancing and provide higher network throughput. However, at the price of additional chip area and power consumption.

Due to the additional complexity and costs of adaptive routing, most NoCs use deterministic routing. Most routing schemes are coupled to certain network topologies. Hereby, the most important aspect of most routing algorithms is to prevent routing cycles, i.e. deadlocks, in the network. A deadlock is a situation, in which multiple packets block each other from moving on in the network for unlimited time. Generally, preventing deadlocks is achieved by limiting the path diversity provided by the network's topology of the network to a set of paths that do not introduce routing cycles (see Section 2.4 for more details).

The most employed routing algorithm in 2D mesh networks is XY-routing. It is a sub-type of dimension ordered routing (DOR), in which packets first travel along rows and then follow a column to their destinations. It avoids deadlocks without additional measures such as virtual channels, and is very simple. XY-routing is a deterministic routing algorithm. Other often employed routing algorithms in mesh networks are turn model based algorithms [20] (west-first, west-last, ...) and odd-even routing [21, 22]. These algorithms avoid deadlocks and are not deterministic like XY-routing, but provide path diversity.

2.2.2 Forwarding Techniques

The process of transporting data through a network is called forwarding. Forwarding techniques do not only have an influence on the throughput or the latency of network, but also on the size of buffers required in the routers. Since the buffers mainly determine the chip area requirements of a router, the selection of the forwarding scheme has also a major effect on the chip area of the router [38].

2.2.2.1 Store and forward

Store and forward (SAF), for example used in the CLICHE NoC [9], is the simplest forwarding technique. It forwards packets step by step on a per hop basis. When a router receives a packet, it is stored in the router's buffer, and only after the packet has arrived completely, it is sent to the next router. This forwarding scheme requires relatively large buffers that are able to store at least a complete packet. This results in high area costs. The reception of the complete packet before forwarding it, also leads to a high latency.

2.2.2.2 Virtual Cut Through

Store and forward only forwards a packet to the next router after its complete reception. In contrast, virtual cut through [39] forwards a packet immediately after the necessary information for the routing decision, i.e. the packet header, has been received. In general the header will arrive in a router before the payload. As soon as the routing decision has been made and the output link is free, the already received parts of the packet are forwarded. If the desired channel is not available the packet is received and fully buffered in the current router. In full virtual cut through, the packet must then be buffered entirely before it can be forwarded, even if the output link becomes free earlier. Thus, a stalling packet does not block any links, however buffer space requirements are as high as in SAF, since a packet is only forwarded if the downstream can accept a complete packet. If there is no blocking, the packet is pipelined through the routers along the path determined by the routing function. In a network with high load, virtual cut through will behave like a store and forward.

Partial virtual cut through is almost the same as full virtual cut through, however if a packet is currently received and buffered, due to a currently blocked output channel, the instant this output channel becomes free, the forwarding of the packet starts immediately. Compared to full virtual cut through, this increases the network performance.

Virtual cut through decreases the latency in comparison to store and forward due to its forwarding in a pipelined fashion. However, it does not change the buffer requirements, since the basic unit of flow control (flit) is still the complete packet.

2.2.2.3 Wormhole Forwarding

In wormhole forwarding [13, 19] the packets are divided into flits, which is the smallest data unit on which flow control is performed on. This is the basic difference to virtual cut through. In general, the size of flit is chosen equal to the width of the physical link (phit). The first flit of a packet containing the header information is called header flit, the succeeding flits are data or payload flits and the last flit is called tail flit. A NoC router employing wormhole forwarding immediately forwards a received header flit after the routing decision, in case the according output link is free. The subsequent data flits are forwarded as they arrive, following the path opened by the header flit. This way the packet spans over several routers along its path through the network, like a worm. If the header flit is blocked the message aggregates in the routers' buffers along the path. As each router can only store a few flits for each port, the message spans over several routers even if it is not forwarded. Therefore, a stalling packet can block several other links along its path.

Only the header flit of a packet carries the routing information. The flits of a packet do not carry information specifying to which packet they belong. Thus, the network has to take care of keeping the flits of a packet together and not mixing them up with flits of other packets. Without additional measures, packets cannot be multiplexed over physical link. Wormhole forwarding provides low latencies and does not require buffering of complete packets in a node resulting in low area requirements. In general, the size of a packet or the length of a worm is variable in NoCs (Myrinet [40], MANGO [41]). However, a maximum size should be defined that is not exceeded, in order to limit the number of blocked links in case of temporary blocked packets.

Wormhole forwarding is the most popular forwarding technique in NoCs. Out of a list of 8 NoCs presented in [42], all use wormhole forwarding. The reason is that it successfully addresses the major design challenges of NoCs: Minimization of the transfer latencies (NoCs are a replacement for buses and crossbars), minimize chip area requirements and power consumption (Buffering is major power consumer of NoC).

2.2.3 Flow Control Techniques

Flow control controls the data transfer between the routers. Therefore, neighboring router exchange information concerning the status of the internal buffers, i.e. whether they are able to accept new flits or not. As described above, the size of a flit (flow control unit) depends on the forwarding scheme that is employed in the network. In wormhole forwarding a flit commonly equals a word on the link, in store and forward a flit is a complete packet. In general, flow control assures correct data transfer between the routers of the network, but it can also comprise optimal utilization of network resources. Thus it takes care of the allocation of virtual channels and buffers to packets.

In off-chip networks, links are often deeply pipelined. Thus, flow control mechanisms like handshaking would lead to long transfer delays. Therefore, dropping packets becomes an adequate alternative in case of full buffers. In contrast, in on-chip networks links are short and if pipelined at all, there are only few stages. Due to this, flow control techniques like handshaking are used rather than dropping flits or packets.

In the following the two mostly used flow control techniques are presented:

- The standard flow control method is based on handshaking. Hereby, a router starts to send a flit to the next router the instant the flit becomes available. If the receiving router has enough free buffer space to accept the whole flit, it signals this by an acknowledgement (ack) to the sending router.

  Depending on the length of the links (number of pipeline stages or buffers) over which the data is transferred between routers, the transitions on the link's wires are a major contributor to communication power consumption. Therefore sending packets that cannot be accepted by the receiving side can result in significant additional power consumption.

- In credit based flow control routers count the free space of their buffers and provide this information to the upstream router, if credit is available. In case of longer links, the sending router thus does not have to wait for the handshake signal on each transfer. In such a situation, the credit based flow control mechanism is more efficient than the handshake method.

While wormhole forwarding depends on flow control, there are also transport protocols, which do not require flow control. Usually, such protocols are either time slot based protocols or based on bufferless forwarding (see Section 2.5). An example is the pipelined time slot principle used for guaranteed service traffic in Æthereal [43].

2.3 Topologies of NoCs

The topology of a network determines how the routers of a network are interconnected. It has a major influence on the network's performance and cost. Topologies with high connectivity have a high number of ports per router and of interconnections between the routers. High connectivity results in higher aggregated link bandwidth (BW_{agg}, total bandwidth of all links in the network) and higher path diversity. While the aggregated bandwidth of a network does not translate directly into throughput, these networks tend to provide better performance in terms of throughput and latency. In contrast topologies of lower connectivity have less links available to transfer the injected traffic, thus they provide less performance. However, they also require less chip area.

Topologies can be regular [24, 44], such as mesh, ring, tree, etc, or irregular. Irregular topologies are mostly application specific networks [45–49], in which the routers are interconnected in a way that closely resembles the edges of the application's communication graph. Regular topologies can be further classified into flat and hierarchical topologies.

2.3.1 Flat Topologies

An extensive comparison of flat network topologies is given in [38]. One of the most popular topologies in NoCs is the 2D mesh [15][37]. This topology is employed in many multi-core processors: Intel's SCC [28, 50], Intel's Teraflops [15, 16], Tilera's Tile64 [37] etc. Nevertheless, there is a multitude of other flat topologies for NoCs: Ring, Spidergon [44, 51], SPIN [52], torus, butterfly, tree topologies, etc. Ring topologies are also very popular due to their low complexity and costs. Besides flat topologies, networks with topologies of higher dimensions have also been proposed [53]. These topologies lead to complex router architectures and even higher area costs than a 2D mesh network.

2.3.2 Hierarchical Topologies

Up to now, hierarchical topologies have mainly been proposed for parallel computer networks. [54] compares a 2D mesh network and hierarchical ring networks with up to 4 hierarchies for use in multiprocessor computers. Assuming locality of the applied traffic patterns, [54] shows that hierarchical rings can outperform 2D mesh networks in terms of latency for systems with up to 120 processors. Therefore, processors have to be distributed on many different sub-networks on the lowest hierarchy level. This means that many additional routers are required to connect all these sub-networks on the higher hierarchy levels, leading to many additional hops for packets traveling between different sub-networks.

[55] presents an analytical approach to investigate the optimized configuration of router buffer lengths of slot-based, hierarchical ring networks for parallel computer networks. The approach allows to determine packet latencies for uniform traffic patterns, but can only be applied to purely ring based hierarchical topologies.

[56] proposes hierarchical ring networks for on-chip multi-core systems. The publication presents an extensive exploration that considers system parameters like burst length, FIFO depth and relative clock frequencies of the rings. However, systems are only investigated up to a size of 16 cores.

Hierarchical multi-topology NoCs have also been published [57, 58]. [57] presents a hierarchical network using 2D meshes at the lowest hierarchy and rings on the two upper hierarchy levels. The forwarding technique used in the evaluated networks is store-and-forward, which is not very popular in NoCs, due to its high buffer requirements. [57] evaluates the proposed network architectures in terms of latency. In contrast, [58] presents hierarchical networks, that use different topologies on different hierarchy levels. For example bus and star topologies are used on lower levels, star, bus and mesh topologies interconnect the sub-networks on upper hierarchy levels.

2.4 Deadlocks in Networks-on-Chip

In packet-switched NoCs deadlocks can arise. Wormhole forwarding based networks are especially susceptible, since blocked packets can span over multiple routers and thus

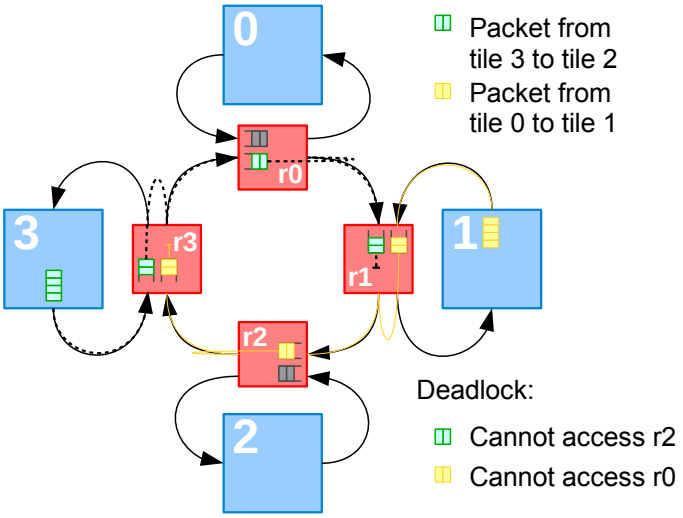

FIGURE 2.2: Deadlock in uni-directional ring network.

block multiple links and other packets. A deadlock is a cyclic chain of packets, that wait for each other to release their buffers. However, as the packets are forming a cycle this event will never occur (see Figure 2.2). The permanently blocked paths along this deadlock will rapidly lead to a completely filled up and blocked network, that can no longer transfer any packets. Two different types of deadlocks are differentiated in NoC based systems: routing dependent deadlocks and message dependent deadlocks (or protocol deadlocks). This differentiation will be explained in the following sections.

The precondition for a deadlock to occur at run-time is the presence of a cycle in the channel dependency diagram [59]. The channel dependency diagram (CDG) is a directed graph. It is the result of a function of the set of all channels of the network and the employed routing function. The nodes of the channel dependency diagram are the channels. The routing function determines the edges of the graph, assuming channel b can be reached from channel a according to the routing function, then node a is connected to node b by an edge in the CDG.

2.4.1 Routing Dependent Deadlocks

For the investigation of routing dependent deadlocks, only the network (the interconnected routers) is considered and it is assumed that packets can always leave the network

at their destination. In other words, the connected tiles can consume all packets coming from the network. Routing dependent deadlocks occur due to routing cycles in the network itself. This means, the packets forming a routing dependent deadlock are only buffered in queues of the routers, and not the network adapters or the tiles.

Approaches to handle these deadlocks can either be classified as 'Deadlock Avoidance' or 'Deadlock Recovery'. Deadlock avoidance completely prevents the occurrence of deadlocks at runtime, while deadlock recovery allows their occurrence, but provides means to revert the network into an operational state again. Deadlock recovery generally relies on low deadlock occurrence rates.

2.4.1.1 Deadlock Avoidance

The standard solution to routing dependent deadlocks in NoCs is deadlock avoidance. Deadlock avoidance is achieved by guaranteeing acyclic routing paths in the network [59], i.e. the channel dependency graph has to be converted into a completely cycle free graph. This can usually done by restricting the routing function, like it is done in mesh networks: XY-routing, turn model [20] or odd-even routing [21, 22]. XY-routing removes all routing cycles from a 2D mesh by prohibiting two turn out of each of the two basic cycles as shown in Figure 2.4. As the name says only turns from X- to Y-direction are allowed, turns from Y- to X-direction are prohibited. In other network topologies, such as rings, additional (virtual) channels have to be implemented and the routing function has to route in a spiral fashion in order to avoid the introduction of a cycle (see Figure 2.3).

2.4.1.2 Progressive Deadlock Recovery

While progressive deadlock recovery concepts have not been proposed for use in NoCs up to now, they have been proposed for parallel computer networks (e.g. DISHA [60]) . Progressive deadlock recovery resolves deadlocks by providing means to first detect a deadlock and then re-route the packets caught in the deadlock. Re-routing of packets can for example be done by providing an additional, reserved escape channel for such packets. Within this escape channel (also called deadlock channel), the occurrence of deadlocks has to be prevented. In DISHA this is achieved by allowing only exclusive

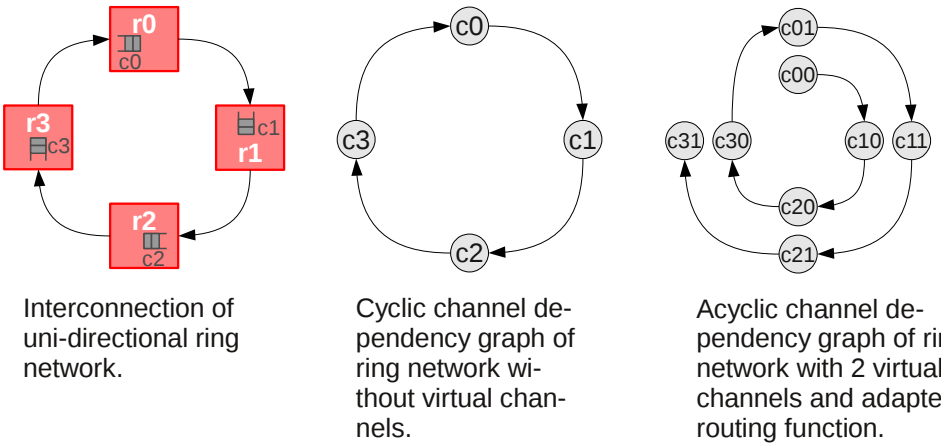

FIGURE 2.3: Uni-directional ring network (left), channel dependency diagram of a ring network without (middle) and with 2 virtual channels and adapted routing function (right).

access of one packet into the deadlock channel. This means, only one packet in the whole network is allowed to travel in the deadlock channel at a time. The authors of [60] realize the exclusiveness by a circulating token that has to be caught by a packet before it can enter the deadlock channel.

The precondition for progressive deadlock recovery is a low occurrence rate of deadlocks. Otherwise, the deadlock channel, which has a very low throughput due to the exclusive access, will get overloaded. In addition, the worst case performance (especially the latency) of the according NoC should not be relevant for the application, since in case of a deadlock the current message latencies will rise significantly.

The performance of a progressive deadlock recovery scheme also depends on the deadlock detection technique. In any case, a deadlock detection technique must satisfy the liveness property [61, 62], i.e. all deadlocks must be detected in finite time. The second property for classification is safety, i.e. only real deadlocks are detected, temporary blockages (also called pseudo deadlocks) are not detected.

Deadlock recovery schemes usually use heuristic deadlock detection. For example the deadlock detection mechanism in DISHA [60] is based on a simple timer in each buffer queue. The timer is started when a packet enters and is reseted when it leaves the queue. The occurrence of a deadlock is assumed, when the timer reaches a certain predefined threshold. As deadlocks are infinitely long blockages, timer based deadlock detection

mechanisms detect all deadlocks. This means they fulfill the liveness property. However, as they also detect deadlocks for only temporarily blocked packets (pseudo deadlocks), that actually are not part of a deadlock, they do not satisfy the safety property.

More advanced detection techniques exist, that either decrease the ratio of wrongly detected deadlocks or that speed up the detection of a deadlock [63, 64]. Fast detection is especially relevant for progressive deadlock recovery, as the bandwidth of the escape channel is usually low. Thus only few packets should be redirected and a deadlock has to be detected fast, in order to avoid that the network fills up with packets and becomes congested completely.

2.4.1.3 Regressive Deadlock Recovery

Regressive recovery resolves deadlocks by discarding one or more packets caught in a deadlock. In order to still provide lossless communication on the higher protocol layers, the network adapters require retransmission capabilities. The Proteo NoC proposed in [65] is the only concept in the field of NoCs to employ regressive deadlock recovery. Proteo does actually not use deadlock detection and back-pressure mechanisms, but drops packets in case of buffer overflow. [65] evaluates the concept in terms of latencies at low network loads. It is obvious, that Proteo NoCs cannot provide high throughput, as the drop rate is already high at moderate network load.

Another regressive concept, published for parallel computer networks, is Compressionless routing [66]. This concept is based on wormhole forwarding and proposes to drop packets in case of deadlocks. However, the deadlocks are not detected in the network itself, i.e. the routers, but in the network adapters. The consequence is, that the packets have to keep some kind of connection to the NA until the header flit of the packet has reached its destination, and the packet can no longer become part of a deadlock anymore. In Compressionless routing this connection to the NA is kept by adding padding flits to the packet after the last data flit until the header flit has entered the NA of the destination. In case a NA detects a deadlock, i.e. the packet does not move forward for a specified time period, it withdraws the packet from the network with a control signal. The drawback of this concept is, that the required padding significantly reduces the effective throughput of the network.

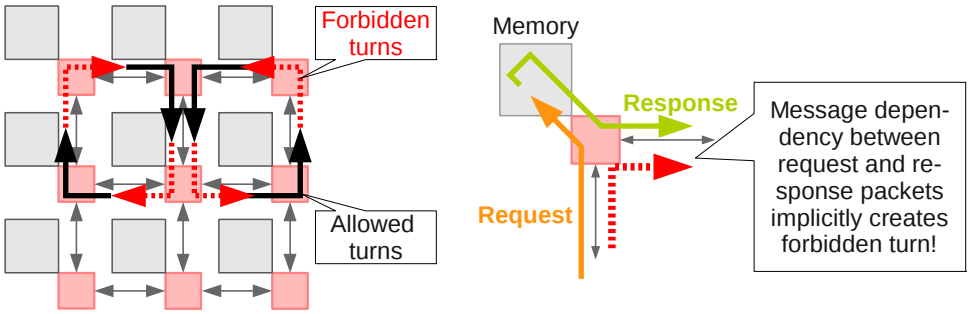

FIGURE 2.4: Message dependencies create forbidden turns in 2D mesh using XY-routing.

2.4.2 Message Dependent Deadlocks

Message dependent deadlocks arise from message dependencies [23][67], which are introduced by transport layer communication protocols. With these message dependencies the assumption we made in section 2.4.1 for the evaluation of routing dependent deadlocks, that cores can consume all arriving packets, is no longer true. The simplest example of such a transport layer protocol is a read access of a core to a memory. The protocol of this example consists of two dependent messages $m_0 > m_1$: a read request from the core to the memory and a read response from the memory back to the core. The message dependency between request and response causes the generation of the read response on the arrival of the request in the memory. In case the output buffer of the memory's NA is already filled up due to some temporary blockage in the network, the memory can no longer process the incoming read requests. If more requests arrive, they will even cause a congestion back into the network. In this way, slaves can implicitly introduce actually forbidden turns in a mesh network, as shown in Figure 2.4 on the right. On the left, Figure 2.4 depicts how two slaves can create all the turns required to form a routing cycle in a network using XY-routing. While this does not necessarily lead to an immediate deadlock in the system, the cycle is the precondition for a message dependent deadlock to occur at runtime.

In contrast to routing dependent deadlocks, the path of packets forming a message dependent deadlock contains channels of network adapters of the system. An isolated view of the NoCs channel dependency graph in order to achieve a deadlock free system is thus not sufficient. The whole system, i.e. the behavior of the cores as well as the architecture of the NoC has to be considered.

While standard SoCs mostly employ simple request- and response protocols with only two dependent messages, future SoCs could also adopt more complex protocols. An example are peer-to-peer streaming protocols in data flow based systems, such as audio/ video en/decoders [68] [69], which can contain even more dependent messages. Other complex protocols are protocols for cache coherency communication [70, 71] in multi-core systems, containing also request-request and response-response dependencies. In general, a communication protocol $m_0 > m_1 > ... > m_k$ with k dependent messages is called k-way protocol. In such a protocol all messages $m_0...m_{k-1}$, except the last m_k, trigger the creation of a new messages on their reception at a core. As the last message m_k does not trigger the sending of a new packet into the network, it is called terminating message.

2.4.2.1 Deadlock Avoidance

The standard solution to message dependent deadlocks is deadlock avoidance. Basically, the consumption assumption, we used in section 2.4.1 for the investigation of the routing dependent deadlocks, has to be restored. The first necessity is to separate different message types m_i into different buffers in the network adapters. In addition, one of the two following conditions has to be guaranteed [67]:

(1): Messages of a type m_i may never prevent the advance of messages of their subordinate type m_{i+1} in the network.

(2): Guarantee that every packet injected into the network can be consumed at its destination.

Considering the channel dependency diagram, again both conditions break the routing cycles in the system. The first of the two alternate conditions breaks the routing cycles in the network, whereas the second removes the effect of the message dependency in the tile, i.e. it removes the dependency edge between the network adapter's input and output channel.

One of the most popular methods to approach message dependent deadlocks in NoCs is strict ordering. It fulfills condition (1) by separating the messages m_i of different type into logically separate networks. One way to realize the logical separation is to

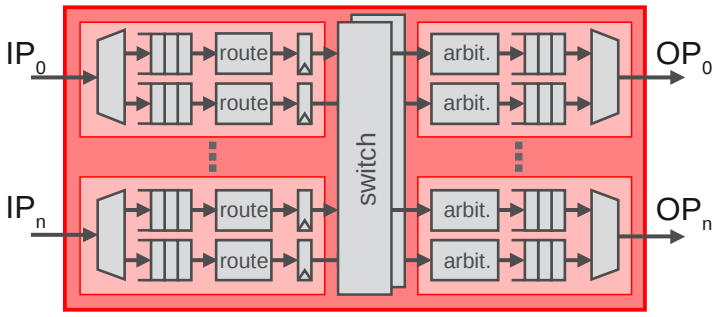

FIGURE 2.5: Architecture of router implementing strict ordering with 2 virtual channels.

implement physically different networks for each message type. This is for example done in Arteris' NoC [72] (separate request and response networks) and in STBus [73] (initiator-target and target-initiator networks). The alternative way to implement strict ordering is to introduce virtual channels in the network, that are statically assigned to the different message types [19]. This method is employed by Intel in their Single Chip Cloud Computer [28]. [74] present a comparison between virtual channels and multiple physical networks. Their conclusion is that virtual channels can outperform multiple networks if the buffers of the virtual channels are large.

The drawback of strict ordering is, that each additional message type, leads to a major increase of required buffer space in the NoC. Although the total throughput of the interconnect is also increasing, with multiple networks or virtual channels, the increase of required chip area is tremendous. A network with two virtual channels (see Figure 2.5) to support 2-way protocols (request > response accesses) requires approximately double the chip area of a network without virtual channels.

Queue sharing [75–77] can help to reduce the buffer space requirements of strict ordering implemented by virtual channels. Hereby, a buffer is dynamically shared among virtual channels of a port or even among ports according to the current requirements of the traffic pattern. With this concept the total buffer space of a router can be reduced or the throughput of a router can be increased in terms of flits/cycle. However, the complex control logic required to administrate the sharing of the buffers partially undoes the advantages. First of all, it also requires chip area and even worse, it reduces the maximum achievable clock frequencies [35] and thus reduces the effective throughput of the network.

An alternative to the spatial separation of the message types done in strict ordering is the separation based on time division multiplexing, also called Virtual circuit. The virtual circuits or connections share the network resources, i.e. buffers and links on basis of time division multiplexing. Hereby each connection is its own logically independent network. This strategy is often employed in NoCs that also provide guaranteed throughput, like Æthereal [78] or Nostrum [79]). Assuming dynamic creation and removal of connections, these concepts exhibit high latencies, as connections have to be set up before data transfers can be executed. Thus, this method is only applicable to static traffic patterns with only one or few destinations per sender that change only rarely. In addition, Æthereal also provides an implementation, in which the connections are statically implemented in the network at design time. Obviously, such a concept offers very low flexibility.

The alternative to separating messages in the network, are approaches that fulfill condition (2) of the two alternate conditions mentioned above: Buffer sizing or end-to-end flow control. Both of these two mechanisms guarantee that every packet injected into the network can be received by the destination and thus be removed from the network. Buffer sizing achieves this by sufficiently large receive buffers in the network adapters. While this is a popular method in parallel computers [23], where buffer space is cheap, it is up to the author's knowledge not applied in any NoC architecture. The receive buffers have to be large enough to drain all messages from the network that could possibly be sent to this destination.

$$Receive\ buffer\ size[bits] = \#\ of\ senders * \frac{messages}{sender} * \frac{\#\ of\ bits}{message}$$

The requirements of the input buffer space can be somewhat relaxed by limiting the sending quota of senders, without any restriction infinitely large buffers could be required. Due to this restriction, the border between buffer sizing and end-to-end flow control is blurred. End-to-end flow control achieves the guarantee that all packets injected into the network can be removed again by the destination tile, by pre-reservation of buffer space in the receive buffer. It is for example applied in the FAUST NoC [80], Tilera's Tile Interconnect [37] and in variants of Aethereal [43]. A popular way to implement the reservation scheme is based on credits, which represent receive buffer space, and have to be distributed among the senders. This either leads to additional traffic in the network, or requires an additional dedicated network. In addition, the pre-reservation either increases transfer latencies or requires multiple credits per sender in order to hide

the round-trip delay. However, each additional credit equals additional receive buffer space.

2.4.2.2 Deadlock Recovery

Up to now, neither progressive nor regressive deadlock recovery has been proposed to solve message-dependent deadlocks in NoCs. However, for parallel computer networks mDISHA [23] has been proposed, which is able to resolve message dependent deadlocks. It is based on DISHA [60] and has been extended to recover a network from message dependent deadlocks. As mDISHA has only one deadlock channel like DISHA and has to prevent the occurrence of message dependent deadlocks in the deadlock channel, all subsequent dependent messages $(m_{i+1}, m_{i+2}, ...)$ of a redirected packet (m_i) also have to be transferred via the deadlock channel. The deadlock channel is then blocked until the last message of the according protocol has reached its destination via this channel.

With this concept mDISHA relies on existing preemption capabilities of all cores or on the possibility to equip the cores with preemption. The network interfaces of mDISHA were equipped with functionality to trigger the preemption capabilities of the connected core for a packet m_i that arrives via the deadlock channel. The message m_{i+1} resulting from the processing of message m_i is then put into an additional output buffer of the network interface, which is reserved for such packets. From there it is sent via the deadlock channel of the network to the next destination. Compared to DISHA, this blocks the deadlock channel even longer and makes the occupation time dependent on the processing time of the correspondent cores. In addition, this creates problems with segmentation of messages into packets, in case a complete message has to be available to be processed by the core.

The authors propose mDISHA for use in networks with up to 16 virtual channels in the standard data path of the network. Their target is to efficiently use this high number of virtual channels, as in contrast to strict ordering where message types are statically assigned to virtual channels and the use of the different channels relies on the current traffic situation.

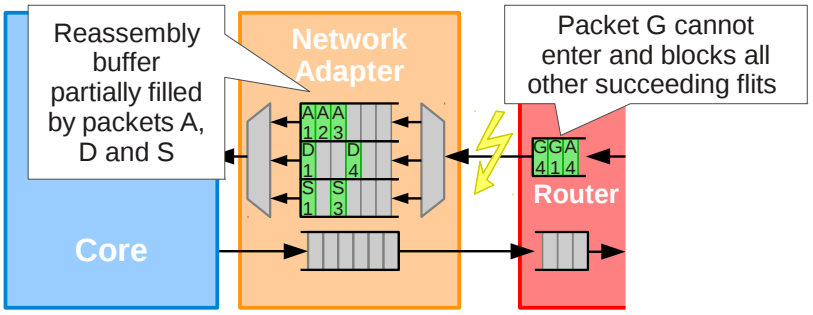

FIGURE 2.6: Packet G causes reassembly deadlock, as the reassembly queues of the reassembly buffer are already occupied by packets A, D and S.

2.5 Bufferless Networks-on-Chip

Bufferless NoCs [81–84] were not specifically targeted at solving routing or message dependent deadlocks, nevertheless they do not suffer from these types of deadlocks. The primary intention of bufferless NoCs is to reduce the buffer requirements, in order to decrease the chip area requirements and power consumption of NoCs. Bufferless NoCs are implemented without FIFO queues in the routers to buffer flits or packets in case of contention. Instead, the routers use only registers to implement multiple pipeline stages for routing or output port determination and arbitration. Thus, flits always have to move on to the next stage or router in the next clock cycle. Consequently, there is no need for a back-pressure signal to control the forwarding of flits, which eases arbitration and switching in the router. It is also the reason why bufferless NoCs are inherently free of routing or message dependent deadlocks. Flits that form a circular path in the network can still move on without being hindered by the back-pressure signal of a downstream buffer. However, in case of contention a router of a bufferless NoC has only the following two options for a packet or flit that lost the arbitration:

- Mis-route flits or packets, i.e. move them away from their destination in this router. If wormhole forwarding is used, the network must be able to dissolve the packet compound of flits in order to avoid deadlocks or the discard of packets. After the dissolution of a packet in the network, the flits have to be routed individually to their destination. Mis-routing also introduces the possibility of live-locks, which have to be prevented by corresponding means, such as an age field of flits and a

prioritization of older flits in the arbitration. The dissolution of the packet compound of flits leads to the requirement of reassembling the packets in reassembly buffers at the destination. This does not only increase the buffer requirements of the NoC, but also introduces the problem of reassembly deadlocks [84] (see Figure 2.6).

A reassembly deadlock occurs if all l queues of a reassembly buffer, which is able to reassemble l packets, are occupied by fragments of l different packets (packets A, D and S in Figure 2.6) and a flit of a $l + 1^{th}$ packet (packet G in 2.6) arrives. To avoid reassembly deadlocks the size of the reassembly buffer has to be chosen according to the number of messages that can arrive at a tile. However, this results in huge reassembly buffer sizes.

- Discard the packet or flit and wait for or request its retransmission from the network adapter.

The bufferless NoC concept BLESS [83] is purely based on mis-routing packets that have lost their arbitration. To completely avoid dropping or deadlocks, all flits carry header information in order to allow the dissolution of the packet compound and individually mis-route flits in the network. The authors of [83] did actually not consider the problem of reassembly deadlocks, they simply assumed infinitely large reassembly buffers in their model. Thus, this kind of problem could not arise in their model.

Other bufferless NoC concepts like BPS [81] or the one proposed in [84] use both, mis-routing and packet discard. Discarding packets solves the problem of requiring huge re-assembly buffers if mis-routing would be allowed only. Mis-routing packets decreases the drop rate, which would be very high if every contention would lead to a dropped packet or flit. The BPS network [81] additionally tries to reduce the drop rate by over-provisioning the network using Space Division Multiplexing (SDM). This means, the data width of all data paths, i.e. links, is multiple flit widths wide, and thus allows to transfer multiple flits in parallel. When packets have to be dropped nevertheless, a router of a BPS network requests the retransmission by sending a NACK packet to the original sender. However, since this NACK packet must not be dropped they are transfered in buffered channels. Scarab a bufferless NoC published in [82] removes the buffers for NACKs by providing an additional circuit switched network reserved for the NACK messages.

While bufferless networks do not have the problem of routing or message dependent deadlocks, they also suffer from several drawbacks. In addition to re-assembly deadlocks or livelocks, they are also relatively inefficient [19]. Bufferless networks waste a significant amount of link bandwidth either by misrouting or dropping of packets or flits. Although, routers of bufferless NoCs can run at higher clock frequencies (up to double clock compared to buffered NoCs [81], their peak throughput is still lower. According to [84] bufferless networks are preferably used for low or medium loaded networks. However, even if a network's load is only medium in average, if the traffic scenario contains hotspots (like memories or I/Os), the network load is comparable to a highly loaded network around the hotspot leading to decreased throughput due to high packet discard rates or misrouting.

It also has to be considered that the major advantage of bufferless NoCs, the savings of buffer space in the network is partially wasted by the reassembly or retransmission buffers, if they are explicitly instantiated in the network adapter. While the retransmission buffer as used in [81] and [82] could also be used to provide fault tolerance in terms of soft errors, the reassembly buffer in [83] and [84] cannot be used for this matter. To make matters worse, it has to be rather big to avoid reassembly deadlocks.

2.6 Fault Tolerance

Due to the constantly decreasing feature size of semi-conductor process technology, today's systems become more and more sensitive to all forms of manufacturing and environmental variations [29, 30, 85, 86]. This includes hard faults, timing problems caused by temperature changes, radiation induced soft errors, device aging and others. Mechanisms to cope with these problems will become essential for future CMOS systems, especially since todays NoC concepts principally assume lossless operation. Possible counter measures against bit flips in the transported flits are either protection by error correction code (ECC) or retransmission of faulty packets or flits after detecting errors via cyclic redundancy checks (CRC). Retransmission can either be realized on a hop-by-hop basis [87, 88] or end-to-end [66].

However, protecting the transported data on the links and in the router is not sufficient in NoCs that operate purely lossless. The routing and arbitration logic also has to be

protected against any type of errors. A wrong routing decision could lead a packet on a path from where it could no longer be able to reach its correct destination (due to XY-routing or the packet was already routed to a wrong local port). An error occurring in the switching or arbitration logic could lead to the destruction of the packet, i.e. the destruction of its sequence of flits in the network. Then the first few flits of a packet would arrive at the correct destination, while the rest of the packet would arrive somewhere else. Such situations could be prevented by protecting also the logic of the router. An example of an according method would be triple modular redundancy (TMR) [89].

Protecting data in NoCs against soft errors has already been addressed by the research community in the past. Most of these publications propose hop-by-hop based retransmission [87, 88]. The authors of [87] present a hardware implementation of such an approach and evaluate it in terms of latency and chip area requirements. [88] also presents a fault tolerant NoC based on hop-by-hop retransmission, which is evaluated with the help of a simulation model. In this concept the router queues are extended, and sent flits are only removed from the queue after an acknowledgement from the downstream router confirms the correct reception of the flit (CRC). However, in comparison to the concept of [87], it is only able to protect the flits against bit flips occurring on the links. If a bit of a flit mistakenly changes while being stored in the router, the bit flip cannot be corrected.

The problem of fault tolerance concepts that are based on hop-by-hop based retransmission is, that error detection has to be executed on each flit at each hop. [85] presents a comparison of an end-to-end retransmission scheme to hop-by-hop based retransmission. The clear advantage of hop-by-hop based retransmission is lower transfer latencies. However, this has to be paid by higher buffer space and energy requirements compared to end-to-end retransmission.

Apart from [88] and [66], all other fault tolerance concepts cannot use its allocated resources to approach the problem of deadlocks at the same time. [88] resolves deadlocks with hop-by-hop based retransmission concept. However, this concept is only applicable to routing dependent deadlocks. [88] does not address the problem of message dependent deadlocks and corresponding modifications that would be required in the network

adapter. [66] (already mentioned in section 2.4.1.3) also propose Fault-tolerant Compressionless Routing. While Compressionless Routing already adds padding flits after the last data flit to keep the route from the source reserved until the header flit has reached the destination, fault-tolerant Compressionless Routing adds padding flits until the last data flit has entered the destination. The additional padding is added in order to be able to trigger the retransmission of the packet by the sender, if a bit error is detected in the destination. Depending on the network distance between source and destination, this can become a quite high number of padding flits and also keeps routes in the network blocked unnecessarily long. According to the authors of [66] this overhead decreases the throughput even further (approximately 30%).

2.7 Conclusions

This chapter presented the basic knowledge on NoCs in the fields necessary to understand the contributions of this dissertation: the building blocks of NoCs, router architectures, protocols, the problem of deadlocks and their solution, etc. Focus was laid on the problem of message dependent deadlocks and of bit errors in NoCs of future semiconductor technologies. A broad range of approaches that solve these problems were described. This encompassed different techniques that mostly avoid deadlocks as well as techniques to achieve fault tolerance. Not only the functionality of these techniques was presented but also their significant costs in terms of network distance, communication latency or chip area requirements of the NoC.

This dissertation presents and evaluates hierarchical, multi-topology networks and adapts deadlock recovery for the use in NoCs with the intention of lowering communication costs. As this chapter showed, most of the research topics mentioned above are generally addressed separately from each other in publications. For example, the problem of message dependent deadlocks is solved by completely different approaches than the problem of bit errors. Since all of these problems have to be addressed in a NoC, the costs of all corresponding measures accumulate. The author of this thesis sees especially great potential in a combined approach that handles message dependent deadlocks and enables fault tolerance by using the same resources.

Chapter 3

Deadlock Recovery in Networks-on-Chip

Up to now, deadlock recovery, whether progressive or regressive, has not been proposed to solve the problem of message dependent deadlocks in Networks-on-Chip. While deadlock recovery has been applied in parallel computer networks, the concepts of choice for NoCs all belong into the category of deadlock avoidance. However, all these approaches lead to significant costs (see Section 2) to avoid a phenomenon that actually occurs very rarely in most traffic scenarios (see Section 3.1.2.1). For example, strict ordering for two dependent messages approximately doubles the chip area required for the network, independent of whether it is implemented by virtual channels or by multiple physical networks. End-to-end flow control realized by credits results in additional traffic and limits the sender quota. Buffer sizing leads to huge input buffers at the receiving tiles. Despite its high area costs the most popular approach in NoCs is strict ordering.

This thesis proposes to use deadlock avoidance only for routing dependent deadlocks in wormhole based networks. For this category of deadlocks, avoidance can be gained with very little effort for most topologies. However, for message dependent deadlocks that can occur despite avoidance of routing dependent deadlocks, this dissertation suggests to use deadlock recovery. Progressive and regressive deadlock recovery both allow to handle this problem with far less router complexity than routers implementing strict ordering. Significant amounts of router buffer space can thus be saved in both recovery concepts. While progressive deadlock recovery requires additional buffer space at the

receive side of the tiles, the regressive recovery concept needs additional buffer space at the sender side. Nevertheless, both concepts enable savings of buffer space for the total network (including the network adapters).

3.1 Progressive Deadlock Recovery in Networks-on-Chip

As mDISHA and DISHA were published for use in parallel computer systems and not in SoCs, the constraints of some parameters of these concepts are quite different than that of a concept used in a SoC. While in both environments high throughputs are desirable, NoCs cannot use an equal amount of buffer space as in networks on rack level. mDISHA is developed for networks with up to 16 virtual channels, in order to enable an efficient use of all virtual channels unlike strict ordering, in which message types are statically associated to certain channels. In NoC architectures such a high number of virtual channels is out of reach. As the chip area requirement of the NoC should be kept low in the proposed deadlock recovery concept, it should not rely on multiple virtual channels in the data path. Unlike mDISHA, the recovery concept presented in this dissertation keeps the order of packets between source and destination.

Another drawback of mDISHA is that it relies on the support of preemption of all connected cores. For some cores this might be easy to achieve, for hard cores bought from external suppliers this might be even impossible. Due to the larger buffer sizes that the DISHA concepts operate with, they can consider buffer fill levels for detecting deadlocks. In NoCs, buffer queues with 2 or 4 flits in size are often completely filled during run-time due to temporary blockages. Thus, this criterion cannot be applied for deadlock detection in on-chip networks. Due to these differences, deadlock recovery concepts from parallel computer networks cannot be simply transferred to the on-chip world, but require major adaptations.

3.1.1 Progressive Deadlock Recovery Concept for NoCs

The proposed deadlock recovery concept is especially targeted at enabling the network to handle message dependent deadlocks with additional chip area requirements kept to a minimum. While it would also be able to recover the NoC from routing dependent deadlocks, it was deliberately chosen not to do so. In this way, the limited bandwidth

of the escape channels of a progressive deadlock recovery concept can be reserved, and not wasted for routing dependent deadlocks that can easily be avoided by accordingly adapted routing algorithms. Therefore, the presented deadlock recovery concept is targeted at wormhole based NoCs using XY-routing.

The central differences of the deadlock recovery scheme for wormhole based NoCs [31] presented in this dissertation are its way of preventing the occurrence of deadlocks in the escape channels (also called deadlock channels in the following) and the ability to operate without multiple virtual channels in the standard data path of the network. In addition, the proposed concept does not rely on buffer queues with random access, but is able to operate with FIFO queues. This results in several major changes compared to the works of [23, 60].

The progressive deadlock recovery concept for NoCs presented in this thesis implements multiple nested virtual channels within the deadlock channel to avoid message dependent deadlocks in the escape channel. In other words, strict ordering is applied within the deadlock channel. While the deadlock detection mechanism and the access regulation to the deadlock channel are principally similar to mDISHA, they have been adapted to work with a standard NoC router and a shift register based FIFOs (instead of random access memory based FIFOs). Since this deadlock recovery concept should work without VCs in the normal network, the concept requires a back-off mechanism that prevents the tiles from further sending of packets into the network when a deadlock occurs. This allows the NoC to recover from a possible congestion after a deadlock occurred. The proposed recovery concept also provides functionality to keep the order of packets between sources and destinations.

3.1.1.1 Handling Message Dependent Deadlocks

The proposed deadlock recovery concept for NoCs solves the problem of message dependent deadlocks by applying strict ordering within the deadlock channel (DC). This means, multiple nested virtual deadlock channels are implemented in the escape or deadlock channel in order to separate the different message types $m_i|i = 1, ..., n$ from each other. The number of these virtual deadlock channels (VDCs) must be equal to the number of different message types existent within the system. Each VDC is associated to one message type m_i. Due to the strict ordering within the deadlock channel the

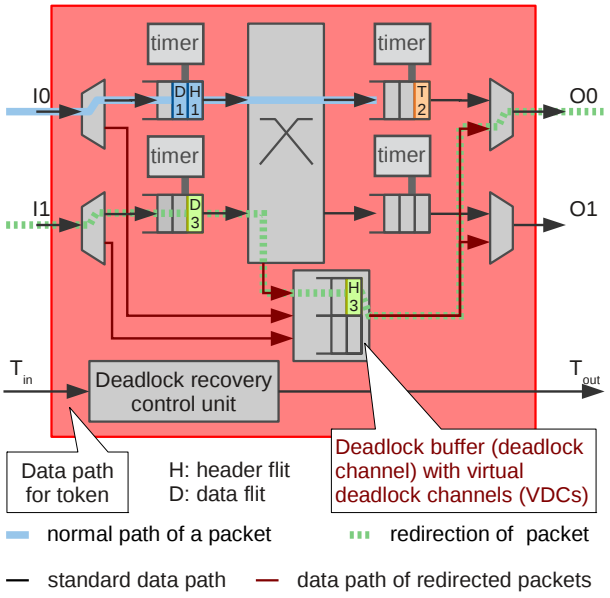

FIGURE 3.1: Router architecture for progressive deadlock recovery with strict ordering in escape channel.

absence of deadlocks is guaranteed, as strict ordering guarantees that the terminating message of a protocol is always able to reach its destination. There, it will be removed from the network and by definition will not trigger the creation of a new packets. As the arrival of the terminating messages (e.g. a response packet) is guaranteed, all subordinate messages (e.g. a request packet) are also able to arrive at their destinations subsequently.

Implementing multiple VDCs within the deadlock channel removes the requirement for preemption capabilities of the cores, which was already described in Section 2.4.2.2. This is because subsequent messages $m_{i+1}, m_{i+2}, ...$ of a redirected message m_i do not automatically travel in the deadlock channel too. Instead, they are put into the standard network interface output buffer like all other packets.

Figure 3.1 shows the deadlock buffer that realizes the deadlock channel (DC). This buffer is central to the router and not instantiated per input and/or output-port. Due to this, adding virtual channels to the deadlock channel is much cheaper than adding them to the buffers of the normal data path. The significance of this advantage rises with the complexity of the communication protocols used in a system. The more complex a communication protocol becomes, that means the higher the number of different, dependent message types becomes, the higher the chip area savings of the proposed

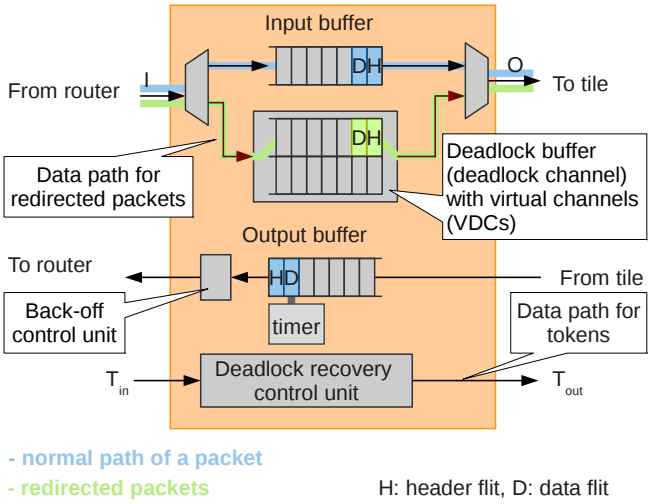

FIGURE 3.2: Network interface with deadlock buffer in the input path.

deadlock recovery concept compared to deadlock avoidance realized by strict ordering. A queue length of 2 flits for the deadlock channels as depicted in Figure 3.1 is absolutely sufficient, as this queue length already prevents a slow stop and go of flits due to the flow control mechanism between routers. Any further increase would not result in any throughput advantages since the deadlock channel does not suffer from contentions. This is due to the exclusive access to the virtual deadlock channels that is explained below.

Like the routers, the network interfaces also require a deadlock buffer in order to receive the redirected packets traveling on the deadlock channels. This deadlock buffer is connected in parallel to the normal receive buffer of the network interface (see Figure 3.2). It must be implemented with the same number of virtual deadlock channels as the routers. While preemption support is not required in the network adapter or the tiles, the network adapter has to support the redirection of packets from its output queue to the deadlock channel of the connected router, just like the routers.

Applying strict ordering within the DC is not sufficient to prevent the occurrence of deadlocks in the VDCs. In addition, exclusive access has to be assured. In contrast to mDISHA, the access must not be exclusive to the DC, but only exclusive to each VDC. The exclusiveness is realized by a token based access system. The heart of this system is a token distribution ring network (TDRN) that is connected to all routers and network interfaces as shown in Figure 3.3. Within the TDRN, tokens are circulating that need to be caught by a router or network adapter in order to get access to a VDC. The number

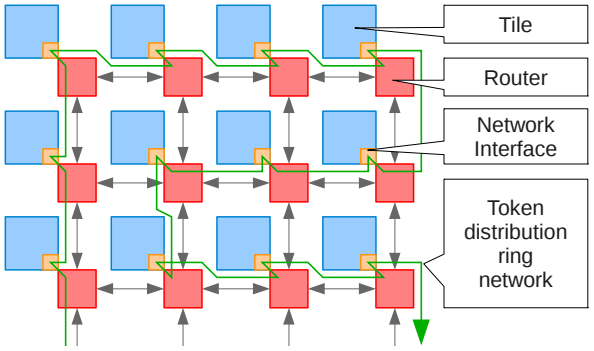

FIGURE 3.3: Token distribution ring network (TDRN) connecting all routers and network interfaces.

of tokens equals the number of VDCs. Each token is associated to one VDC. A router that wants to redirect a packet has to catch the access token for the VDC that fits to the message type of the packet that it wants to redirect. The token is then removed from the TDRN and is piggy-backed on the redirected packet. After the packet has been completely received by its destination, the token is again released into the TDRN. With the release of the token a command is sent via the TDRN to all tiles to stop sending of flits into the network for a predefined back-off time (see later).

Enabling exclusive access to the VDCs instead of the complete DC provides two significant advantages:

- Increased throughput of the deadlock channel. All available virtual deadlock channels can be used at the same time without resulting in the occurrence of a deadlock in the deadlock channel.
- The time a virtual deadlock channel (VDC) is occupied by a packet, that is redirected, is significantly reduced in comparison to mDISHA. In the deadlock recovery concept presented by this thesis a VDC is only occupied until the redirected packet has reached its destination. Since the DC is served with a higher priority in the link arbitration process of the routers than the normal data channels, this time depends only on the number of hops to the destination and the number of other packets in the deadlock channels taking the same links. In contrast to mDISHA, the progressive deadlock recovery for NoCs proposed by this dissertation releases the token for the VDC immediately after the packet has been received. In addition, the occupation time of the VDCs is independent of the processing time of the cores and the number of dependent messages.

- By minor modifications to the router architecture and redirecting additional packets it is possible to retain the packet order of packets with the same (source, destination)-pair without the requirement of additional reordering buffers. These measures are described in more detail in Section 3.1.1.5.

The additional wiring effort for realizing the virtual deadlock channels is very limited. As with standard virtual channel a router has to signal its downstream counterpart which channel is currently active. A NoC router implementing this deadlock recovery concept for a n-way protocol requires one standard data channel and n virtual deadlock channels to transfer flits. Accordingly, the link interface has to be able to signal transfers on $n+1$ different channels and the number of additional wires is thus $\lceil ld(n+1) \rceil$. First of all, this wiring overhead is very small in absolute numbers. Second, this wiring overhead is almost the same as the wiring overhead that standard deadlock avoidance with strict ordering requires. Strict ordering with n virtual channels needs $\lceil ld(n) \rceil$ additional wires on the link interface, compared to a NoC without virtual channels.

3.1.1.2 Deadlock Detection

The deadlock detection in the routers of the proposed progressive deadlock recovery concept is a heuristic approach based on timers. The occurrence of a deadlock is assumed if a predefined threshold is reached. Such a deadlock detection mechanism satisfies the liveness property (all deadlocks are detected) but not the safety property (only deadlocks are detected, no pseudo deadlocks). This is not a problem, since not supporting the safety property leads to an occupation of the deadlock channels by actually non-deadlocked packets, but does not lead the network into an non-operational state.

According to wormhole forwarding, the redirection of a packet always has to start from its header flit. Consequently, all stages in a router that could buffer a header flit have to be equipped with such a timer. In the router architecture shown in Figure 3.1 timers are added to all input and output buffers. Since packets in the network adapters are always part of a message dependent deadlock, the output queue of the network adapter also has to be able to redirect packets. Therefore, this queue also has to be equipped with the same deadlock detection capabilities. In contrast to mDISHA, buffer fill levels are not taken into account for the decision of whether a deadlock has occurred or not. In

wormhole based NoCs with small buffer queue sizes, this criterion would be true anyhow most of the time due to temporary blockages caused by the worm-like advancing packets.

A timer is started when a header flit enters a buffer and is reset when the header flit leaves. This means timers only run if a header flit is inside the associated buffer queue. If the timer reaches the predefined deadlock detection threshold, the deadlock recovery control unit (DRCU, see Figure 3.1) marks the header flit for redirection. However, the DRCU must not begin the redirection before it has not captured the according access token. In case the packet marked for redirection can proceed within the normal data channel (pseudo deadlock) before the required access token arrives, the redirection is simply aborted. The redirection of packets that are only part of a pseudo deadlock actually, poses no problems besides occupying the VDC for the time until they reach their destination.

Appropriate values for the deadlock detection threshold underlie a trade-off. On the one hand, a deadlock should be detected very fast in order to solve the deadlock as fast as possible and thus avoid a congested network that has to be emptied with the help of a back-off mechanism (see section 3.1.1.4). This calls for low detection thresholds. On the other hand, the heuristic, timer based deadlock detection mechanism should not detect too many pseudo deadlocks in order to not waste the limited bandwidth of the deadlock channel. This calls for higher deadlock detection thresholds. Due to this trade-off, appropriate thresholds have to be determined by simulation. In Section 3.1.2.3 different threshold values are simulated and evaluated.

3.1.1.3 Redirection of Packets

Enabling the redirection of packets from the standard data channels into the deadlock channel requires several changes to the standard architecture and behavior of a NoC router. Figure 3.1 shows some of the necessary architectural changes of the router. The deadlock buffer has to be connected to all input and output ports of the router with the help of de-multiplexers and multiplexers. This enables the transfer of packets within the deadlock channel to the destination and the redirection of packets from the output buffer to the deadlock channel. That is because the redirection of packets from the router's output buffer to the deadlock channel takes actually place on the link. The same holds true for a redirection from the output buffer of a network adapter. The router or network

adapter executing the redirection puts the flit from the normal buffer on the link and signals an active virtual deadlock channel at the same time. The receiving router will then store the flit into the corresponding VDC buffer queue.

In contrast, the redirection from the input buffers to deadlock channel requires an additional output port of the router's internal switch. This port is then connected to the deadlock buffer as shown in Figure 3.1.

Nevertheless, these architectural changes to the router are not sufficient to allow the redirection of packets in all possible situations. Especially for NoCs that are based on wormhole forwarding further measures need to be taken. In contrast to the field of parallel computer networks, the router buffers in the on-chip world are not implemented by random access memories but mostly by FIFOs based on shift registers. However, a FIFO allows only access to the first data word, or in this case a flit, of the queue. Still, a router has to be able to redirect any of its packets for a which deadlock was detected. In order to guarantee this, the header flit of a packet always has to stand on the first position in the queue. This can be achieved by changing the flow control, the arbitration and the switching behavior of the router.

The flow control functionality of the router is changed in such a way that it only allows the upstream router to send a new packet if its according input buffer is empty, i.e. after the last tail flit has left. Then, the newly arriving header flit will immediately be on the first position in the queue. The arbitration and switching functions are changed in the same way. A header flit in an input buffer is only transferred to an empty output buffer. This situation is depicted in Figure 3.1 in which 'Packet 1' is not allowed to enter the FIFO of output port 0, although there is free space available in this queue. 'Packet 1' may only be switched after the last flit of 'Packet 2' has left the output queue.

While these changes to flow control and arbitration function are necessary for the redirection of packets, they also decrease the effectively usable buffer size of the network. If the average size of packets in the network is small, there are more transitions from tail to header flits (T>H) and thus more buffer space is wasted in between these transitions. In case of larger packet sizes, less buffer space is wasted. The reduced effectively usable buffer space translates into a reduced throughput and also affects the latency of the network. The significance of this effect depends on the load of the network. At lower loads, when all packets are moving and almost no contentions occur, the effect

is insignificant and the network latency is only slightly increased. At higher loads the latency increases noticeably. See Figure 3.10 in Section 3.1.2.2 for a detailed latency and throughput evaluation.

3.1.1.4 Back-Off Mechanism

The deadlock detection mechanism used in the recovery concept presented in this thesis requires some time to detect a deadlock after it actually occurred. Since reasonable deadlock detection thresholds for the deadlock recovery prove to be well above 50 cycles (see 3.1.2.3 for a detailed evaluation of different timings), the time between occurrence and detection of a deadlock is not negligibly small. In fact, this time is long enough to get the network completely congested, originating from the deadlocked part of the network. This means, all buffers of the network will be completely filled up. Redirecting one or a few packets in this situation, while the tiles would immediately send new packets into the network would not get the network back into an operable state. Instead, the network would immediately block again because of further deadlocks. The network would then remain in a state in which its throughput would be approximately equal to the throughput of the deadlock channel.

To avoid this scenario, this thesis proposes a back-off mechanism that prevents the tiles from sending further packets into the network after a deadlock has been detected. This allows the network to be drained of most packets. The back-off command only prohibits the start of new transfers, i.e. sending packets of message type m_0 of an n-way protocol. Messages of higher message types $m_1, m_2, ...$ (e.g. response packets) may still be sent into the NoC in order to enable the consumption of possibly congested non-terminating packets (e.g. request packet) at their destinations (e.g. memories). The back-off command is distributed to all network interfaces via the token distribution ring network (TDRN) and prohibits the initiation of new transfers for the back-off time period. Since this back-off time is preset at design time, no values have to be transferred over the TDRN. Only the differentiation between access token and back-off command has to be made.

The link width of the TDRN is calculated according to Equation 3.1, assuming the NoC has n virtual deadlock channels and consists of N tiles.

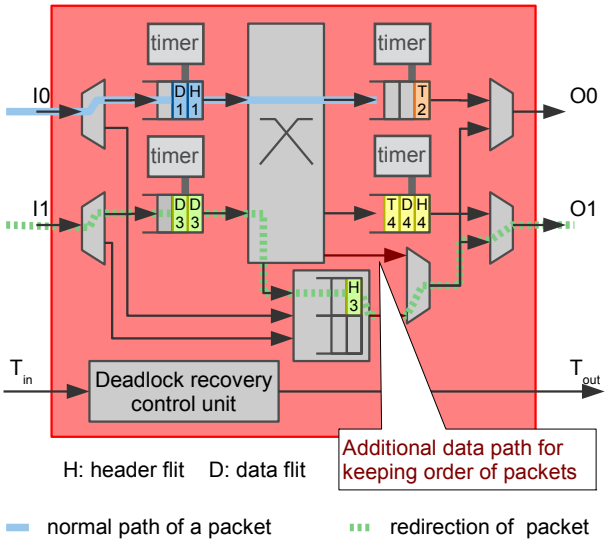

FIGURE 3.4: Additional data path for deadlock recovery router to keep order of packets.

$$w_{TDRN} = \lceil ld(n) \rceil + 2 + \lceil ld(N) \rceil \tag{3.1}$$

$\lceil ld(n) \rceil$ wires are for the discrimination of the access tokens for the different VDCs. One wire is required to signal the arrival of valid data and the other wire is necessary for the differentiation between back-off command and access token. In addition, $\lceil ld(N) \rceil$ wires are necessary to encode the id of the tile that issued the back-off command in order to be able to remove the back-off token after a full cycle in the TDRN. Since congestions are not possible in the TDRN a back-pressure signal is not required. For a NoC that handles communication protocols with up to two dependent messages, the width of the TDRN would then be $w_{TDRN} = 3$.

3.1.1.5 Retaining Packet-Order

The deadlock recovery scheme of the router architecture depicted in Figure 3.1 does not necessarily keep the order of packets sent by a source to one destination. Any redirection of a packet is capable of changing the order of packets with the same (source,destination)-pair if it is ahead of the redirected packet. Since the deadlock channel is serviced with a higher priority than the normal data channels, the redirected packet will overtake the one in the standard data path. This situation can easily be seen from Figure 3.4 which

shows how packet 3 is redirected. Assuming packet 4 in the output buffer at port 'O1' has the same destination as packet 3, then packet 3 would overtake packet 4 and arrive at the destination first.

The most obvious solution to restore the packet order would be the implementation of reordering buffers at the receive side of the network adapters. However, this contradicts the design target of reducing the required buffer space and hereby the required chip area of the communication infrastructure. Especially since the maximum required size of such reordering buffers could become very large in order to guarantee correct packet orders even in the worst case.

An alternative without additional buffer costs, is to simply keep the order of packets of a source with the same destinations in the network when doing redirection. This can be achieved by additionally redirecting packets that would otherwise be overtaken into the deadlock channel in front of the originally redirected packet. This is possible because the deadlock channel is realized like a virtual channel parallel to the normal data channels. Therefore, the redirected packets of a (source, destination)-pair take the same path (i.e. sequence of routers) to the destination as packets in the standard channels. Despite the actual exclusive access to the deadlock channels, this leads to multiple packets entering one deadlock virtual channel at the same time. Nevertheless, this cannot result in a deadlock, since all packets entering a virtual deadlock channel according to this mechanism are addressed to the same destination tile.

In case of packet 3 in Figure 3.4 which would overtake packet 4 in the output buffer of port 'O1', the order of packets can be kept if packet 4 is simply redirected into the deadlock channel while the header flit of packet 3 waits in the deadlock buffer. This requires that the header flit of packet 4 is still in the output buffer of the current router. For cases in which the header flit of the overtaken packet (as here for packet 4) is already in the input buffer of the downstream router, a small adaptation of the router architecture as shown in Figure 3.4 is necessary. The additional connection between the switch and the output ports that bypasses the deadlock channel buffers and standard data channel buffers allows to redirect the otherwise overtaken packet (here packet 4, redirection in downstream router not shown in Figure 3.4) and put it into the deadlock channel in front of the redirected packet (here packet 3).

The realization of this concept requires additional functionality that checks for every packet entering the deadlock buffer whether there is another header flit with equal source and destination in the buffers of the standard data path. If such a header flit is found, then the redirection of the according packet with bypassing of the deadlock buffer has to be triggered, while the originally redirected packet waits in the deadlock buffer. While this concept does not require additional flit buffers in the routers or network interfaces, it obviously increases the occupation time of the deadlock channels. The significance of this effect depends on the traffic characteristics, i.e. whether there are many packets on an according path with the same (source, destination). On the one hand this depends on how many of such packets a source is sending out in a sequence. On the other hand, it is affected by the network load, i.e. how many packets are interleaved into this sequence of sent packets by the routers.

3.1.1.6 Optimized Memory Controller Placement

As already stated before, a prerequisite for using deadlock recovery is that the occurrence rate of deadlocks is low, since the bandwidth of the deadlock channel for redirecting packets is relatively low. Due to this, the proposed progressive deadlock recovery scheme is used in combination with XY-routing. This prevents any routing dependent deadlocks that would otherwise also have to be handled by the deadlock recovery mechanism and hereby occupy the limited bandwidth of the deadlock channels. Although, it was chosen to handle message dependent deadlocks with recovery and not totally prevent them, the number of packets that have to be redirected has to be kept as low as possible. The rate of redirections can be influenced in two different ways:

- Occurrence rate of message dependent deadlocks. For each deadlock at least one redirection has to be executed in order to solve it. Thus, reducing the probability of deadlocks helps to improve the performance of the network.
- Redirect only deadlocked packets and not temporarily blocked packets. The share of detected pseudo deadlocks, i.e. packets that are just temporarily blocked but nevertheless detected as a deadlocked, can be influenced by the deadlock detection threshold of the timer to some degree. A too low threshold will lead to many redirections of packets that are actually not deadlocked.

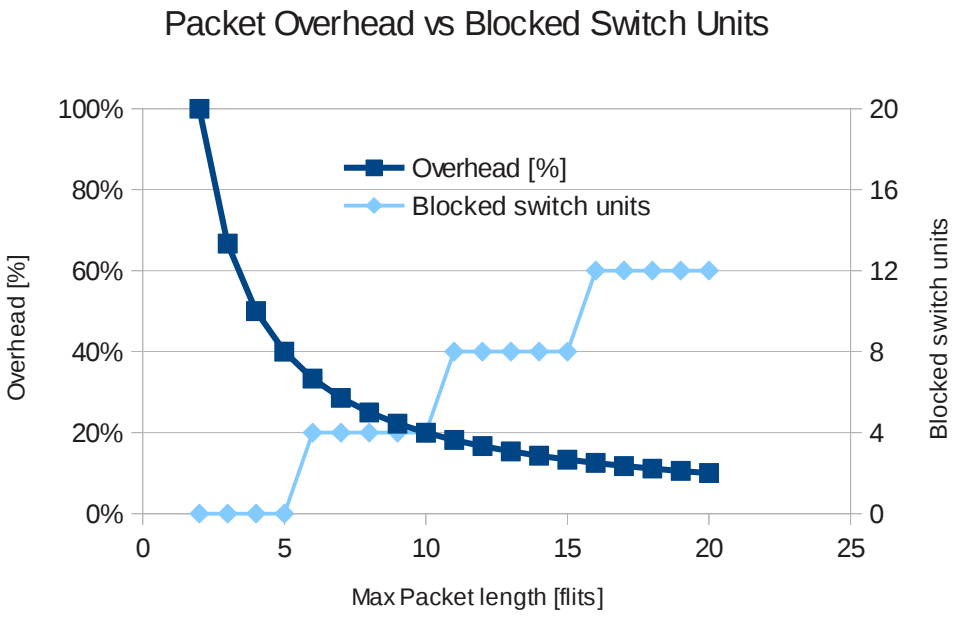

FIGURE 3.5: Effect of packet length on communication overhead and number of blocked switch units for a router with 3 flit long input and 2 flit long output queue.

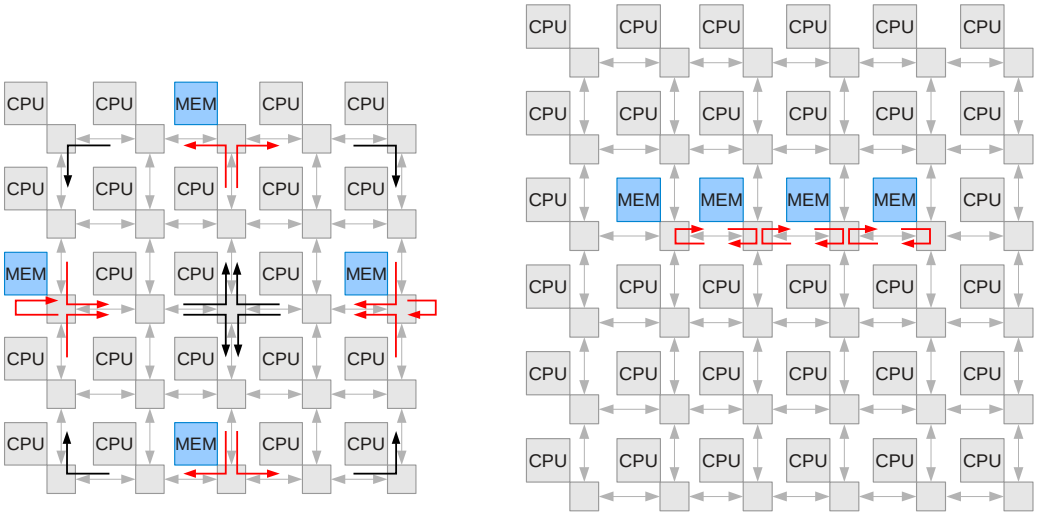

FIGURE 3.6: Mapping of memories at borders (left) and in a row in the center (right) of the NoC.

Since the second item of this list was already discussed in Section 3.1.1.2, the occurrence rate of message dependent deadlocks should be in the focus of this section. This rate depends on a few factors, which can be influenced by the system designer to some extent, but are also part of trade-offs:

1. The length of packets affects the occurrence of deadlocks. The longer a packet the more and the longer it blocks other packets temporarily along its route. If such a blockage occurs on a routing cycle in the network, the situation can result in a deadlock. Assuming a packet is moving in the network, then the number of other possibly blocked routes or packets is independent of the router queue length. In this case, the number of routes or switch units the packet can block depends on the pipeline stages inside the router. For example, the router shown in Figure 2.5 has 3 stages, 2 queues and one additional register. A router transferring a packet without any blockages will contain a maximum of 3 flits of this packet.

   In case of temporary blockages, the situation changes. The number of other packets that this packet may hinder in traveling depends on the packet length relative to the buffer queue length. The extreme case would be buffer queues that can store a complete packet. Then, a blocked packet would not necessarily block or span over switch units. However, this contradicts the central property of wormhole forwarding of not storing complete packets in a buffer in order to keep chip area requirements low. The alternative to increasing the buffer size is to decrease the maximum packet length allowed in the NoC. This reduces the number of switch units that a moving as well as a (temporarily) congested packet can block. However, this approach increases the communication overhead in the NoC. Figure 3.5 depicts this trade-off. The overhead of a packet drops with every flit added to the packet. For this graph an overhead of one flit, the header flit, is assumed. For the determination of the number of blocked switch units the following situation is assumed:

   The router can buffer 3 flits on the input side. The header flit of the packet is at the first position in the input queue, before the switch and is awaiting its arbitration. For example because a packet from another input port is currently traveling over the desired output port. Two additional flits can now be stored in the output queue of the upstream router without the packet blocking any other routes than

the one it is using itself. A packet with more than 5 flit length spans over the switch unit of the upstream router, and thus also blocks other packets traveling behind that would possibly be routed to free output ports.

2. The occurrence of deadlocks also depends on number of routing cycles in the network. The number of cycles generally depends on the number of tiles that implement message dependencies (e.g. memory tiles) and the mapping of these tiles. An extreme example would be a system with only one slave or memory in the network, since this slave can only create one forbidden turn in XY-routing no cycle can be created.

   With two tiles implementing message dependencies there is at maximum only one routing cycle in the network. With increasing number of memories or tiles introducing message dependencies the number of possible cycles rises. There are some special cases of mappings that create no routing cycles even with more than one memory. If the row of memory depicted in Figure 3.6 (right) is shifted completely to the left, the turn "East to West" which is required to form a cycle cannot be created. If there is only a very limited number of tiles in a network that create message dependencies and if all these can be placed in a row or column at the border of the network, this is the best solution in terms of message dependent deadlocks. In the given examples here, this approach is only possible because communication containing dependencies is abstracted and limited to the few memories. However, if processors also send responses or small acknowledgements on some requests from other processors, it is no longer possible to completely avoid routing cycles.

3. The rate of message dependent deadlocks depends on the number of packets that underlie message dependencies, that are at the input of an according tile and that can introduce the actually prohibited turn. A message dependency only becomes a problem if a message in the input buffer of a tile cannot be consumed by the tile because it would not be able to put the message it has to create according to the communication protocol into the output buffer. In addition, a packet must be at the output of the memory that tries to go in the direction of the forbidden turn. That means, a scenario with packets that create the prohibited turn must actually occur. If the rest of the routing cycle created by forbidden turns is then also congested a deadlock can occur. The less such packets there are at the input of such tiles the less the chance for a deadlock to occur.

The share of packets that can create an actually forbidden turn depends on the placement of the memories and the location of the accessing processors. This can be shown if the memory at the left border of the network shown in Figure 3.6 (left) and the forbidden turn "North to East" are considered.

This turn can be created by accesses from all processors below the center row that are not part of the first column. In the 5×5 mesh network in the figure, this are 7 processors. In contrast, in the 6×6 mesh network on the right of the same figure there is only one processor that is able to create the actually prohibited turn "East to West" at the left memory. Of course, in a network using XY-routing introducing one prohibited turn does not lead to deadlock, but the shown mappings also introduce the other actually prohibited turns.

4. Not only the mere number of cycles is important for the creation of a deadlock, but also the traffic load along the cycles. The more packets along a cycle the higher the probability for a deadlock to occur. This traffic must not necessarily consist only of dependent messages. A message dependent deadlock is not necessarily only formed by packets that are part of an n-way communication protocol. Instead, such a deadlock only has to contain a minimum of two dependent messages (e.g. read requests) at two tiles that are part of the cycle and implicitly create the forbidden turns. All other packets along the path of a cycle do not necessarily have to be part of a communication protocol with dependent messages. According to this, the probability for a message dependent deadlock does not only depend on the amount of packets with dependencies that are in the input buffer of an according tile, but also on the load of the network along the cycles introduced by the message dependencies.

(1) is more or less independent from the other factors and the mapping of the tiles. (1) can be relatively freely chosen by the NoC designer, as long as the applications requirements in terms of throughput and energy efficiency are still provided. In contrast (2), (3) and (4) are highly affected by the mapping of the tiles introducing the message dependencies.

The reduction of deadlocks is not the only design target in the mapping of tiles, instead one of the most important targets is to keep the network distance between different communication partners small. For certain types of tiles there are also physical placement

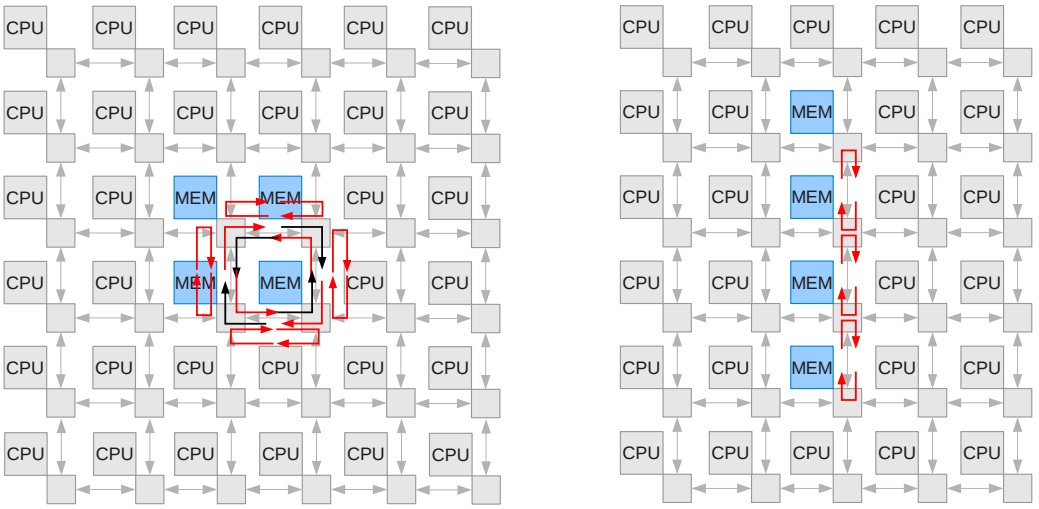

FIGURE 3.7: Mapping of memories in the center of the network in form of a square (left) or a column (right).

constraints, as for example I/O tiles, that have in some cases to be placed at the border of a chip or at specific places because of the connections to their pins.

Figures 3.6 and 3.7 show different mappings of 4 memory tiles in a network that exhibit the different parameters and cases considered above. While the number of possible cycles with 4 memory tiles is the same for all the different mappings shown, the network load around the memories and the communication distance to the CPUs varies a lot. While the mapping of the memories at the border of the network as shown in Figure 3.6 (left) reduces the link load of non-memory traffic around the memories, it also increases communication distance of the CPUs to the memories. This does not only increase the access latency but also requires more link bandwidth per access and thus decreases the effective throughput of the network. In addition, it enables a lot of processors to create the actually forbidden turns. According to these considerations, the mapping of the tiles in the center of the network, as shown in Figures 3.6 (right) and 3.7 (left) and (right), is significantly better. The problem is however, that in this way the memories draw additional traffic into the center of the network which generally already suffers from higher load in XY-routed networks. Section 3.1.2.9 shows how the different effects influence the deadlock probability and the throughput of the network.

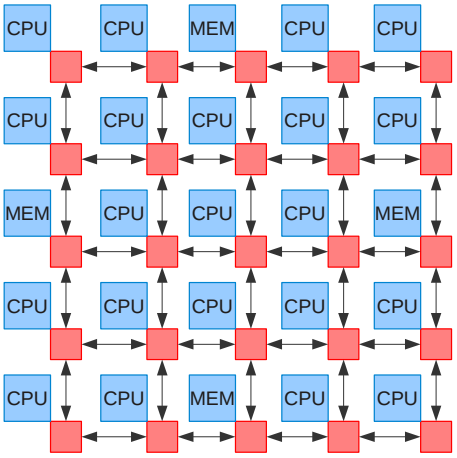

FIGURE 3.8: Mapping of processors and memories within the mesh topology.

3.1.2 Performance Evaluation

This section presents the performance evaluation of the proposed progressive deadlock recovery concept compared to strict ordering as a reference. The following investigations cover the parameters that were already mentioned in the presentation of the concept above. This includes the deadlock detection threshold, the back-off period, the packet length etc. However, not only architectural parameters of the concept are investigated, but also different traffic scenarios. The model used to simulate both competing concepts is based on OMNeT++ [90, 91].

3.1.2.1 Simulation Setup

Figure 3.8 shows the basic network architecture and the mapping of the processors (CPU) and memory (MEM) tiles. The NoCs evaluated in the following are $n \times n$ meshes employing wormhole forwarding and XY-routing. If not stated otherwise the networks under investigation are 8×8 tiles in size. The basic router architecture is input and output buffered as it was shown in the figures above (for example Figure 2.5 or Figure 3.1). The default queue length of the input buffers is 3 flits (modeling the included the additional register stage) and of the output buffers 2 flits. Other queue lengths are marked explicitly. The queues of the deadlock channels are also 2 flits deep. The length of these queues is not varied in the simulations as already explained above (see 3.1.1.1). Each input and output queue of the network adapters is 10 flits in size. This

equals the maximum packet size that will be existent in the network during the following evaluations.

To enable the evaluation of the proposed recovery technique in respect to message dependent deadlocks, the simulated traffic has to contain message dependencies. That means, at least a two-way protocol has to be employed in the simulated traffic. In the following simulations, such a protocol is used by the CPUs for the read accesses to the memories. Thus, the message dependency exists between the read request from the CPUs and the read responses of the memories. If not stated otherwise, the modeled SoCs consist of 4 memory tiles, while the rest of the tiles are CPU tiles. By default the memory tiles are mapped in the middle of the mesh network's borders as shown in Figure 3.6 (left). This mapping is comparable to Tilera's Tile64 [37] with the difference that it has 4 memory interfaces per network border. The following list describes the basic nomenclature employed for the networks investigated in the following:

- Pdr: Abbreviation for the on-chip network applying the **P**rogressive **D**eadlock **R**ecovery concept proposed above.
- So: NoC implementing **S**trict **o**rdering with 2 virtual channels.
- i*nom*: Denotes the queue length of the router queues in number of flits. n specifies the **i**nput queue length and m the **o**utput queue length. Here, n is mostly one flit larger than m as n also counts the additional flit register between the routing unit and the switch (see Section 2.1.1). Default values are $n = 3$ and $m = 2$.

For example So:i3o2 is a network using strict ordering with a 3 flit input queue and 2 flit output queue per virtual channel. With the exception of the simulations presented in section 3.1.2.2 (contains an evaluation of the adapted switching and flow control functions) a traffic scenario consisting of the following two classes of traffic is applied to the network architectures:

- Memory access traffic: Traffic based on the 2-way communication protocol, that is used by the CPU tiles for read accesses to the memories of the system. This traffic class provokes message dependent deadlocks in the NoC applying the progressive deadlock recovery concept. The read request packets are 3 flits in size, the read

response packets sent from the corresponding memory to the source of the request is 10 flits long. If not stated otherwise, the access of the memories by the CPU tiles is uniformly distributed. That means in the long term, the shares of accesses of a CPU to the different memories should be equal. Most evaluations in the following show graphs in which the request rate is iterated over a certain range. The read request flit generation rates are relatively low (0.003 to 0.023 per cycle per CPU tile) since 60 CPU tiles are accessing 4 memories, which in addition have to send 10 flits for each 3 flits they receive.

- Inter-processor traffic: This class of traffic serves as background traffic in the following evaluations (except section 3.1.2.2). All CPU tiles in the system send each other packets of 5 flit length. The target addresses of this class of traffic are uniformly distributed, i.e. each CPU sends the same share of packets to all of the other CPUs. The flit generation rate of this traffic class is fix (0.15 flits/cycle/tile), it is set to the half of the maximum throughput of a 8×8 mesh network without virtual channels for this type of traffic (approximately 0.30 flits/cycle/tile).

The throughput, latency and redirection rate graphs shown in the subsequent evaluations are generated from the simulation results of the modeled network architectures. Each point of the graph is the average of five simulations with the same traffic scenario but different seed values for the random number generators used in the traffic generators of the tiles. The measurement of a value in a simulation starts only after a minimum of 100,000 cycles allowing the simulation to settle and ends after a measuring period of a minimum of 250,000 cycles.

3.1.2.2 Influence on Performance from Adaptation of Flow Control and Arbitration Functionality

Independent of the occurrence rate of deadlocks and their handling by the deadlock recovery mechanism, the standard data path of the network applying the progressive deadlock recovery suffers from throughput reductions due to the changes of the flow control and switching function. As described in (3.1.1.3) these changes result in a reduced effectively usable buffer size. This first simulation will try to quantify this performance reduction and determine its dependence on the packet length in the absence of message

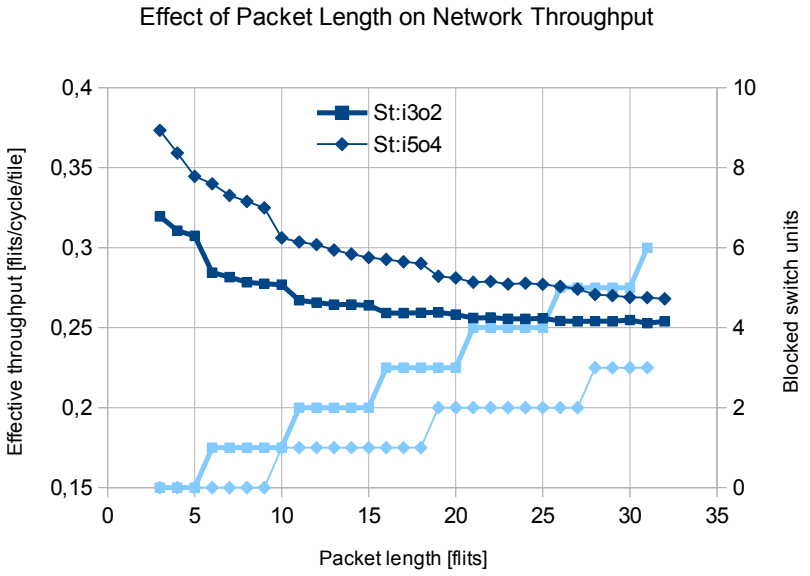

FIGURE 3.9: Influence of packet length on network throughput for different router queue lengths.

dependent deadlocks. This means, it will not perform a comparison between strict ordering and deadlock recovery. Instead, two networks without virtual channels are compared:

- St: **St**andard network with standard flow control and switching functions, that are also used in network applying strict ordering as described above.
- Pdr: A NoC applying **P**rogressive **D**eadlock **R**ecovery as proposed above. The flow control and switching functions of this network are adapted to the requirements of the progressive deadlock recovery concept. This means input and output queues of the routers may only be entered by a header flit if the queue is completely empty. One or more freely available flit spaces in a buffer queue are not sufficient, as long as the queue is not empty.

Only CPU tiles are connected to the NoCs, which send each other packets with a constant length. This packet length is iterated over multiple simulations in order to get the results for the graphs shown in the following. The target addresses of these packets are uniformly distributed and the inter packet times are generated according to the Poisson distribution. Since the deadlock detection and recovery mechanisms should not

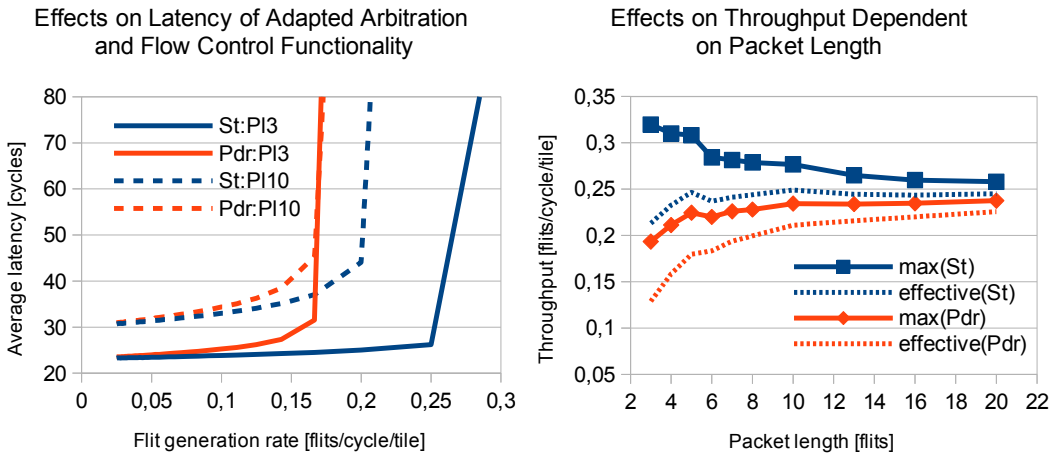

FIGURE 3.10: Influence of flow control and arbitration adaptation on network throughput and latency.

be evaluated in this investigation, the traffic does not contain any message dependencies and thus does not provoke any deadlocks.

The throughput of the standard network (St) with standard behavior of flow control and switching function suffers from the increased packet length. Figure 3.9 shows the throughput graphs for two standard networks with different router queue lengths. The standard network St:i3o2 has a $2+1$ flits input queue and 2 flits output queue, network St:i5o4 has corresponding queue lengths. Both networks show constantly decreasing network throughput not considering the packet overhead. However, major down steps in the throughput graphs exactly correlate the up steps of the graphs representing the number of blocked switch units by a temporarily blocked packet in an input buffer queue. The value of blocked switch units and its determination was introduced in Section 3.1.1.6.

Figure 3.10 shows the latency and network throughput for both networks, the standard network (St) and the one with switching and flow control functions adapted to progressive deadlock recovery (Pdr). Both networks are simulated with the router buffer configurations i3o2, i.e. 3 flit large input buffer and 2 flit output buffer. As expected, the figure shows a lower throughput for the deadlock recovery network (Pdr). From Figure 3.10 (right) it becomes obvious that this network (Pdr) is influenced by two effects:

- The effect described above for the standard networks (St:i3o2, St:i5o4): There is a noticeable down step of the throughput graphs max(St) as well as max(Pdr) when

increasing the packet length from 5 to 6 flits.

- However, the dominating effect for the throughput of the network with progressive deadlock recovery (Pdr) is the reduction of the gap between the effectively usable router queue lengths and the implemented router queue length with rising packet lengths. The figure confirms this, as the reduction of throughput compared to the the standard network (St) is significantly higher for smaller packet sizes than for larger ones. Note, that these results are only valid for traffic scenarios that do not create message dependent deadlocks. The dotted graphs (with the prefix effective) in Figure 3.10 (right) represent the throughput considering the packet overhead of one flit per packet. While the effective throughput for network applying progressive deadlock recovery (Pdr) is steadily going up, there are two local maximums for the standard network (St) for the simulated traffic scenario: For packets with length of 5 flits and 10 flits.

Figure 3.10 (left) shows the latency graphs of the two networks for two specific packet lengths. These different **p**acket **l**engths of 3 flits and 10 flits are denoted by Pl3 and Pl10 respectively in the diagram. The graphs also show that the throughput penalty of progressive deadlock recovery (Pdr) is larger for smaller packets than for longer packets. In addition, it shows that for low traffic loads, at which no or only few temporarily blockages occur, the latency of the networks is the same. The performance of the networks differs only at higher loads, at which the reduction of the effectively usable buffer space becomes significant.

3.1.2.3 Evaluation of Deadlock Detection Threshold

The following investigations explore different values of the deadlock detection threshold and an appropriate value is then chosen for the evaluations in the subsequent sections. The graphs of the network applying the proposed progressive deadlock recovery concept are denoted with Pdr:dxb150, with x representing the different values of the **d**eadlock detection threshold $x = \{50, 100, 150, 200\}$ in cycles. The postfix b150 denotes the **b**ack-off period, which is set to 150 cycles for the simulations of this section.

Figure 3.11 shows the memory throughput (left) and the inter processor traffic throughput (right) of the investigated networks. The memory throughput is the send rate of the

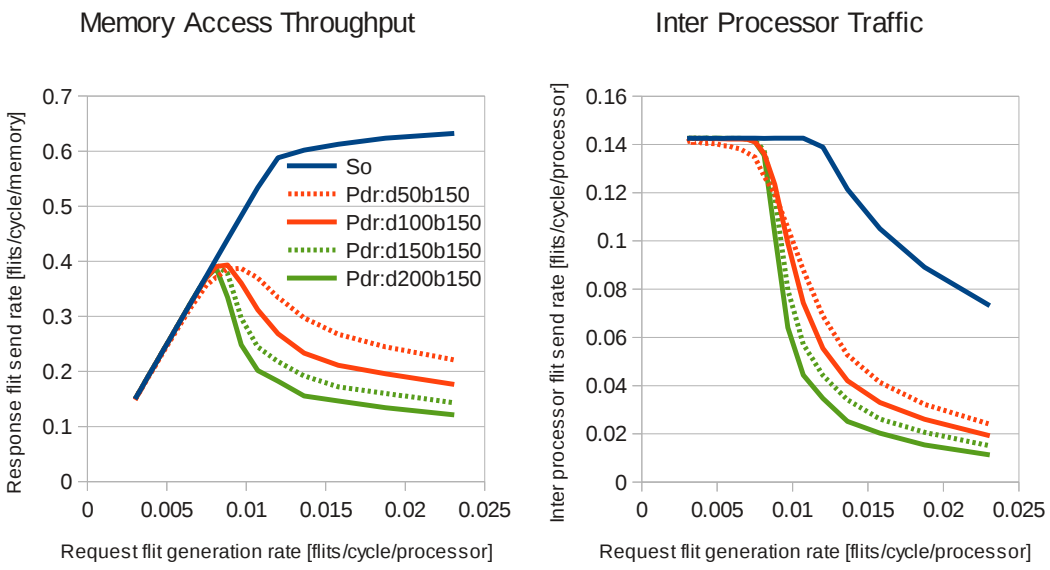

FIGURE 3.11: Memory access throughput (left) and average memory access latency (right) for different deadlock detection thresholds.

response flits per memory shown in dependence of the generation rate of read request flits in the left of the figure. On the right this figure shows the send rate of flits of the inter processor traffic class per CPU tile. While the generation rate of this traffic class is actually not changed it is from a certain read request rate on suppressed by the memory traffic to some extent.

Up to a memory throughput of 0.4 flits/cycle/memory the progressive deadlock recovery networks (Pdr) can keep up with the network based on strict ordering (So, see Figure 3.11 left). Up to this value the number of redirections is relatively low, for the simulated deadlock thresholds larger than 50 cycles almost zero, as can be seen in Figure 3.12 (right). However, with further increasing read request generation rates, the memory throughput as well as the inter processor traffic throughput drops significantly. The reason is, that the high load of the network results in an increasing number of deadlocks, that reduce the throughput of the NoC. In addition, the deadlocks lead to a rising number of redirections and thus activations of the back-off mechanism preventing the tiles from sending for a certain period of time (here 150 cycles). After the back-off period, the tiles try to send again into the network with relatively high rates, provoking new deadlocks, redirections and the activation of the back-off mechanism immediately.

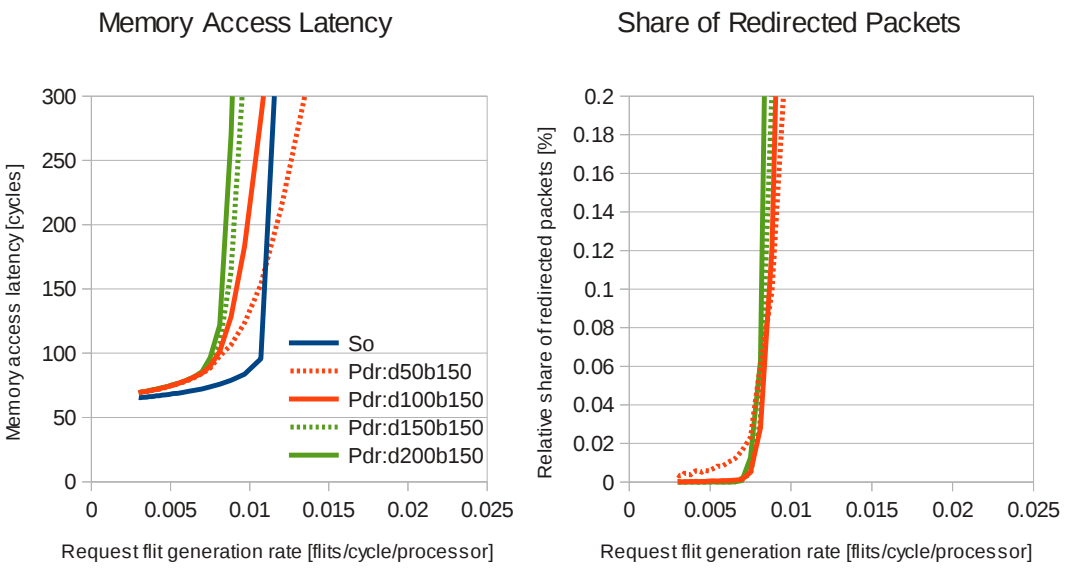

FIGURE 3.12: Inter processor communication throughput (left) and share of redirected packets (right) dependent on deadlock detection thresholds.

From both figures it can be deduced that lower deadlock detection thresholds are better at higher traffic loads. In these situations, networks with high deadlock detection thresholds of 150 or even 200 cycles (e.g. Pdr:d150b150 and Pdr:d200b150) simply require too much time to detect the deadlock. In the meantime, the network throughput goes almost down to zero and the network becomes completely filled up and congested. If the back-off timings are also increased to values such as 400 or 600 cycles the redirection rate can be reduced. But due to the long times to detect the deadlocks these networks cannot compete with the progressive deadlock recovery with 100 cycles deadlock detection threshold and 150 cycles back-off period (Pdr:d100b150).

While the throughput of all networks based on progressive deadlock recovery shows a major drop in throughput after their peak, the one with a deadlock detection threshold of 50 cycles (Pdr:d50b150) shows the best performance at high network loads. However, the problem with such a low deadlock detection threshold is that compared to the NoCs with thresholds $\geq$ 150 cycles a higher number of pseudo deadlocks is already detected at low traffic loads (see Figure 3.12 (right)). Due to this, the back-off mechanism is already activated at low traffic loads and thus the throughput of the network is decreased. The reduced throughput at low loads of the network with deadlock detection threshold of 50 cycles (Pdr:d50b150) can be seen in both throughput graphs of Figure 3.11.

The left graph of Figure 3.12 shows the average memory access latencies of the evaluated NoCs. The memory access latency starts when the read request packet is put into the output queue of the CPU tile's network adapter and ends when the corresponding response has been received by the CPU tile. At very low request flit generation rates the latency of all networks is quite similar. With increasing flit generation rates, the latencies of the networks with progressive deadlock recovery (Pdr) increase more than that of the network applying strict ordering (So) due to their lower throughput. While this is quite as expected, the lower latency of represented by the graph Pdr:d50b150 compared to graph So at request flit generation rates higher than 0.012 flits/cycle is surprising at first glance. This effect is caused by the back-off mechanism which leads to a less congested network at the theoretically higher generation rate, but at an actually much lower throughput of the network.

According to the evaluations made above a deadlock detection threshold of 100 cycles seems to be a good compromise. Nevertheless, the throughput of the networks based on the progressive deadlock recovery concept is significantly lower than that of the NoC using deadlock avoidance. One reason is the adapted flow control and switching function, required to enable the redirections. The second reason is that the network using strict ordering simply has almost double flit buffer space available in its routers. Organizing this buffer space in form of virtual channels helps to reduce temporarily blockages, increases the link utilization and thus the throughput of the network. This gain in throughput is for example up to 100% for virtual channels that can be used independent of message types [38]. While this gain would of course be lower for a network architecture in which the virtual channels are statically assigned to different message types, it is still existent and significant. In addition, it can be deduced that the networks based on progressive deadlock recovery should not be operated above their peak throughput permanently, as their network throughput will not saturate as networks implementing strict ordering do, but will even decrease.

3.1.2.4 Evaluation of Back-Off Period

After an appropriate value for the deadlock detection threshold has been found, the influence of the back-off period on the performance of the NoC should be evaluated. The router buffer configuration of all the simulated NoCs presented in this section is

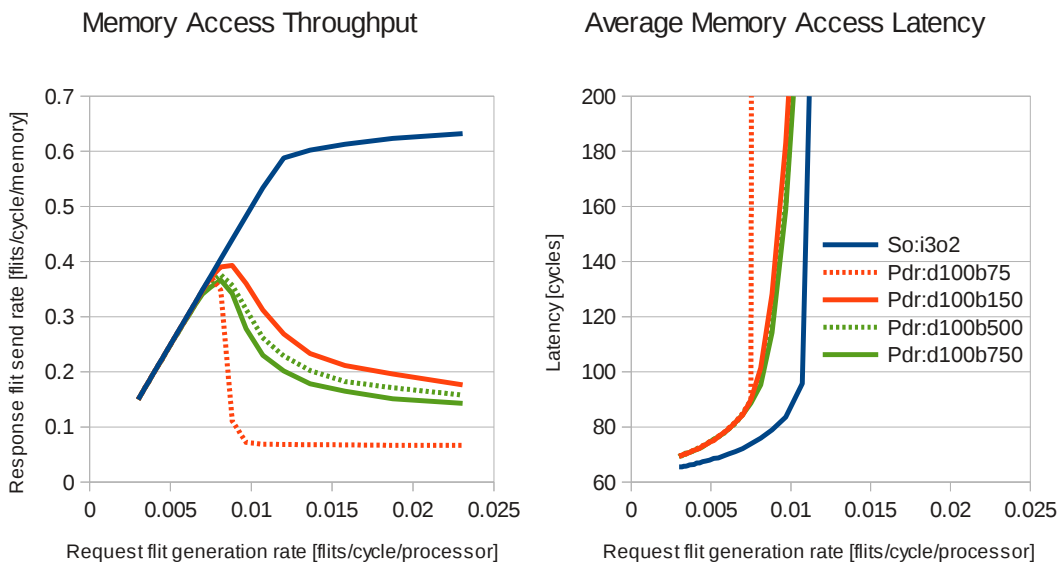

FIGURE 3.13: Memory access throughput (left) and inter processor request throughput (right) for different back-off periods.

i3o2, i.e. the router queues are 2 flits long. The deadlock detection threshold of the networks based on the progressive deadlock recovery concept is 100 cycles.

The back-off period plays a crucial role in the deadlock recovery process. Depending on how long it took to detect the deadlock and start its resolution, the network is congested to a certain degree. This means most of the router queues are completely filled up and the tiles already try to send additional flits into the network. Under these circumstances the network would not be able to recover from a deadlock, the network throughput would be more or less reduced to the throughput of the deadlock channel. The same behavior can also be seen in Figures 3.13 and 3.14 for the progressive deadlock recovery network with a back-off period of 75 cycles (Pdr:d100b75), as the back-off period is still too low. It is not sufficient to empty the network and return it into an operable state after a deadlock has occurred. The exploding rate of redirections of this network shown in Figure 3.14 on the right also proves that these 75 cycles are not sufficient to return the network into an operable state. Instead deadlocks or pseudo deadlocks occur immediately again, resulting in a significant drop in the throughput graphs of this network (Pdr:d100b75).

Increasing the back-off period to 150 cycles is already sufficient to clear the network of packets after a deadlock has occurred and to provide acceptable throughput values as the

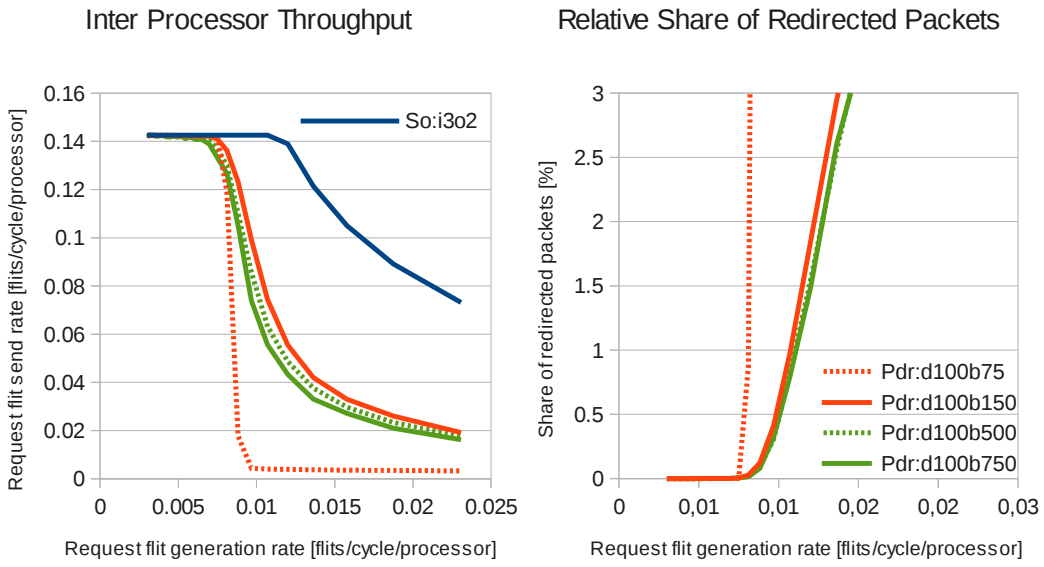

FIGURE 3.14: Average memory access latency (left) and share of redirected packets (right) for different back-off periods.

performance of the corresponding network shows (Pdr:d100b150). For back-off periods of up to 300 cycles (not shown in the figures) there is almost no change in performance compared to this network (Pdr:d100b150). A further increase of the back-off period beyond 300 cycles leads to a steady reduction of the network throughput, as shown by the throughput graphs Pdr:d100b500 and Pdr:d100b750. This behavior is expected, since the more the back-off period is increased over the time that is actually required to clear the network, the longer the tiles are prevented from sending packets into a network that is ready. Thus, the throughput decreases if the back-off time is increased further.

For the subsequent investigations a back-off period of 150 cycles is chosen for the modeled progressive deadlock recovery networks.

3.1.2.5 Localization of Memory Access Traffic

In the previous investigations, the memory access traffic was uniformly distributed, i.e. each CPU sent an equal share of read requests to every memory. This assumption is not very realistic, as in practice a system designer would try to store data or instructions required by a processor in the nearest memory. This is not only advantageous in terms of latency but also in terms of energy. In this investigation the effect of localizing the

memory accesses on the network throughput will be investigated. The inter-processor traffic is not localized but stays uniformly distributed. While the CPUs send as many read requests as before, the share of requests they send to memories that are nearer to them in terms of network distance is increased, while the share of packets sent to memories farther away is reduced. The degree of this localization is described by the localization factor. The share of requests S_m sent from a CPU c to memory m is calculated in dependence of the **loc**alization factor loc and the network distance $d(c, m)$ as described in Equation 3.3. The network distance $d(c, m)$ is measured in hops (routers) between the CPU tile and the memory.

$$s_m = \left(1 + \frac{loc}{d(c, m) + 1}\right)^2 \tag{3.2}$$

Normalizing s_m gives the share of requests S_m sent from a CPU to a memory $m = \{1, ..., M\}$:

$$S_m = \frac{s_m}{\sum_{i=1}^{M} s_i} \tag{3.3}$$

Three traffic scenarios with different localization values $loc = \{0, 10, 100\}$ are applied to the evaluated networks based on strict ordering (So) and progressive deadlock recovery (Pdr). The simulations are denominated with the postfix loc$x$ with x specifying the localization factor of the memory access traffic (e.g. So:loc10). A localization factor of 0 equals a uniform distribution of the memory accesses. The networks based on the progressive deadlock recovery concept simulated in this section use a deadlock detection threshold of 100 cycles and a back-off period of 150 cycles.

While the networks applying progressive deadlock recovery (Pdr:loc$x$) still provide a lower throughput than the ones based on strict ordering (So:loc$x$), progressive deadlock recovery profits from an increasing localization of the memory access traffic due to a reduced deadlock occurrence rate (see Figure 3.16 right). The memory throughput as well as the inter processor traffic throughput increase with rising localization as depicted by Figures 3.15 and 3.16. As expected, the memory access latency of these networks is also reduced, as the average network distance of the memory accesses is reduced with rising localization.

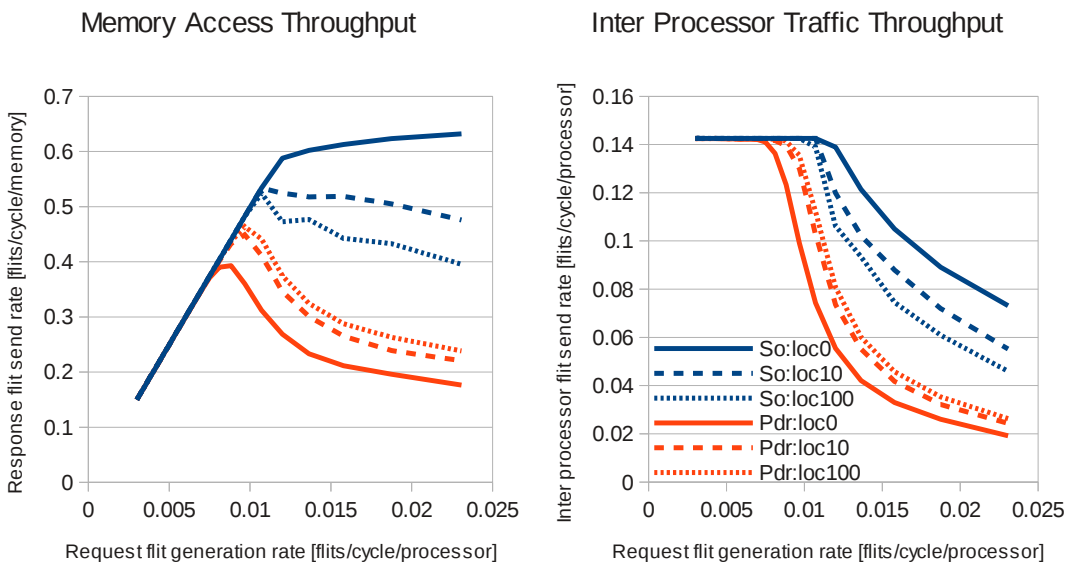

FIGURE 3.15: Effect of increasing localization of memory access traffic on memory response send rates (left) and inter-processor traffic send rates (right).

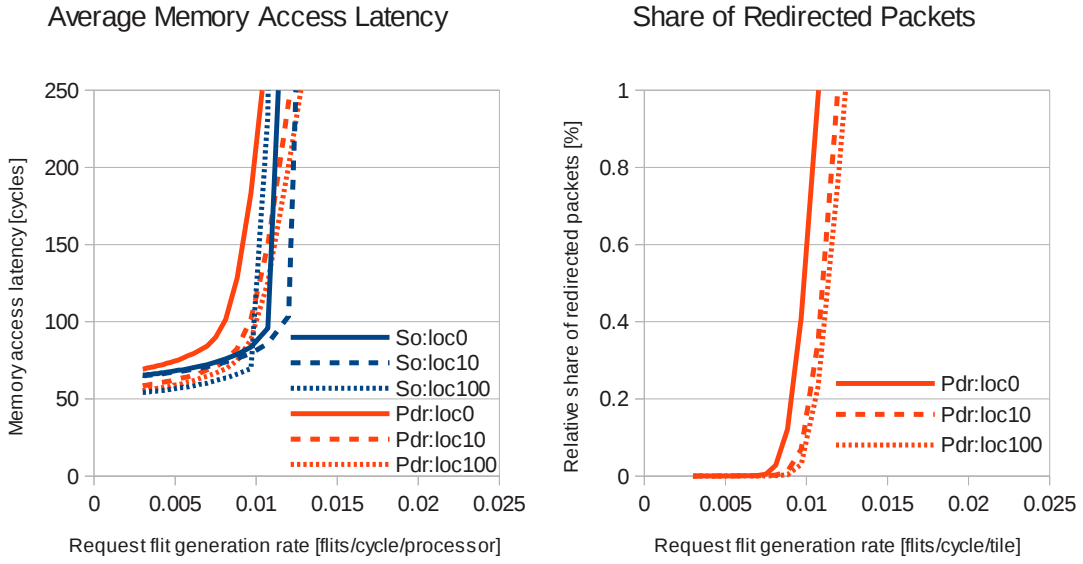

FIGURE 3.16: Memory access latency (left) and share of redirected packets (right) for different localization of memory accesses.

However, quite surprisingly, the NoC applying strict ordering cannot profit from the localization of the memory accesses in terms of throughput. In contrast, both throughput diagrams in Figure 3.15 show a decrease of the throughput graphs. This effect is caused by an increased load in the routers around the memories. If there is no localization, the uniformly distributed read requests lead to a good distribution of the traffic on all the routers along the row of routers at which the memories on top and bottom are located. However, the increasing localization of the requests leads to more and more traffic in the routers directly around the memories. These routers having to cope with the increased load, limit the network throughput. The progressive deadlock recovery NoCs (Pdr:loc$x$) do not suffer from these effects as their throughput is still below that of the NoCs using strict ordering (So:loc$x$), although it is already relatively close for a localization factor of 100.

3.1.2.6 Evaluation of Router Queue Sizes

Figures 3.17 and 3.18 show the influence of the router queue sizes on the throughput of the networks. In the previous simulations, the queue sizes of the routers was 2 flits ($2 + 1$ for the input queue, i3o2). Now, networks with queues extended to 4 and 8 flits per queue are also simulated. With the additional flit register at the input queue, this results in the router buffer configuration denominated as i5o4 and i9o8.

Figure 3.17 shows the simulation results for uniformly distributed memory access traffic. Both network architectures, progressive deadlock recovery (Pdr) and strict ordering (So) profit from the increase of the router queues. The gain in throughput with the step from 2 flit queues to 4 flit queues is larger than from 4 to 8 flits. The reason is that at some point other parts of the network, such as links, the switches, and the memory interfaces become the bottleneck for the network throughput.

In case of the uniformly distributed memory accesses strict ordering again clearly outperforms the networks based on progressive deadlock recovery. However, looking at the throughput graphs for localized memory access traffic (localization factor is 100, see Section 3.1.2.5 for definition) in Figure 3.18 the situation changes significantly. Progressive deadlock recovery (Pdr:i3o2) with half the total flit buffer space compared to strict ordering (So:i3o2), provides only approximately 10% less throughput. Comparing networks with similar total buffer space, i.e. progressive deadlock recovery with 4 flit

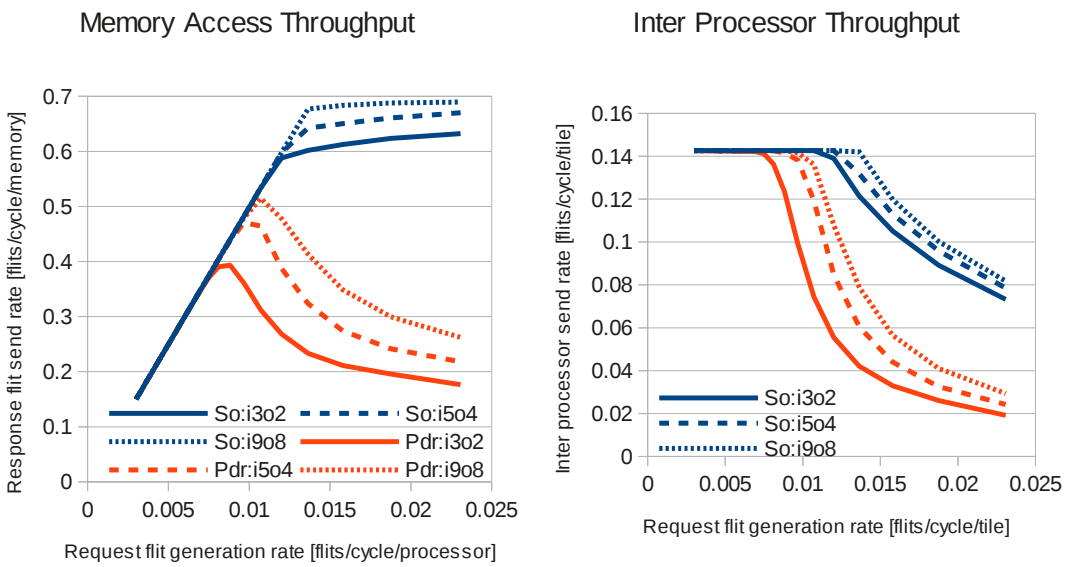

FIGURE 3.17: Memory response send rate (left) and inter-processor traffic send rates (right) for different router queue sizes and uniformly distributed memory access traffic.

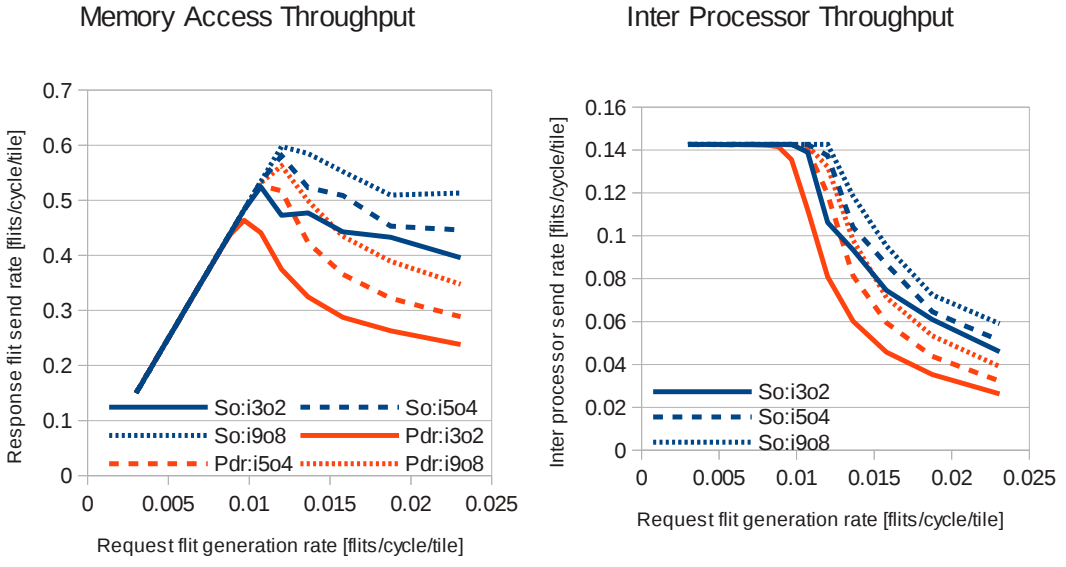

FIGURE 3.18: Memory response send rate (left) and inter-processor traffic send rates (right) for different router queue sizes and localized memory accesses.

router queues (Pdr:i5o4) and strict ordering with 2 flit queues (So:i3o2), gives identical peak memory throughput.

3.1.2.7 Packet Length

Section 3.1.2.2 already showed that the length of packets in the network has an influence on the throughput of a network, also if there are no message dependent deadlocks. In this section, the influence of the packet length on the network performance in the presence of message dependent deadlocks will be evaluated. To do this, the packet length of the background inter processor traffic is varied (iptpl: 3, 5, 7 and 10 flits) in different simulations while the inter processor flit generation rate is kept the same (by reducing the packet generation rate). The memory access traffic is not changed in comparison to the previous evaluations. It is uniformly distributed (localization factor 0). Only networks based on the progressive deadlock recovery concept are simulated and the deadlock detection threshold and the back-off period are 100 cycles and 150 cycles respectively. The router queue configuration of all networks is i3o2, i.e. the length of all router queues is 2 flits.

Figures 3.19 and 3.20 show basically two things:

- A packet length of 5 flit gives the best results of simulated packet lengths.
- However, the difference in throughput and redirection rate is very small. Note the changed scale of the diagrams in comparison to the diagrams of the previous investigations. There is only a clear difference if the packet overhead is considered in the throughput (see Figure 3.20 (right)).

The share of redirected packets of for packet lengths of 3 flits (Pdr:iptpl3) and of 5 flits (Pdr:iptpl5) is more or less equal at low request generation rates. For packets with 7 flits length (Pdr:iptpl7) the redirection rate is already slightly increased, while it is clearly higher for packets of 10 flits length (Pdr:iptpl10). It could be assumed that the reason for the increase of the redirection rate is that the deadlock detection threshold was optimized for the traffic scenarios that contained background traffic with packets of 5 flits. However, a threshold of 100 cycles is quite longer than the waiting time of packet being temporarily blocked by one or even multiple 5 or 10 flit packets. In

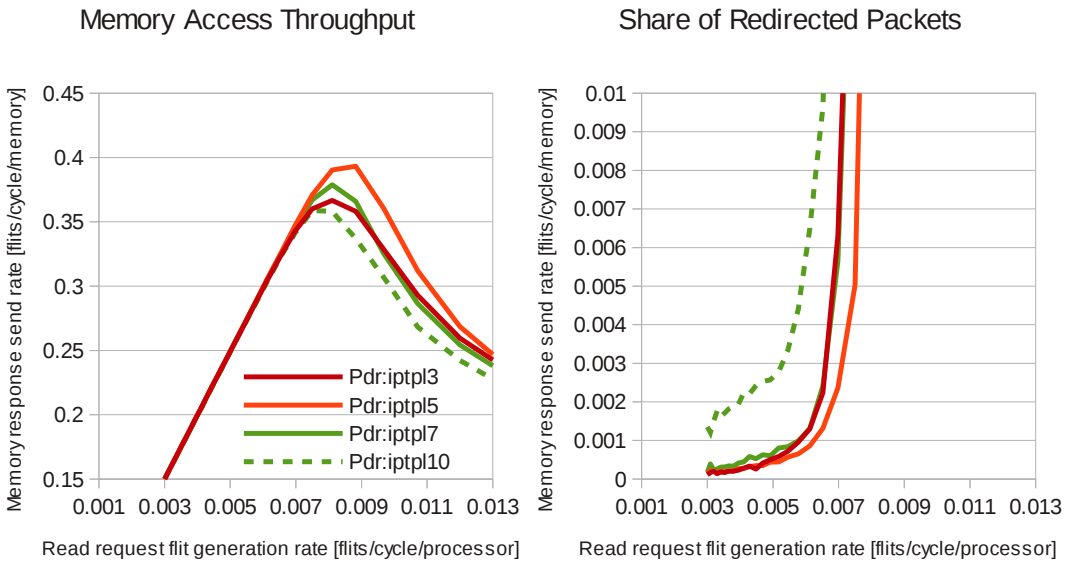

FIGURE 3.19: Memory response send rate (left) and redirection rate (right) for different packet sizes of the inter processor background traffic.

addition, simulations with higher deadlock detection times were also made (not shown in the diagrams) that did however not improve the results.

We could have assumed that the smaller the packets the lower the deadlock occurrence rate or redirection rate and thus the highest the throughput. However, Figure 3.10 showed that the wasted buffer space in networks based on progressive deadlock recovery rises with decreasing packet length. This Figure showed an increase of throughput for rising packet lengths due to the adapted flow control and arbitration function in the absence of message dependent deadlocks. This effect reduces the throughput for the network loaded with 3 flit packets (Pdr:iptpl3), although the redirection rates shown in 3.19 are quite similar for 3 and 5 flit packets. In case of 7 and 10 flit packets (Pdr:iptpl7 and Pdr:iptpl10) the effect of reduced "wasted" buffer space cannot compensate for the increased redirection rate and the resulting throughput reduction.

The inter processor throughput in flits/cycle in Figure 3.20 (left) shows exactly the same picture on behalf of the different packet lengths. Only when the packet overhead (Figure 3.20 right) is considered, the picture changes. This means the higher packet lengths provide a better data throughput, despite the increased redirection rates. However, since the overhead in case of the memory accesses is still the same for all modeled networks, a packet length of 5 flits for the inter processor traffic (Pdr:iptpl5) provides

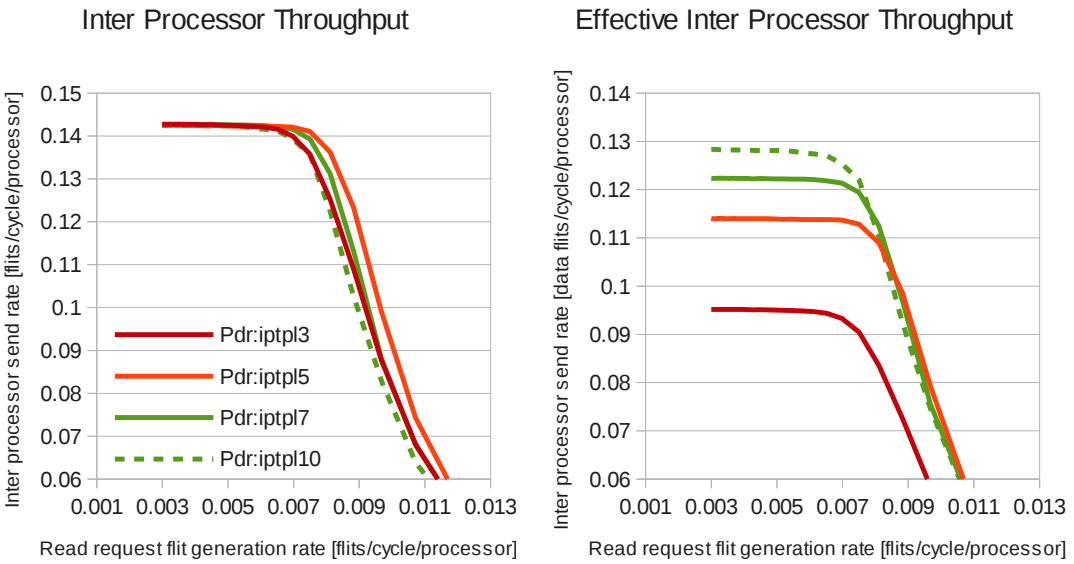

FIGURE 3.20: Inter processor flit send rate (left) and inter-processor data send rates (right) for different packet sizes of the inter processor background traffic.

the best memory throughput. All in all, the number of deadlocks and pseudo deadlocks rises with increasing packet length. However, the effect is quite small at low traffic loads. The rate is below 0.002% for 5, 7 and 10 flit background traffic.

3.1.2.8 Increased Memory Bandwidth

The mapping of the memory tiles at the borders of the network is based on the mapping of Tilera's Tile64 processor [37]. However, the major difference of Tilera's system architecture is that it connects each memory interface at a border of the NoC to four routers, thus quadrupling the theoretical memory bandwidth. In this section, the behavior of the progressive deadlock recovery concept should be evaluated for this system configuration. It is modeled by connecting 4 independent memory tiles to the four routers in the middle of each of the NoC's border. In the simulations shown in this section, the memory access traffic is localized. The corresponding localization factor is 100 (see Section 3.1.2.5 for definition). The simulations are performed for different router queue sizes.

The results shown in Figures 3.21 and 3.22 are in accordance to the results of the previous investigations. The memory response send rate or memory access throughput is again scaled to one memory tile, thus it is correct that the values are not in the range of

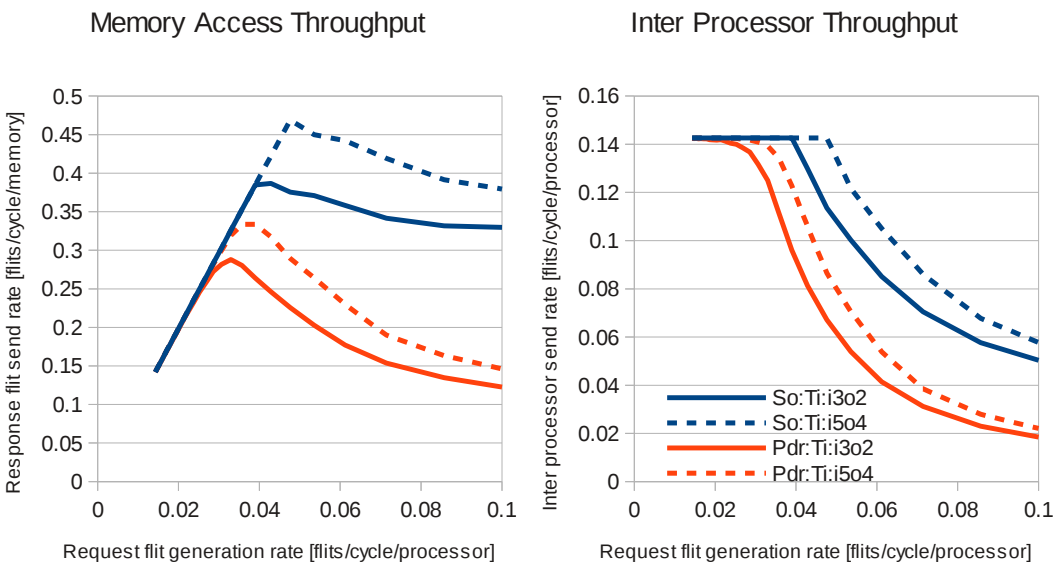

FIGURE 3.21: Memory response send rate (left) and inter processor traffic throughput (right) for different router queue sizes in a system with more memory tiles.

four times the values of the preceding investigations. The reason why the throughput per memory is even decreasing compared to the investigations with only four memories in total is, that now the network is increasingly becoming the bottleneck. In Section 3.1.2.6 strict ordering with router queues of 2 flits (So:i3o2) profited only by 7% in peak memory throughput by an increase of the router queue sizes to 4 flits (i5o4). Here, in the investigation for 16 memory tiles, the throughput gain is 16% for the same increase of router buffer space.

The peak memory access throughput of the progressive deadlock recovery networks (Pdr) is approximately 25% below that of the networks applying strict ordering, when comparing equal router queue sizes. However, comparing networks based on progressive deadlock recovery and strict ordering which have approximately the same implemented total buffer space (Pdr:i5o4 and So:i3o2, see Section 3.1.3), the advantage of strict ordering is reduced to around 10% in terms of memory peak throughput. The graphs in Figure 3.22 show that both network architectures (Pdr and So) profit from the increase of the router queue sizes as it was already expected from the previous investigations. While the performance of the deadlock recovery based networks is clearly below that of the networks relying on strict ordering, the peak memory throughput for network architectures with same total buffer space is within the same range.

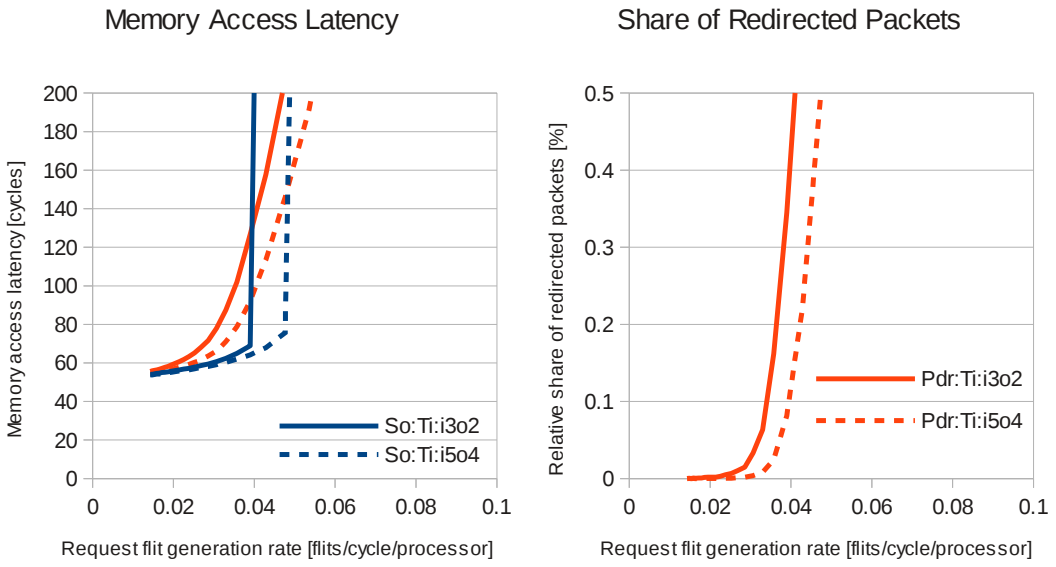

FIGURE 3.22: Memory access latency (left) and redirection rate (right) for different router queue sizes in a system with more memory tiles.

3.1.2.9 Evaluation of Memory Mappings

The mapping of the memories in the previous investigations is based on the mapping of the memory interfaces of Tilera's many-core processor [37]. However, if the tiles implementing message dependencies are not memory interfaces and are not subject to physical or other restrictions, these tiles could also be mapped differently. This section will evaluate different mappings of these slave tiles. The following mappings also shown in Figure 3.23 are modeled:

- Borders: This mapping was also applied in all the previous evaluations, it is derived from the mapping of memory interfaces in [37]. Each memory is placed in the middle of a border of the network (also shown in Figure 3.6 (left) and 3.8).
- Center: The memories are located in the center of the network in form of a rectangle. This mapping is also depicted in Figure 3.7 (left).
- Left: The four memories are placed in one column in the middle of the network's left border.
- CenterCol: As above, but the column of memories is not placed at the left border but in the middle of the network.

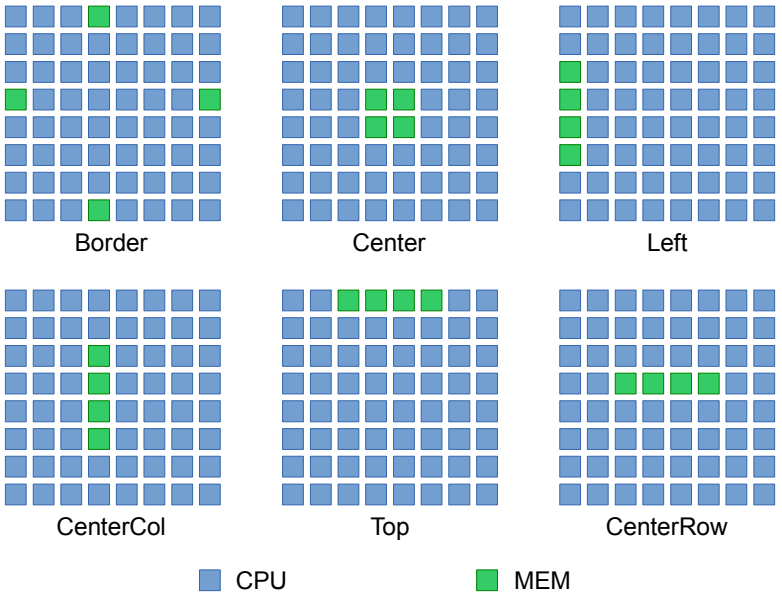

FIGURE 3.23: Overview of the evaluated memory mappings.

- Top: The four memories are mapped in one row in the middle of the top border of the mesh.
- CenterRow: As above, but the row of memories is not put to the top border of the network but in the center of the network.

The applied traffic scenario remains the same. Packets sent between the processors serve as background traffic, uniformly distributed memory access traffic introduces message dependencies and in networks based on deadlock recovery also deadlocks. The router queues are 2 flits long (i3o2), and the deadlock detection threshold and back-off period are 100 cycles and 150 cycles respectively.

The most noticeable fact from the memory throughput diagram in Figure 3.24 is, that only the mapping of the memories at the borders shows the significant drop in throughput after its actually very good peak throughput. The other mappings show a much more stable memory throughput behavior, some however at a lower peak throughput. The reason for this drop is the exploding redirection rate as it is depicted in Figure 3.25 (right). While the rate of deadlocks and pseudo deadlocks is quite low at first, it has the steepest increase at a rate of 0.01 request flits per cycle, overtaking the rate of all other mappings. The high number of deadlocks is due to the high number of packets that can create the forbidden turns and thus the routing cycles.

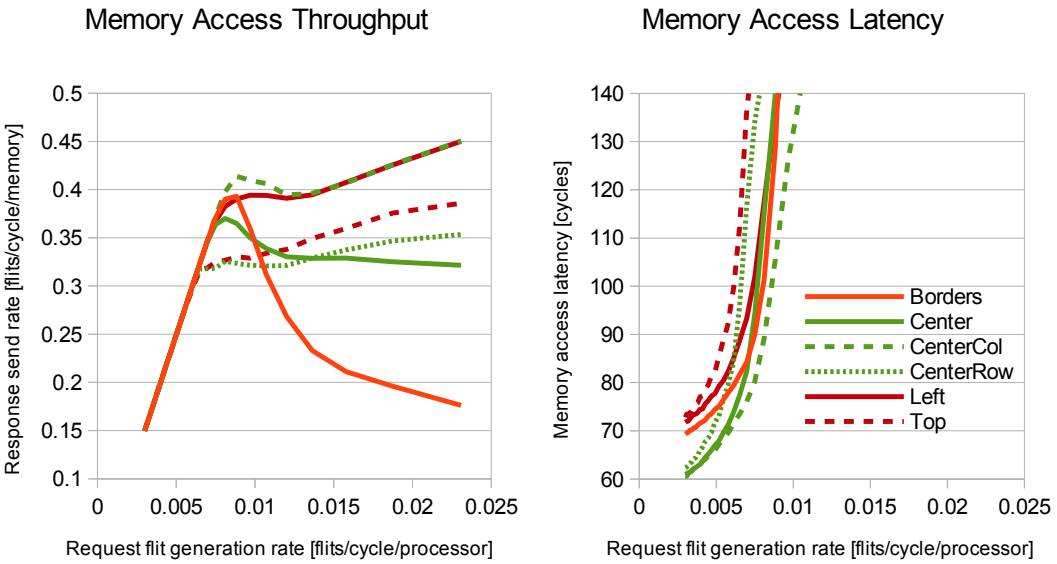

FIGURE 3.24: Effect of memory mapping on memory access throughput (left) and average memory access latency (right).

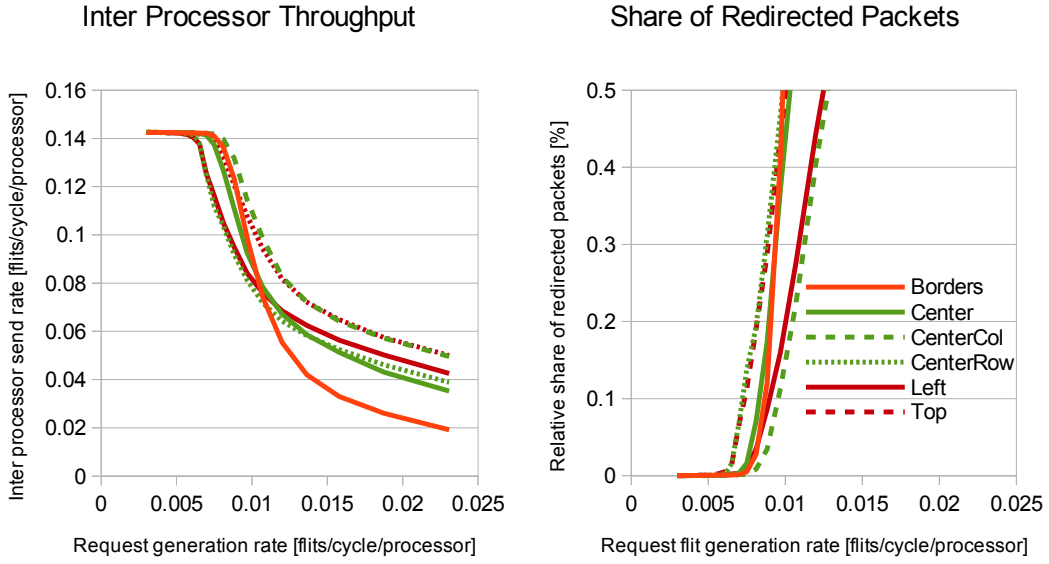

FIGURE 3.25: Effect of memory mapping on inter processor throughput (left) and redirection rate (right).

Mapping the memory tiles in the center of the network already reduces the number of processors or packets that can create the prohibited turns a lot. Accordingly, the drop of the memory throughput after its peak is already reduced by this mapping. However, the other mappings where all memories are put in a row or column enable an even further reduction. These mappings provide an even further increase of the memory throughput after their peak, although at different levels.

The throughput diagrams in Figures 3.24 and 3.25 show that the mappings with the least memory throughput are those that place all memories in one row. In combination with XY-routing this leads to a problem. The read responses sent out by the memories block and hinder each other since they first have to distribute on the horizontal links. This problem applies to the mappings which allocate the memories in rows: CenterRow and Row. When the memories are mapped in a rectangular fashion in the center, like the mapping Center does, this problem is already reduced since only two memories have to share the horizontal links. This is proven by the already increased peak bandwidth of this mapping. The best performance is provided by the mappings allocating the memories in columns: CenterCol and Col. In these mappings, none of the memories has to share its horizontal links with one of the other memories, which leads to a high peak throughput. In addition, these two mapping show no or just a local drop in memory throughput. This is due to the low number of processors, that can provoke the forbidden turns. These turns can only be introduced by the processors that are in the same column as the memories. In case of a 8×8 network, this are only 4 processors. These arguments are also emphasized by Figure 3.25 showing the redirection rate. Both mappings that put the memories in one column have the lowest redirection rates.

The latency diagram given in Figure 3.24 is in accordance to the throughput graphs at high load values. At lower loads the memory latencies of the mappings are quite different. Here, all the mappings that place the memories in the center of the network have a clear advantage of up to 20% in comparison to the other mappings. The reason is the reduced average network distance between communicating processors and memories.

The mappings CenterCol and Col which arrange the memories in columns in the network show the best performance in terms of memory throughput. In addition, CenterCol shows a much better inter processor throughput. This is due to the lower network distance between the processors and the memories. The higher this network distance,

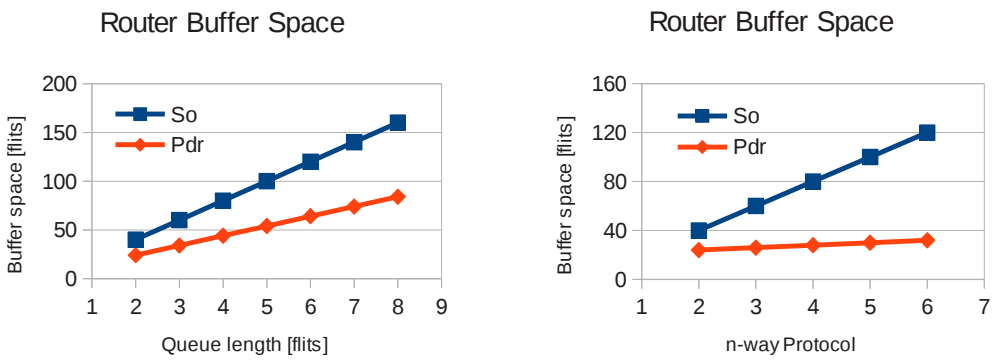

FIGURE 3.26: Comparison of router buffer space for strict ordering and deadlock recovery for for different queue lengths (left) and for protocols with different number of dependent messages (right).

the more link bandwidth has to be used for each memory access of a processor, which is then no longer available for the inter processor traffic. As the average network distance is higher if all memories are located at the left, more link bandwidth is required for the memory accesses and correspondingly the inter processor throughput is worse.

This evaluation shows that the throughput significantly depends on the mapping of the tiles that introduce the message dependencies. If there are no physical constraints for the mapping of such a tile, then the findings from above should be considered and simulations be made in order to find optimized mappings for a system.

3.1.3 Evaluation of Noc Wide Buffer Space Requirements

Regarding the required buffer space of a router the progressive deadlock recovery concept enables significant reductions. Figure 3.26 shows the buffer space requirements of routers based on an input and output buffered architecture, that implement either strict ordering or the proposed deadlock recovery mechanism. Nevertheless, the following evaluations also hold true for only input buffered router architectures, with the difference that the savings will be a little smaller in absolute and relative numbers.

In the routers, almost half of the buffer space can be saved relative to strict ordering for networks supporting 2-way protocols. The savings are not exactly 50% since the deadlock channel also requires some buffers. With increasing queue lengths in the routers (see 3.26 on the left) the savings of progressive deadlock recovery compared to strict

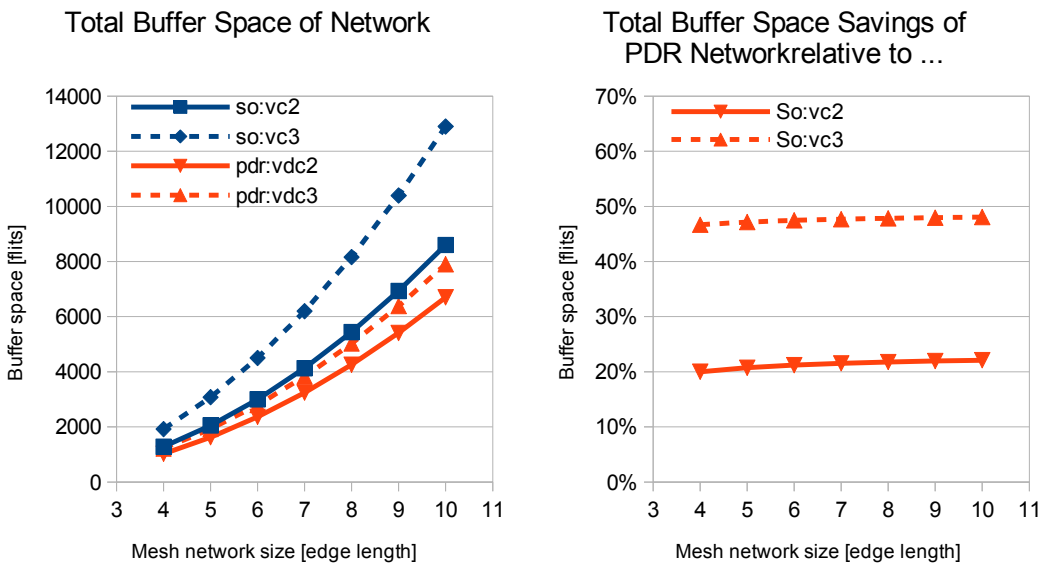

FIGURE 3.27: Absolute buffer space requirements of complete network (router and network adapter) (left) and relative savings of progressive deadlock recovery compared to networks based on strict ordering (right).

ordering approaches more and more the limit of 50% for networks supporting protocols with 2 dependent messages. In this diagram, it is assumed that the queue length of the deadlock channel buffers does not increase. This would not have any positive effect on the throughput anyhow, since the exclusive access to the deadlock channels avoids any contentions and congestions within the deadlock channel. Thus, the buffer requirements of the deadlock channel only rise with the number of message dependencies in the communication protocols. For rising numbers of dependencies, the total buffer space of a router implementing strict ordering rises even faster. Therefore, the savings of deadlock recovery compared to strict ordering increase even more as depicted on the right in Figure 3.26.

Considering the network adapter, the savings are significantly less, since each virtual deadlock channel requires its own input buffer. Assuming equally long input and output buffers, a network adapter with 2 virtual channels on its input and output side requires 4 buffer queues. A network adapter supporting the proposed deadlock recovery concept requires the same number of queues: two queues for the standard input and output path, and 2 input queues for the two virtual deadlock channels. Chip area is saved in the network adapter for deadlock recovery only, if the network has to support n-way

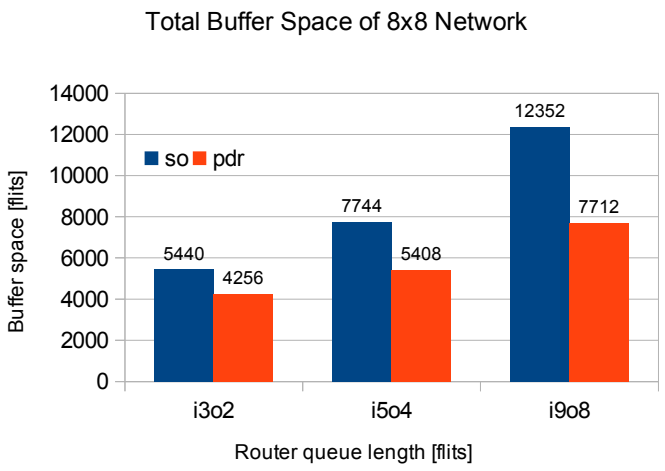

FIGURE 3.28: Total buffer space of routers and network adapters of a 8×8 NoC for different router queue lengths.

protocols with $n \geq 3$. In addition, the compared network adapter implementing strict ordering must require all input and output queues for all virtual channels of the network. If a tile does not receive or send all message types existent in the network, then some of the input or output queues of the network adapter could be saved. However, the same holds true for the network adapter of the deadlock recovery concept. This step should be applied very carefully, as it has to be guaranteed that later on, the software does neither introduce message dependencies in the corresponding tile.

The required buffer space of complete networks applying either progressive deadlock recovery or strict ordering is shown in Figures 3.27 and 3.28. The left diagram of Figure 3.27 shows the total buffer space of the two competing network architectures for support of 2- and 3-way protocols. As expected, the gap between progressive deadlock recovery (Pdr) and strict ordering (So) increases with the network size, as each router enables buffer space savings for the deadlock recovery compared to strict ordering. This translates into 20% buffer space savings for networks for 2-way protocols and 50% for 3-way protocols in the diagram on the right. Hereby, the relative savings show only a weak dependence on the network size.

The total buffer space savings for networks implemented for 2-way protocols rise significantly, if the router queue sizes are increased. The comparison for the router queue

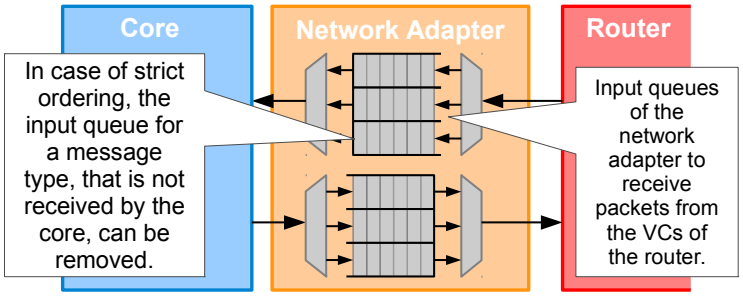

FIGURE 3.29: Removal of input queues for message types not received by a core.

configurations i3o2 (router queue length 2 flits), i5o4 (4 flits) and i9o8 (8 flits) are depicted in Figure 3.28. Besides the savings of the progressive deadlock recovery concept in case of equal router queue sizes, this figure also shows that the router queue sizes of routers implementing progressive deadlock recovery (Pdr) can be doubled in comparison to those implementing strict ordering (So) while still requiring less buffer space.

3.1.4 Summary and General Evaluation

As demonstrated in the previous sections, the progressive deadlock recovery concept for NoCs saves significant amounts of buffer space in the routers compared to strict ordering. Almost 50% for networks that support two-way protocols, and even more for networks with more complex protocols.

In contrast, the buffer space requirements of network adapters of the two competing concepts are not that different and not that easy to determine. Buffer space savings in the network adapter for deadlock recovery only occur if the network has to support protocols with more than 2 dependent messages and if the connected tile receives and sends all types of messages. If not, the corresponding input or output queues can be removed in the network adapter for strict ordering (see Figure 3.29) as well as in the one for deadlock recovery. Thus, the savings or whether there are any at all in the network adapter depend on the specifics of the application or the traffic scenario. However, removing a tile's support for certain virtual channels or dependent messages should be carefully considered. As software running on a system can also create dependent messages in a tile, it has to be taken care that situations in which previously removed queues would be required for such messages do not occur.

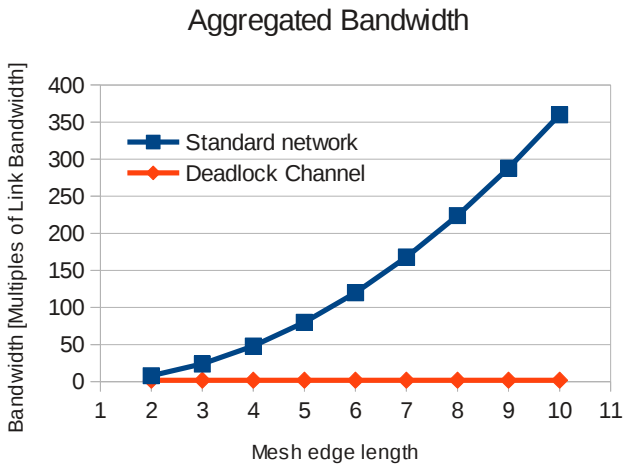

FIGURE 3.30: Aggregated bandwidth of standard mesh network and of its deadlock channel for 2 virtual deadlock channels.

Depending on the traffic scenario and mapping of the tiles, the progressive deadlock recovery concept can provide comparable throughput to strict ordering. For some mappings there is a significant drop in throughput after their peak throughput in overload situations. However, neither networks based on progressive deadlock recovery nor strict ordering should be operated in such overload situations for longer periods of time or even constantly. Such a scenario would lead to steadily increasing network access latencies.

The central problem of the proposed deadlock recovery concept that limits its performance is the relatively low bandwidth of the deadlock channels. Since the original design target in the development of this concept was to save chip area, the deadlock channel is realized with only a minimum of buffer space. This leads to only one central buffer within a router and exclusive access to each virtual deadlock channel. Figure 3.30 shows the huge difference in bandwidth of the normal network and the deadlock channel of the network for different sizes of mesh. For example in a 8×8 mesh network the aggregated link bandwidth is $224 * BW_{link}$. In contrast, the bandwidth of the deadlock channels is $n * BW_{link}$ (assuming n virtual deadlock channels and that the redirected packets of different message types travel on non-intersecting routes). The reason for this difference and for the fact that the aggregated bandwidth of the deadlock channels does not scale (see Figure 3.30), is the exclusive access to the virtual deadlock channels.

Due to this low bandwidth of the deadlock channel, redirections should only be performed very rarely. This means, the deadlock channels should only be used for real deadlocks and not for pseudo deadlocks. In case many temporary blockages are detected as deadlocks and one of the corresponding packets is redirected, then in case of a real deadlock the time until it can be redirected is not only the time until it has been detected but in addition also the time to remove the previously redirected packet from the escape channel. In consequence, the timer based deadlock detection must be operated at high thresholds in order to redirect mostly real deadlocks and as few pseudo deadlocks as possible. However, the long detection times in turn lead to a congested and filled up network in case of a real deadlock. This leads to the necessity of the back-off mechanism in order to empty the network and return it into an operable state. The back-off mechanism also reduces the throughput. The more often it is activated the more the effective throughput of the network is reduced. For high traffic loads this leads to a vicious cycle: the high traffic load leads to a high deadlock occurrence rate, this in turn leads to a high activation rate of back-off mechanism which reduces the throughput of the network.

All in all, the progressive deadlock recovery concept is able to save significant amounts of buffer space especially in case of systems that have to support protocols with more than 2 dependent messages. The provided network performance depends on the application and the mapping of the tiles. In some scenarios it can even reach the same performance as strict ordering. This dissertation suggests its use for applications with low bandwidth requirements, that do not need the additional bandwidth provided by the virtual channels of strict ordering. By increasing the router queue length, the progressive deadlock recovery concept can be adapted to applications with higher bandwidth requirements.

3.2 Regressive Deadlock Recovery for Networks-on-Chip: sdNoC

Although dropping packets is done in many communication networks, regressive deadlock recovery or discarding packets due to any reason is not popular in on-chip networks. Generally, it is not done because of unreliable communication media, and thus because of inevitable data loss. Discarding single packets is actively done in queues of network equipment in context of a method called RED (Random Early Discard [92]). This mechanism selectively drops packets depending on buffer fill levels in order to avoid local congestion and hereby increase the total network throughput. To provide lossless communication for higher communication layers, many network protocols, such as TCP/IP, are anyhow designed to cope with bounded packet loss.

This dissertation proposes to apply this idea of actively discarding packets under certain circumstances to packet switched, wormhole based Networks-on-Chip that are especially susceptible to message dependent deadlocks [32]. Then, these deadlocks do not have to be completely avoided by costly measures, but can be simply solved by dropping deadlocked packets. This dissertation will also show that selectively dropping packets can have a positive effect on the throughput in NoCs, as it has in other networks. In order to provide lossless communication to the tiles or cores retransmission capabilities are added to the network adapters. These capabilities can be used to provide tolerance against transient errors at the same time. This aspect will become more and more important since nanometer CMOS technologies will become increasingly sensitive to all forms of manufacturing and environmental variations. Thus, efforts to cope with data corruption or loss will become a necessity anyhow.

Regressive deadlock recovery has several advantages over the progressive deadlock recovery concept. Deadlocked packets do not have to be redirected but can simply be dropped. Thus, the solution of a deadlock is not dependent on a limited resource of an infrastructure, such as the low bandwidth of the recovery channels, which was the major bottleneck of progressive deadlock recovery. This removes the necessity for implementing a back-off functionality for the tiles and thus also the implementation of an additional ring network for access mechanisms and propagation of back-off signals.

Another drawback of the progressive deadlock recovery concept are the additional input buffers required in the network adapters to enable reception of packets from the deadlock channels. This diminished the savings of buffer space that are possible in the routers of the network. The regressive deadlock recovery also requires additional buffers in the network adapter. However, since these queues are required at the sender side, they can be used to provide fault tolerance in the communication at the same time.

The concept employing the ideas listed above is called sdNoC in this thesis: Selective Discard NoC. sdNoC is described and evaluated in detail in the following sections.

3.2.1 sdNoC Concept

The sdNoC concept relies on two criteria for discarding packets. A timer based mechanism attached to the queues in the routers, enabling the queues to drop packets that have been buffered in the queue for a certain time without moving on. This resolves not only deadlocks, but also temporary local congestions and can hereby also increase the total network throughput. The second criterion for discarding packets is the detection of bit flips in the flits of a packet. The evaluation of this criterion is performed in the network adapter of the destination.

Independent of why packets are discarded the sdNoC concept has to consider the special characteristics of wormhole forwarding and the router architecture of NoCs. According to the wormhole forwarding packets can only be dropped from the header flit on and all flits have to be dropped completely, even if the packet stretches over multiple routers. The target of reducing chip area and complexity of the routers in comparison to other approaches, such as strict ordering, requires that the corresponding queue managing functionality enabling deletion of packets is implemented with a minimum of additional overhead. If possible, it has to be taken care that the extended FIFO control logic does not lead to an extension of the critical path in the NoC router, so that the maximum link bandwidth at the routers is not reduced in comparison to alternate approaches.

For the realization of the retransmission capabilities required in sdNoC, two alternative architectures are proposed by this thesis:

- The first one stores the sent packets in dedicated buffers in the network adapters for retransmission.

- If the tile is equipped with tile local memory, a part of this local memory can be reserved and used to store packets for their retransmission. This is the preferred architecture for the retransmission functionality.

Packets are removed from the retransmission buffer, whether implemented in the network adapter or local memory of the tile, when their reception has been acknowledged by the destination. The acknowledgement can either be performed explicitly by an acknowledgement packet or implicitly by an anyhow sent response packet, as in case of a read response.

3.2.1.1 Packet Discard

In sdNoC the criterion for discarding a packet in the router is based on the queue latency. If the header flit of a packet stays in a queue or a pipeline stage of a router longer than a specified threshold the corresponding packet is discarded. In contrast to RED [92] the proposed criterion does not consider buffer fill levels to calculate discard probabilities. Considering this input parameter is not applicable in NoCs, since the small router queues of two or four flits fill up immediately even in case of short temporary congestion.

The timer based criterion is able to detect all deadlocks, independent of their type, since all deadlocks are caused by packets forming a circular path in the network and waiting for each other to free next buffers. As this event will never happen due to the circular dependency, these packets are stuck in a infinitely long waiting state and will thus be discarded by the described criterion (also see Section 2.4 for detailed description of deadlocks). SdNoC resolves only message dependent deadlocks by dropping packets in order to keep drop rates and thus retransmission rates low. Routing dependent deadlocks are handled by deadlock avoidance as it is common in NoCs.

In addition, the timer based, heuristic discard condition also solves temporary congestion in the NoC. With wormhole forwarding, a temporarily blocked packet can block many other packets and their desired links, which can even be idle. As described in Sections 2.2.2.3 and 3.1.2.2 link bandwidth is wasted in this case. By discarding the corresponding packet, the other packets can move on and the link bandwidth is no longer wasted. Undoubtedly, there are also cases in which the packet that is allowed to enter the queues

of the discarded packet simply results in the same congestion as before. Nevertheless, the discard of single packets is able to improve the overall network throughput.

The discard criterion is realized by adding a timer to each queue or stage in the router that possibly stores a header flit of a packet. The timer is started or reset every time a header flit enters the queue, thus the discard criterion is always evaluated for the last packet that entered the queue. If the timer reaches the specified threshold (adequate values are explored in Section 3.2.2.2), the packet is dropped. The freed buffer space allows a new packet to enter the queue, for which the timer can easily be started again.

An alternative approach would be to execute the discard criterion on the first header flit in the queue. However, then it would be necessary to search the queue for further header flits after deleting a packet. Thus, this approach is not used in sdNoC.

In case a packet that should be deleted is stored completely in one queue, the packet can be deleted within one cycle by manipulating the write pointer of this queue. However, if the packet spans over several queues of a router or even over several routers as it is possible with wormhole forwarding, then, manipulating the write pointer is not sufficient. Two alternative approaches are possible:

- The command for deleting all flits of the corresponding packet could be propagated backwards along the path of this packet until its tail flit is reached. However, forwarding the delete command requires additional wires between the routers and additional logic to forward the command in reverse direction over the router's internal switch.
- The second alternative which is chosen for sdNoC requires no such extensions. After deleting the flits in the queue that triggered the discard command, the queue allows all subsequent flits of this packet to enter the queue. In order to delete all these flits, the write pointer is not updated again until the tail flit of this packet arrives.

In this way, the complete functionality of deleting packets can be implemented by adapting the queue management functionality that controls the read and write pointers. This eases the complexity of implementation significantly, since the changes are local to some

modules of the router, and do not require changes of the router architecture nor changes of any interfaces.

Basically, sdNoC uses a specified, fixed discard threshold in all routers, that is equal for all packets. Accordingly, the packets are treated in the discard process independent of how many times they already have been retransmitted. At high loads of the network which result in relatively high discard rates, this can lead to single packets being retransmitted multiple times before their transfer is successful. This can be a problem for applications that suffer from too high packet transfer latencies. However, it is not possible to simply guarantee the transfer of packets that have for example already been retransmitted one or two times. In this case, it would be possible that a deadlock is created that consists only of packets that have been retransmitted multiple times and thus cannot be dropped. Then, the deadlock could not be resolved. Nevertheless, it is possible to reduce the probability for discarding packets that have already been transmitted in comparison to packets that have not been retransmitted yet. This can be achieved by increasing the discard threshold for packets depending on their retransmissions. While the mechanism is able to reduce the number of multiply retransmitted packets, neither the share of dropped packets nor the average packet latencies change significantly. In consequence, this means that the number of once retransmitted packets even increases.

The second criterion for dropping packets in sdNoC are bit flips caused by transient errors during transfer in the network. To enable the detection of all possible bit flips that could occur, two extensions to the packet and flit format are necessary (also see Figure 3.31):

- All flits have to be extended by CRC bits, to enable the detection of data corruption and of flips of the flit type.
- Introduce a length field in the header flit that stores the length of the packet, i.e. number of flits. Together with the CRC bits of every flit, this measure helps to detect corruptions in the packet compound. For example, if a data flit becomes a header flit due to a bit flip, and the rest of the original packet is then sent to another destination.

After a packet has been completely received by the network adapter at its destination, it is checked for errors. In case of any errors it is simply dropped, no acknowledge is

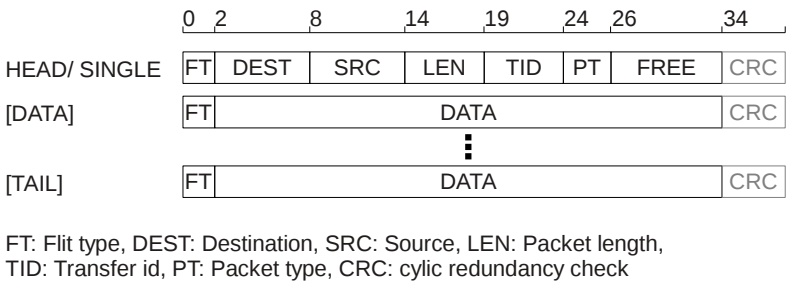

FIGURE 3.31: Packet and flit format of sdNoC.

sent to the source, and the retransmission of the packet is awaited. The error detection functionality is only implemented in the network adapters and not in the routers. First of all, performing the error check of each flit at each hop would result in a significantly higher power consumption. Second, in case an error is detected for a non-header flit, it is hard or even impossible for the router to remove the complete packet. The header flit of this packet might already be in the next router.

In order to implement fault tolerance in the communication infrastructure of a system, error detection functionality is not necessary in the router. Nevertheless, the routers have to be able to continue their operation in the presence of packets of which bits, that are evaluated by the router, have been corrupted. Examples of bit fields evaluated by the sdNoC routers are the flit type and the destination field. In a standard router bit flips of these fields could lead to a non-operational state of a router and then of the complete network. In sdNoC, corruptions in the destination field of a header flit are handled by using the local output port of the router as default value in the routing logic. Thus, packets with corrupted destination fields leave the network immediately and can no longer disturb the operation of the network. Note that the router does not execute CRC checks on the flits. Only the network adapter receiving this packet, detects the error in the header flit due to the CRC code and can then drop the entire packet.

The second case of errors that the routers must be tolerant against are bit flips in the flit type field. Such bit flips lead to the destruction of the packet compound in the network. Assuming a bit flip converts a tail flit into a data flit (T $\rightarrow$ D), the following packet would be transferred to the destination of the previous packet if the routers only reset their internal routes (input to output channel) after tail flits. In order to handle

all possible flips of the flit type, the control logic of the routers has to implement two rules in addition to its standard routing and wormhole forwarding functionality.

- Rule 1: After receiving a tail flit, the router has to drop all flits until the arrival of a new header flit. This rule makes sure that a packet of which the header flit flipped to a data flit (H $\rightarrow$ D) is completely removed from the network. Rule 1 also handles the type flips $H \rightarrow T$ and $D \rightarrow T$. In case of $H \rightarrow T$ the packet of the flipped header is completely dropped by the router.

  If a type flip $D \rightarrow T$ occurs, only the remaining data flits and the tail flit after the flipped data flit are dropped. The first part of the packet, that is transported to its correct destination, is then dropped in the corresponding network adapter which detects that the packet is incomplete due to the length field in the header.

- Rule 2: Route every header flit, i.e. perform the determination of the output port for every header flit that arrives, even if there was no tail flit to close the previous route. This rule guarantees that after a type flip $T \rightarrow D$ the following packet is nevertheless forwarded to its correct destination, since the router sets the correct output port for the header flit of the following packet.

  This rule also handles the corruption of a tail flit that becomes a header flit $T \rightarrow H$. In this case the first part of the packet (except the original tail flit) is routed to the correct destination, where the reduced packet length is detected and the packet dropped by the network adapter. The tail flit that became a header flit is routed to another destination, where it is also discarded.

  Packets that are subject to a $D \rightarrow H$ flip are handled similarly. The only difference is that the misdirected second part of the original packet is not only one flit, but multiple flits, depending on which data flit struck the transient error.

These two rules enable the routers to handle all possible six flit type flips and to keep the network in an operable state. Note that the routers do not have to check the CRC code of the flits. Corrupted packets are either dropped immediately or routed to a network adapter where the corruption can be detected with the help of the CRC bits of every flit and the packet length field of the header flit. Then, the corrupted packet is dropped in the network adapter.

Providing tolerance against type flips in the routers and implementing error detection at the receive side of the network adapters provides fault tolerance against all possible transient bit errors. Errors can be handled independent of whether they occur in the routers, on the links or in the router functionality. In contrast, most ECC mechanisms or hop-by-hop based retransmission concepts can only secure against transient faults on the links. However as [93] states, the share of errors occurring in the router itself cannot be neglected in comparison to the faults on the links. While hop-by-hop based retransmission with error detection in each router would lead to less traffic in the network, compared to end-to-end retransmission, the fault tolerance mechanisms of sdNoC as described above result in less router complexity. This concept allows for simpler router architecture, enables higher clock rates and requires significantly less chip area than a hop-by-hop based retransmission concept requiring error detection units at every input or output port of the routers.

3.2.1.2 Retransmission and Transmission Control

The network adapters of sdNoC implement retransmission of packets to provide lossless communication for the connected cores despite the lossy communication in the network itself. Therefore, packets that are sent into the network also have to be stored in the source tile. The packets can either be stored directly in the network adapter or if available in a local memory of the tile. The retransmission mechanism is timer based, which means a timer has to be implemented for each packet that should be sent out, without waiting for the acknowledge of a previously sent packet. An alternative to triggering the retransmission by a timer would be triggering it by retransmission request packets. However, since these packets must not be dropped an additional lossless network that transfers only these packets would be required. Therefore, the timer based retransmission is chosen for sdNoC.

The resend or retransmission period specifies the time after which a packet has to be sent again if its corresponding acknowledgement packet has not arrived in the meantime. This period has to be chosen carefully. In case it is set too high, resending dropped packets takes very long. This increases the latency of these packets and reduces the network throughput at the same time. Even more critical are too low resend periods, which can lead to overload situations of the network as packets are resent that actually just got

delayed by a few cycles because of temporary blockages in the network. As overload situations lead to even more and longer blockages this can lead into a vicious circle.

If the resend periods are purely static it is possible that the same deadlock situations occur repeatedly in the network, thus resulting in a livelock. Let us assume that the system depicted in Figure 2.2 implements sdNoC and both tiles 1 and 3 started to send their packets within several cycles (so that a packet has not yet passed the router of the other tile) and the resend period and discard threshold are the same. In this case, the deadlock depicted in this figure would be repeated infinitely. This situation can also be called a livelock. In drop-retransmit based networks the absence of such situations cannot be guaranteed, although their occurrence is insignificant. In the performed simulations, problems concerning network throughput arose only in case of clearly too small resend periods. Nevertheless, livelocks can be probabilistically avoided by adding a small number of randomly determined cycles to the static resend period.

The size of the retransmission buffer determines the number of packets that can be sent by the network interface without having to wait for the acknowledgement of a sent packet. Thus, it is also called outstanding requests in the following. After a network adapter has sent out its maximum number of outstanding requests, that means the retransmission buffer is fully occupied, it has to wait until one of these transactions is completed and the corresponding space is freed in the retransmission buffer. If the size of the retransmission buffer is too small, the injection rate of the network interface into the network will be limited by the retransmission buffer.

Considering an isolated tile, its maximum possible injection rate rises with increasing size of the retransmission buffer up to a certain size. It saturates when the tile is able to send a flit every cycle without having to wait for free space in the retransmission buffer. If we consider a system with many tiles the situation changes. Larger retransmission buffers no longer necessarily translate into higher throughput. Instead there is an optimal value for the number of outstanding requests assuming that all tiles try to inject packets into the network. As described above, designing the retransmission buffer too small limits the send rate of the network adapter. However, enabling a high number of outstanding requests also affects the throughput negatively, as this can lead to overload situations in the network that reduce the effective network throughput. This issue is discussed in Section 3.2.2.3 in more detail. If the resend buffer should be implemented larger than

the actually optimum size considering total network throughput, for example because this increases the efficiency of the processor core, then the sending rate of the network adapter has to be limited (e.g. by windowing) to the optimum size.

In sdNoC, not all messages of an n-way protocol are stored in the retransmission buffer, if this can be avoided. Where possible the retransmission should only be applied to the first message of the chain of dependent messages. In case of a read access from a processor core to a memory that consists of a request and a reply packet (2-way protocol) only the request packet would be stored in the retransmission buffer of the sender. Independent of whether the request or the reply packet is dropped, which serves as acknowledgement packet at the same time, the request packet is retransmitted. Of course, this approach can only be applied if the tile sending the read reply can send the requested data again and can guarantee that it has not yet been overwritten or deleted. This approach avoids large retransmission buffers at slaves like memories. If every reply packet sent by a memory would have to be stored for retransmission in the retransmission buffer, the buffer would be quite large, since the memory can receive requests from many cores of the SoC. The approach of just covering the first packet of a communication protocol for retransmission also reduces the additional traffic for acknowledgements, as long as drop rates are reasonably low. Alternatively, an acknowledgement packet would have to be sent for each transmitted packet.

However, this approach also suffers from two drawbacks. First, it becomes inefficient for protocols with higher numbers of dependent messages. For example, if the fourth packet of a 4-way protocol gets dropped it is inefficient to start again with the first packet. Second, this concept does simply not work for all kinds of communication partners. As already mentioned above, it can only be applied if the destination of the original request can provide the requested data again for a retransmitted request. Considering the read access to an I/O interface, this approach is not applicable. Here, all requests and responses have to be covered by the retransmission functionality of the network interface. In memories, coherence problems can arise. It cannot be simply assumed that a specific set of data will be available for retransmission at some later time. For example, a block of data is read from the memory and then sent in a response packet to the requesting tile (t1). Now, while this response packet is dropped in the NoC, the considered block of data is overwritten by another tile (t2). After some time, tile t1 retransmits its original request but can no longer receive the actually requested data,

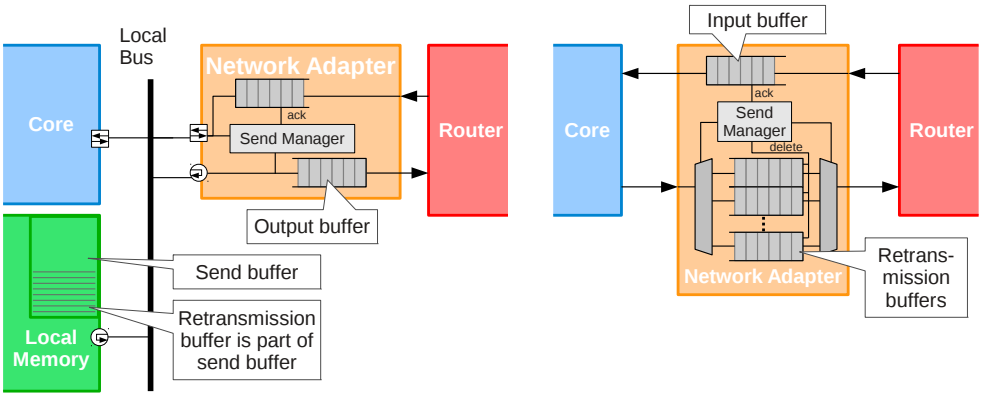

FIGURE 3.32: Network adapter architectures implementing the retransmission buffer in local memory (left) or in the network adapter itself (right).

as it was already overwritten. This coherence problem has to be solved by a superior protocol layer. This layer has to guarantee that the according data is not overwritten by other processes before possibly ongoing transfers have not been successfully completed.

3.2.1.3 Storage Requirements

Basically, there are two alternative architectures to implement the retransmission buffer described in the section above. If the tile contains a local memory, a part of this memory can be reserved to operate as retransmission buffer, while the network interface just has to implement the required functionality but not the storage. If no local memory is available in a tile, then the alternative is to realize the storage for the retransmission directly in the network interface. While this architecture leads to higher chip area requirements of the network interface, it removes load from the tile's local interconnect. Independent of the architecture that is used to realize the retransmission storage, it has to be protected against soft errors by ECC.

Whenever possible, this thesis proposes to use the architecture shown in Figure 3.32 on the left in which a small part of the tile's local memory is reserved to store sent packets for retransmission until their acknowledgement. In this architecture, the network interface just implements the functionality to retransmit and delete acknowledged packets, but not the storage of the packets. In addition to this, the interface requires the conventional logic for sending packets into the network and receiving them from the network. It also requires an input and an output queue. The output queue is able to store outgoing

flits so that accesses to the retransmission memory over the bus in order to retrieve single flits can be avoided. While the retransmission buffer size or the injection rate of the network interface into the network should be limited to the optimum number of outstanding requests, the sending buffer of the tile in the memory can nevertheless be larger to avoid stalls of the processor core to wait for free space.

Although it does not have to store the data of the packets, the network adapter still has to store some data on the packets, such as the request type, the destination tile, the address of the data in the retransmission buffer and others. Some of this information could theoretically also be placed in the local memory of the tile. However, the network interface would then have to access the local memory for every piece of information it might require. This would lead to a higher load on the tile's local interconnect, and could lead to an additional delay of packets that the network interface wants to send but cannot due to an already occupied local interconnect. Such situations would then also lead to a limitation of the tile's send rate. In contrast, the proposed architecture that stores some basic information on the packets or messages already in the network interface, can already create the header flit and possibly also the second flit (containing the read or write address at destination tile) without access to the memory. It can already trigger the memory access when it starts to create the header flit, so that it will have the data for the following flits available at the time it is required.

In an 8×8 many-core processor a retransmission buffer of approximately 5 packets is sufficient (see simulations in Section 3.2.2.3). Assuming a maximum of 10 flit packets (32 bits per flit), this translates into 1.6k Bit. This is a very small share in comparison to the typical sizes of such local memories of 100 kBit to 1 MBit.

The alternative architecture (shown in Figure 3.32 right) stores the packets for possibly required retransmissions directly in the network interface. The advantage of this architecture is that the network interface does not rely on the local interconnect for (re-)transmitting a packet. Thus, sending of packets cannot be hindered by long access times to the tile local interconnect. In addition, more internal communication bandwidth is available for the processor cores. The major drawback of this architecture is that implementing the retransmission storage in the network adapter leads to a significant increase of required chip area. In this case, the size of the retransmission buffer will probably be chosen according to the optimum number of outstanding requests as described above. If

chip area requirements should be reduced, the size of the retransmission buffer can also be chosen below the optimum number of outstanding requests.

3.2.1.4 Acknowledgement of Transfers

In sdNoC the confirmation of successfully sent packets is done by acknowledgement packets. In case of a write request, the confirmation will be done by an explicit acknowledgement packet. Explicit acknowledgements are single flit packets. The confirmation of a read request is implicitly done by the read reply. In this case, no additional traffic is generated by acknowledgements.

Using retransmission requests instead of acknowledgements is not an option for sdNoC, since these requests would then have to be transferred lossless. As sdNoC relies on discarding of packets to solve deadlocks, the retransmission requests could not be transferred in the lossy NoC. Instead an additional lossless NoC would be necessary. This is not an option, as one of the design targets of sdNoC is the reduction of chip area requirements. Besides, acknowledgements would still be necessary as otherwise the network interfaces would not know when a packet was successfully transferred to its destination. Without acknowledgement packets this approach is only possible in networks that provide deterministic packet latencies (e.g. bufferless NoCs).

Although it would be possible to use both retransmission request packets and acknowledgement packets, it was deliberately decided not to use this approach. The problem with retransmission request packets is that they generate additional traffic in situations in which the network is dropping packets because of high load in order to reduce it. Whereas the concept of timer based retransmission also provides a limitation of the injection rate of the tiles into the network, the request based retransmission resends packets immediately. At high loads of the network this can easily result in a vicious circle. This approach would only be practicable at low loads or with an additional network for the retransmission requests.

After the reception of an acknowledgement packet that confirms a successful transfer, the network interface can delete the corresponding packet from the retransmission buffer. The relation between the originally sent packet and its acknowledgement packet is done by a transfer id that is unique to each transfer of a tile. The transfer id is stored in the

header flits of the packets that belong to a transfer, such as a read access to a memory. The transfer ids only have to be unique for the currently outstanding requests of one tile. Thus, a field of 4 bits for the transfer id in the packet header would still be sufficient for even 10 outstanding requests. Nevertheless, there should be a safety margin between the maximum number of outstanding requests limited by the retransmission buffer or a deliberately introduced sender limit and the maximum number of transfer ids (here 2^4). This avoids the situation that transfer ids of packets that just got acknowledged are immediately used again. This could lead to a problem if a packet is retransmitted shortly before its acknowledgement packet arrives, the acknowledgement packet of the retransmitted packet would then still carry the transfer id of the previous transfer. However, since the transfer id was already assigned to a new transfer, the corresponding acknowledgement performed by the network interface would be wrong.

3.2.2 Performance Evaluation

This section presents the evaluation of the sdNoC in terms of performance. The evaluation is again performed with the OMNeT++ [90, 91] based simulation model, which was also used in the evaluation of the progressive deadlock recovery concept. The exploration shows the dimensioning of key parameters that were introduced above, such as the discard threshold, the retransmission period, the retransmission buffer size, and others. The focus is again on traffic scenarios that contain message dependencies in order to evaluate the ability of sdNoC to solve the problem of message dependent deadlocks. In addition, sdNoC is also evaluated with other basic traffic scenarios that are widely used in the literature. Like the progressive deadlock recovery concept for NoCs, sdNoC is also compared to a NoC that implements strict ordering by virtual channels.

3.2.2.1 Simulation Setup

The basic system setup is the same as in the evaluation of the progressive deadlock recovery concept described in Section 3.1.2.1. The networks use a 2D mesh topology ($n \times n$) and the default mapping of the four memories realizing the message dependencies is again at the middle of the borders (see Figure 3.8). This mapping is similar to the one in [37]. All simulated networks employ wormhole forwarding and XY-routing. If not stated otherwise, the size of the networks is 8×8. The router architecture is in-

and output buffered. The size of the input queues is set to 3 flits, modeling a two flit input queue plus the register stage between the port determination functionality and the switch unit (see Figure 2.1). The output buffer size is set to 2 flits. This queue configuration is denominated i3o2 in the following. If queue sizes differ from these values in one of the simulations it is explicitly stated. In terms of timings and forwarding, the router models the same behavior as the router implemented in Verilog, that is described and evaluated in Section 4.

The implementation of the sdNoC network interface in the simulation model is quite similar to the architecture shown in Figure 3.32 on the right. As stated before, this is not the preferred architecture for the realization of sdNoC in a real system. However, the different implementation of the model does not make any difference for the determination of the possible throughput of the sdNoC concept. In addition, the tiles are only modeled as traffic generators with a network interface in the simulator. Processor cores, local tile interconnects and memories are not modeled. The retransmission buffer is organized in queues of 10 flits of which each can store one packet. In the model, the queues of the retransmission buffer serve as output queues at the same time. In addition, the network interface of the sdNoC based system has a 10 flit input queue. The model of the network interface that is used for the network employing strict ordering has an input and an output queue for each virtual channel. All of these queues are also 10 flits in size.

The denomination of the compared networks is similar to the one used in the performance evaluation of the progressive deadlock recovery concept:

- Sd: NoC applying the **sd**NoC concept.
- So: Denominates the NoC applying the **s**trict **o**rdering concept implemented by virtual channels.
- i*nom*: Suffix for the two previous network labels that specifies the queue sizes of the router. n specifies the size of the **i**nput queue in flits (including the additional register stage), m is correspondingly the size of the **o**utput queue.

The traffic scenario applied to the network using strict ordering is the same as the one described in Section 3.1.2.1 in the evaluation of the progressive deadlock recovery

concept. The traffic used in the evaluation of the sdNoC based system is of course basically also the same. However, it adds acknowledge packets to the inter processor traffic that are required by the sdNoC concept. The following list, will only shortly repeat the basic facts and will focus on the differences of the traffic in the sdNoC:

- Memory access traffic: Read request packets of 3 flits in size are sent by the processors to the memories. The memories return 10 flit long reply packets back to the sender of the request. These replies implicitly acknowledge the read request packet and thus complete the transfer. If not stated differently, the processors uniformly distribute their accesses to the available memories. In most evaluations the read request generation rate is iterated from 0.003 to 0.023 flits per cycle per processor. The read request generation rates are relatively low since 60 processors are accessing 4 memories.

- Inter processor traffic: Five flit packets that are sent from one processor to another one. In the NoC using strict ordering these transfers are completed when the five flit packet is received by its destination. In contrast, in the sdNoC the destinations still have to return a single flit acknowledgement packet to the sender. Only then, the sender can delete the acknowledged packet from the retransmission buffer and use the entry for the next packet.

  The destinations of the inter processor packets are uniformly distributed and their generation rate in each processor is fix. It is set to 0.15 flits/cycle/processor, which is approximately half the throughput of a 8×8 mesh without virtual channels for this type of traffic.

In investigations, in which different traffic scenarios are used, this is clearly stated.

The fault tolerance mechanisms of sdNoC described in Section 3.2.1.1 are not modeled. Accordingly, the compared NoC with strict ordering is also not equipped with any fault tolerance functionality.

The simulation results presented in the following evaluations are averaged over a minimum of 250,000 cycles per simulation run and over 5 runs with different seed values for the random number generators used by the traffic generators. The measurement in the simulations starts only after a minimum of 100,000 cycles in order to enable settling of the simulation.

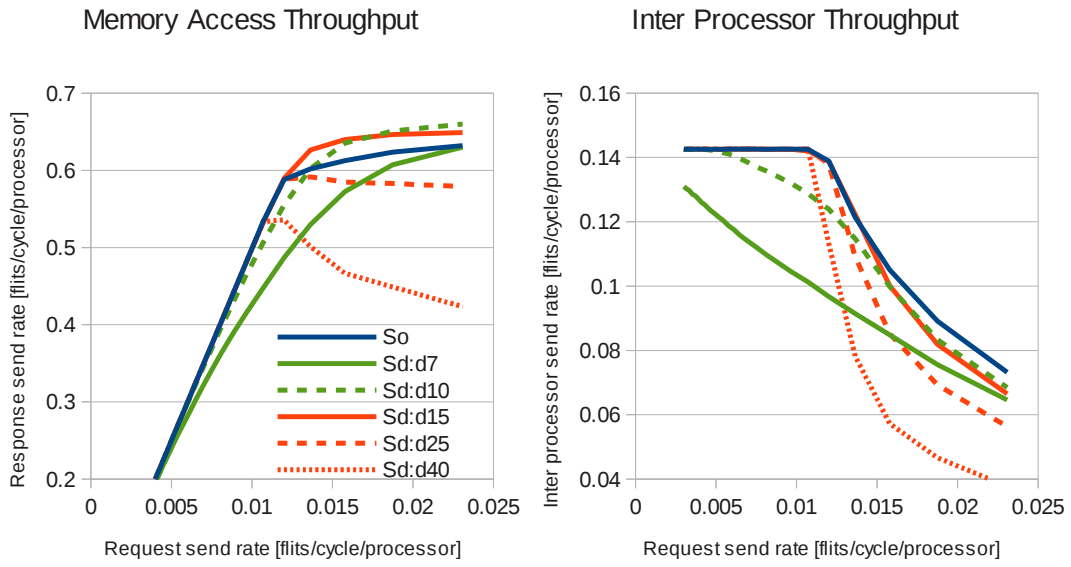

FIGURE 3.33: Effect of discard threshold on memory access throughput (left) and inter processor traffic throughput (right).

3.2.2.2 Discard Threshold

In the first simulations, adequate packet discard thresholds for sdNoC (Sd) should be found and the basic performance of sdNoC presented. The discard thresholds are specified in cycles and added as suffix to the network label, e.g. Sd:dx. The retransmission buffer size is set to 4 packets and the retransmission period is 400 cycles.

Figure 3.33 shows the network throughput of the different traffic classes: the memory throughput diagram (left) and the throughput of the inter processor traffic (right). Both diagrams show that a too low discard threshold of 10 (Sd:d10) or even 7 cycles (Sd:d7) leads to a significant reduction of throughput because of too many discarded packets. This can also be seen in Figure 3.35. The discard rate of the sdNoC with a discard threshold of 7 cycles (Sd:d7) is already above 0.8% at low loads and a sdNoC with a discard threshold of 10 cycles (Sd:d10) still has a quite high rate of dropped packets. Although, the peak memory throughput of this network (Sd:d10) is higher than that of the strict ordering based NoC, this is paid by a squeeze out of the inter processor traffic. The sdNoCs with discards thresholds $\leq 10 cycles$ (Sd:d7 and Sd:d10) are not even able to transfer all the packets at low network loads, as the inter processor throughput diagram shows.

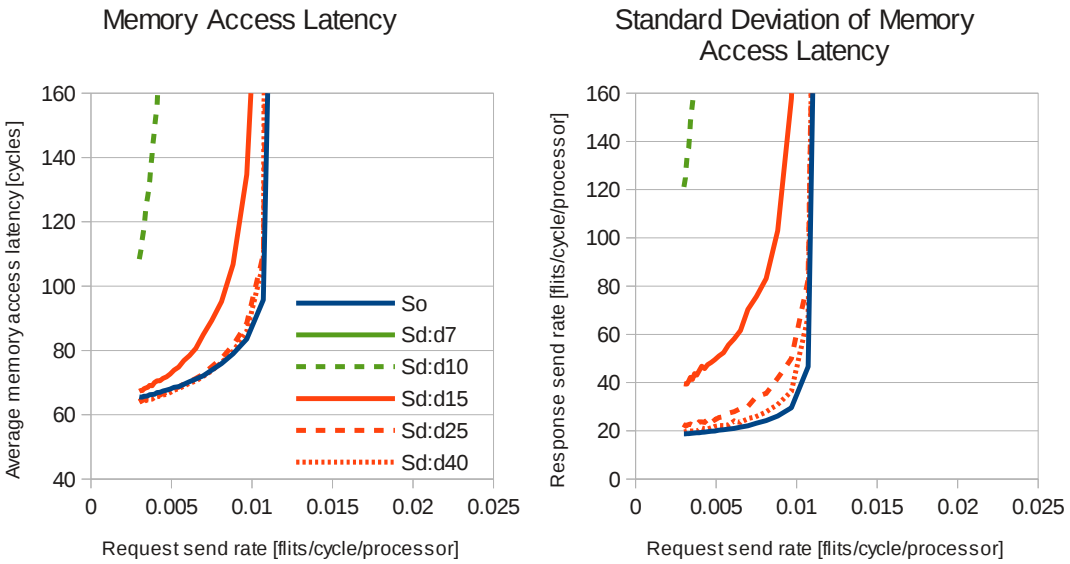

FIGURE 3.34: Effect of discard threshold on average memory latency (left) and its standard deviation (right).

The best throughput is provided for a discard threshold of 15 cycles (Sd:d15). Although the inter processor throughput graph of this network is slightly below that of So in the saturation region, the sdNoC (Sd:d15) is outperforming the strict ordering based network with this discard threshold in terms of memory throughput. The share of dropped packets of the sdNoC with a discard threshold of 15 cycles (Sd:d15) is below 0.5% in the non-saturated region (see Figure 3.35). By increasing the drop threshold, the share of discarded packets can be further decreased. However, the throughput diagrams show that this also leads to a reduction of the throughput. Using high discard timings prevents the solution of temporary congestions, and also delays the solution of message dependent deadlocks. For very high thresholds, like 40 cycles, the throughput behavior of sdNoC (see Figure 3.33 left) resembles the throughput of the progressive deadlock recovery scheme (see Section 3.1.2.3).

The effect of the discard thresholds on the packet latencies is completely different than that on the throughput, as can be seen in the diagrams in Figure 3.34. The left diagram of this Figure shows that with increasing discard thresholds the average packet latency decreases. This is due to the lower share of discarded packets, that do not have to be retransmitted and thus have lower latencies. The sdNoC based networks with thresholds higher than 25 cycles are even able to provide approximately the same average packet

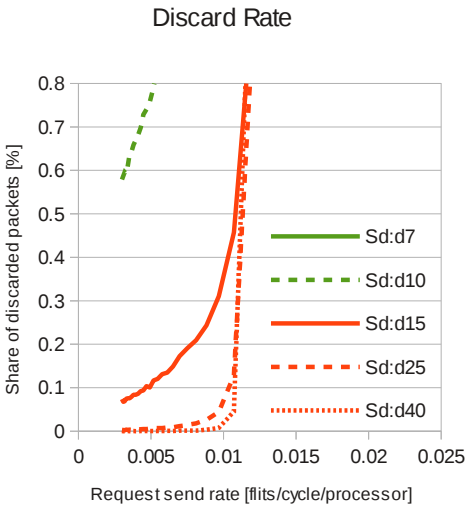

FIGURE 3.35: Discard rate of packets for different discard thresholds.

latency as strict ordering. However, as shown above, this has to be paid by a lower throughput. The standard deviation of the packet latencies shown in Figure 3.34 (right) underlies the same effects as the average packet latencies. The less packets that are discarded and do no have to be retransmitted, the less the variability of the latency.

Figure 3.36 shows the drop map of the sdNoC with 15 cycles discard threshold (Sd:d15) for different request generation rates. Thus, the discard rate is different in the three maps, which can also be seen by the different scales of the color codes. The drop maps show the number of packets that each router dropped during the simulation. At 30% of maximum memory send rate, the network load is dominated by the uniformly distributed inter processor traffic. Correspondingly, the distribution of the discarded packets is quite similar to the link load of a network that transfers uniformly distributed traffic. At 86% memory load, this pattern can still be recognized but it is super-positioned by a dominating pattern created by the memory traffic. This horizontal pattern is generated by the response packets from the memories that first travel in the horizontal direction according to XY-routing. This results in a higher share of other packets getting dropped along this way. At 99% of the maximum memory send rate when the network is already overloaded the pattern changes again. Again most packets are dropped at the hot-spots, but now there is also a vertical pattern. This is caused by the read requests that arrive at the memories by taking the Y-direction at last (according to XY-routing). Since the

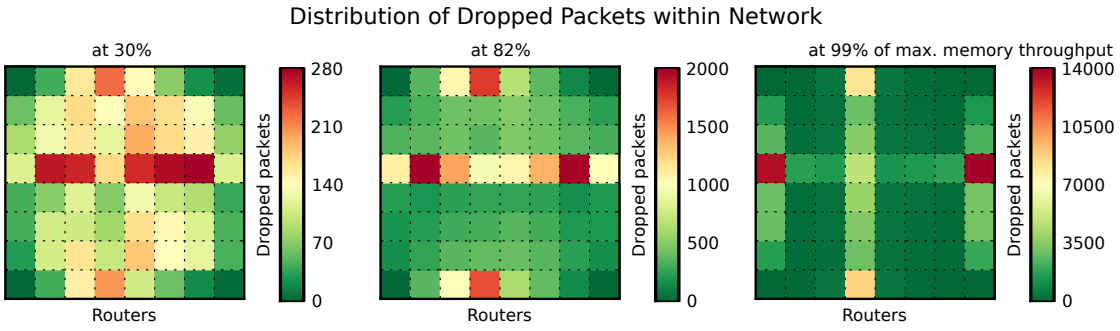

FIGURE 3.36: Distribution of dropped packets within network Sd:d15 for different load values 30%, 82% and 99% of maximum memory injection rate.

memories are already at their limit, the requests queue up along this path and a part of them is discarded.

The optimum discard threshold strongly is tied to the length of the packets transferred in the network. In the scenarios modeled here, the majority of the transferred packets are 5 flits long. The optimum threshold of 15 cycles thus equals the waiting time of a packet if it looses the contention against three other 5 flit packets. This seems a reasonable value, since a packets should not be dropped because it lost a contention against one packet. Due to this close relation of the packet size and the drop threshold, the size of the packets in the network should not vary too much. Assuming there would be some 30 flit long packets in the networks simulated here, all packets that loose the arbitration against those would be discarded. In consequence, longer messages should be partitioned into smaller packets. As already shown in the evaluation of the progressive deadlock recovery concept in Section 3.1.2.7 shorter packets also help to reduce the deadlock probability.

Compared to the deadlock detection timings of the progressive deadlock recovery concept, the discard timings are significantly shorter. In the progressive recovery concept, the redirections had to be restricted to real deadlocks only, due to the limited bandwidth of the recovery channel. However, in sdNoC such restrictions no longer exist and it was deliberately chosen to discard also packets that are stuck only temporarily in order to resolve temporary blockages.

These simulation results show that actively dropping a small number of packets, to solve deadlocks and temporary congestions, can even increase the effective throughput. With only half the allocated buffer space in the routers, sdNoC can achieve equal performance as the network based on strict ordering, assuming adequately set parameters. The profit of the two virtual channels in strict ordering is limited, since the network cannot freely

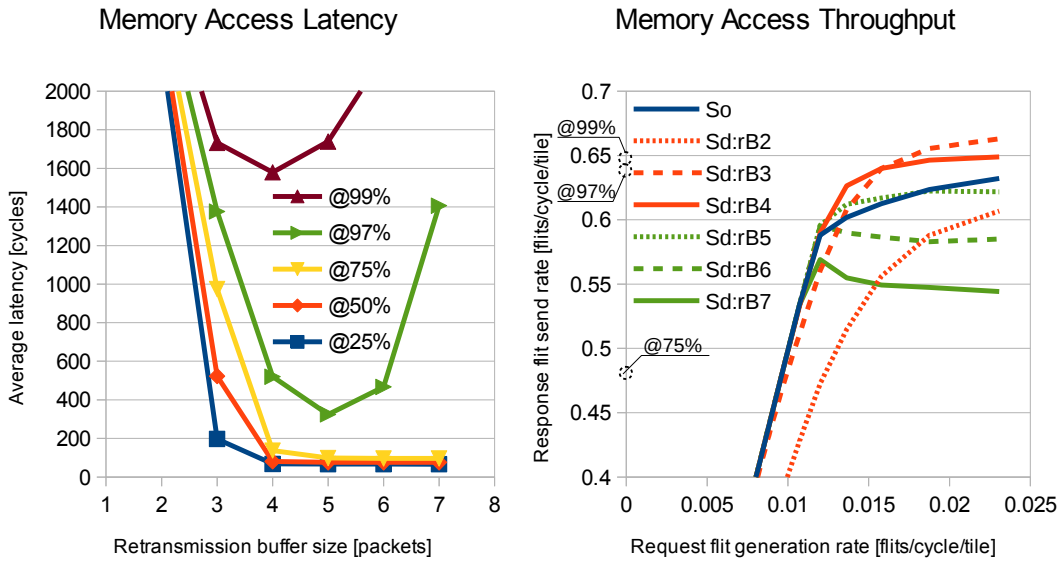

FIGURE 3.37: Average memory access latency (left) and memory access throughput (right) for different retransmission buffer sizes.

assign them to the packets. Instead, the virtual channels are statically assigned to the corresponding message types (requests and responses).

The simulations also show that the discard threshold underlies a trade-off between higher throughput and lower latencies. Within a certain range, lower thresholds provide better throughput, higher thresholds better average latency and also lower latency variability.

3.2.2.3 Retransmission Buffer Size

The size of the retransmission buffer not only affects the performance of the network, but also the cost. Depending on the realization of the retransmission functionality, it either has to be paid by a higher share of reserved space in the tile's local memory or by a larger network adapter in terms of chip area. If the size of the retransmission buffer is too small the network interface has to wait for acknowledges of already sent packets, before it can send new packets. Thus, the send rate of the tile is limited by the size of the retransmission buffer. However, if all tiles in a system are actively sending packets and the network interfaces are able to send too many packets, then the network will be overloaded and the effective throughput of the network will drop significantly. Therefore, an adequate size has to be found for the corresponding system by simulation.

For the exploration of the retransmission buffer size, the discard threshold is set to 15 cycles and the retransmission period to 400 cycles. The graphs in Figure 3.37 showing the performance of sdNoC are labeled by Sd:rBx, with x specifying the size of the retransmission buffer in packets. The network based on strict ordering is labeled So in the diagrams. The latency and throughput diagrams of this Figure show that there is no optimum retransmission buffer size in regard to throughput and latency. The memory throughput diagram on the right of Figure 3.37 shows the best throughput for a retransmission buffer size of 4 packets. Although the sdNoC with a 3 packet retransmission buffer (Sd:rB3) reaches a higher peak throughput, this has to be paid by very high latencies even at low network loads as the left diagram shows. This diagram shows the average latency of the memory accesses for different retransmission buffer sizes and at different memory response send rates. The response send rates are given in % of the maximum send rate of the sdNoC with a retransmission buffer of 4 packets. For illustration, three of these load values (75%, 97%, 99%) are also shown in Figure 3.37 on the right. The left diagram of this figure shows basically two things:

- The retransmission buffer must have a minimum size of at least 4 packets. Otherwise the latency is already very high (and the throughput limited) at low generation rates. This is because the network interface cannot send new packets as the retransmission buffer is full and has to wait for transfers to be acknowledged.
- If all tiles are actively injecting packets into the network and their generation rates are high, a too large retransmission buffer becomes a disadvantage. A smaller retransmission buffer can have a limiting effect on the injection rate and thus can avoid overload of the network. In contrast, larger retransmission buffers can lead to too high injection rates resulting in an overloaded network, if send rates are not limited by additional measures.

The simulations show that for the modeled scenarios a retransmission buffer size of 4 or 5 packets is a good choice.

3.2.2.4 Retransmission Period

Like the discard criterion, the retransmission is also timer based in sdNoc. Therefore, an adequate value for the resend or retransmission period has to be found. In these

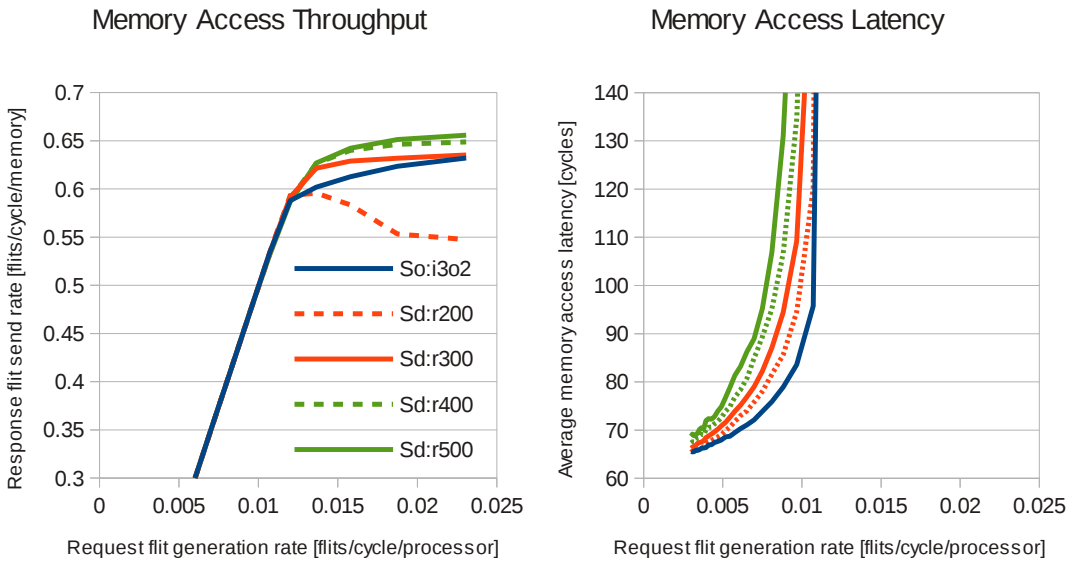

FIGURE 3.38: Memory access throughput (left) and average latency of memory accesses (right) for different retransmission periods.

simulations, the discard criterion is set to 15 cycles and the retransmission buffer is set to a size of 4 packets. The sdNoCs with different retransmission periods are labeled Sd:rx with x being the resend period in cycles. So denotes the network based on strict ordering.

Figure 3.38 shows the memory throughput and the memory access latency for the competing network architectures. The left diagram presenting the send rate per memory tile shows that high retransmission periods allow to achieve higher throughputs, while relatively short resend periods (200 cycles) lead to a drop of the send rate at high load. This behavior is expected, since short resend periods provoke high network loads by already resending packets, although the original packet or the acknowledge packet were only shortly delayed. However, by early resending the packets the network load goes even further up, thus all packets get delayed even more. In this way a vicious circle is created, that leads to a drop of the effective network throughput, like it can be seen in the graph of the sdNoC with a retransmission period of 200 cycles (Sd:r200). In contrast to this, high retransmission periods enable higher throughputs, since they rather limit the injection rates of the tiles than injecting additional traffic by early retransmission. Correspondingly, the sdNoC with a resend period of 500 cycles (Sd:r500) shows the best

behavior in terms of throughput, even better than the network using strict ordering (So).

The right diagram of Figure 3.38 depicting the average memory access times shows a completely different situation. Higher resend periods result in higher average packet latencies and also higher variability, since after a packet has been dropped it takes more time until it is retransmitted. While high retransmission periods deliver the best throughput values (e.g. Sd:r500), the packet latencies of sdNoCs with such retransmission periods are the worst among the simulated networks. The latency values of the sdNoC with a resend period of only 200 cycles (Sd:200) are quite close to these of strict ordering based network, however the short retransmission time prevents better network throughput.

All in all, choosing an adequate resend period is a trade-off between latency and throughput. Lower resend periods provide lower packet latencies and less variability of latencies, higher resend periods enable better throughput. For the following simulations it was chosen to resend packets after 400 cycles if they have not been acknowledged.

3.2.2.5 Localized Memory Access

In this section the behavior of sdNoC in the presence of localized memory access traffic in comparison to strict ordering will be investigated. In the previous simulations the memory accesses of the CPU tiles were uniformly distributed among the four available memories at the borders. Now, the memory accesses become localized, this means CPU tiles send a higher share of read requests to the memories that are nearer to them. How strongly they prefer nearer memories depends on the localization factor. The higher this factor, the higher the share of requests to the memories with the lower network distance. The exact formula for calculating these values is given in Section 3.1.2.5 in which the same evaluation was performed for the progressive deadlock recovery concept.

In contrast to the memory access traffic, the inter processor traffic remains uniformly distributed. The two competing network architectures, sdNoC (Sd) and strict ordering (So), are both simulated with traffic scenarios of different localization factors. The different simulations are labeled with the postfix loc$x$ with $x = 0, 10, 100$ representing

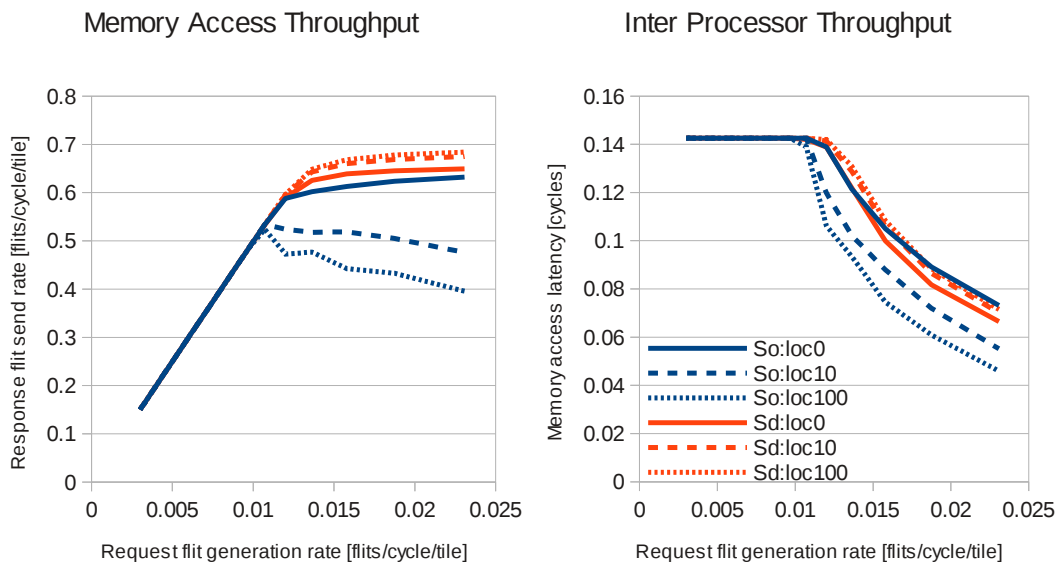

FIGURE 3.39: Memory response send rate (left) and inter-processor traffic send rates (right) for different degrees of localization of the memory accesses.

the localization factor in the shown diagrams. The discard threshold is 15 cycles, the retransmission buffer 4 packets and the resend period is 400 cycles.

Figure 3.39 shows the memory throughput (left) and the inter processor traffic throughput (right). As we already saw in Section 3.1.2.5 in the investigation of the progressive deadlock recovery concept, the strict ordering based networks cannot profit from a localization in terms of throughput. This is due to an increased load of the routers around the memories, as the memory access traffic is no longer as equally distributed in the network as before. Nevertheless, both diagrams of Figure 3.40 show that in networks implementing strict ordering the average packet latency and its standard deviation are reduced with increasing localization. This can be easily explained by decreasing network distances of the communication partners due to the rising localization.

SdNoC profits from the localization of the memory access traffic in terms of throughput and latency. At a localization factor of 100, sdNoC is even able to outperform the network using strict ordering (So) which allocates double buffer space in the network, in terms of memory and inter processor throughput. While sdNoC cannot outperform strict ordering in regard to average latencies or their standard deviation, it still profits from more local accesses in this field too. Also the share of redirected packets decreases minimally with increasing localization of the memory accesses. This is a consequence

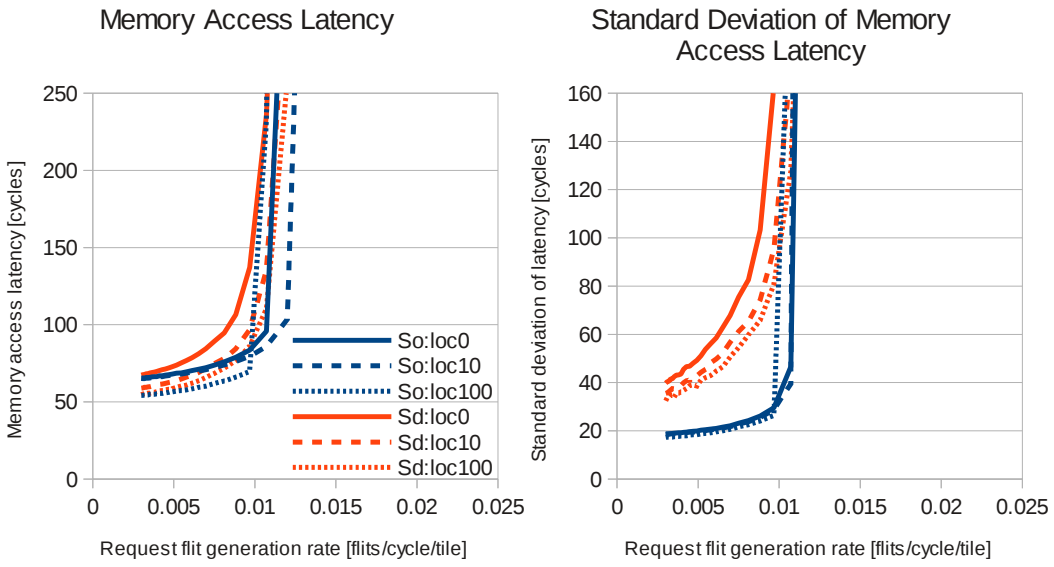

FIGURE 3.40: Average memory access latency (left) and its standard deviation (right) for different degrees of localization of the memory accesses.

of the reduced occurrence rate of message dependent deadlocks (as we already saw in the investigation of progressive deadlock recovery) and a decreased network load due to the reduced network distance of the memory accesses. These two effects outweigh the increased load of the routers directly around the memories.

3.2.2.6 Evaluation of Router Queue Sizes

In all simulations above, sdNoC was only equipped with half the buffer space allocated in the routers than the routers realizing strict ordering. Nevertheless, sdNoC was able to show equal or even better performance in terms of throughput. This sub section will explore how sdNoC can profit from increasing router queue length, and how it performs in comparison to strict ordering with equal amounts of allocated buffers in the network.

The different router queue sizes are labeled with the postfix imon to the network identifiers (Sd,So) in the diagrams, with m specifying the queue length of the input queue in flits and n the output queue length. m includes the additional flit register between routing unit and switch unit in the router hardware model (see Section 4.1.2). The parameters of sdNoC are set as follows: discard threshold 15 cycles, retransmission buffer 4 packets, resend period 400 cycles.

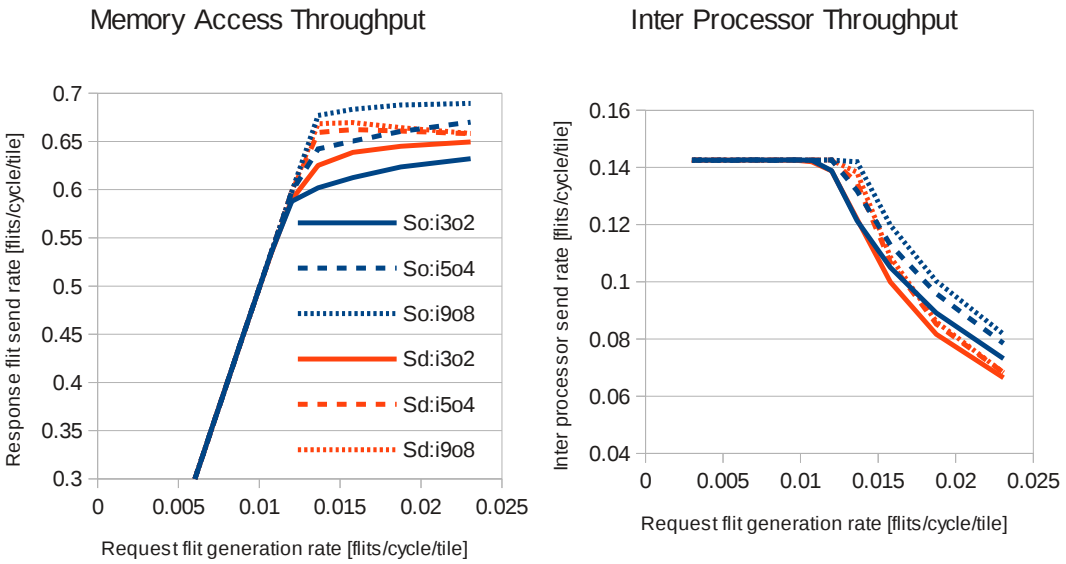

FIGURE 3.41: Memory response send rate (left) and inter-processor traffic send rates (right) for different router queue sizes and uniformly distributed memory access traffic.

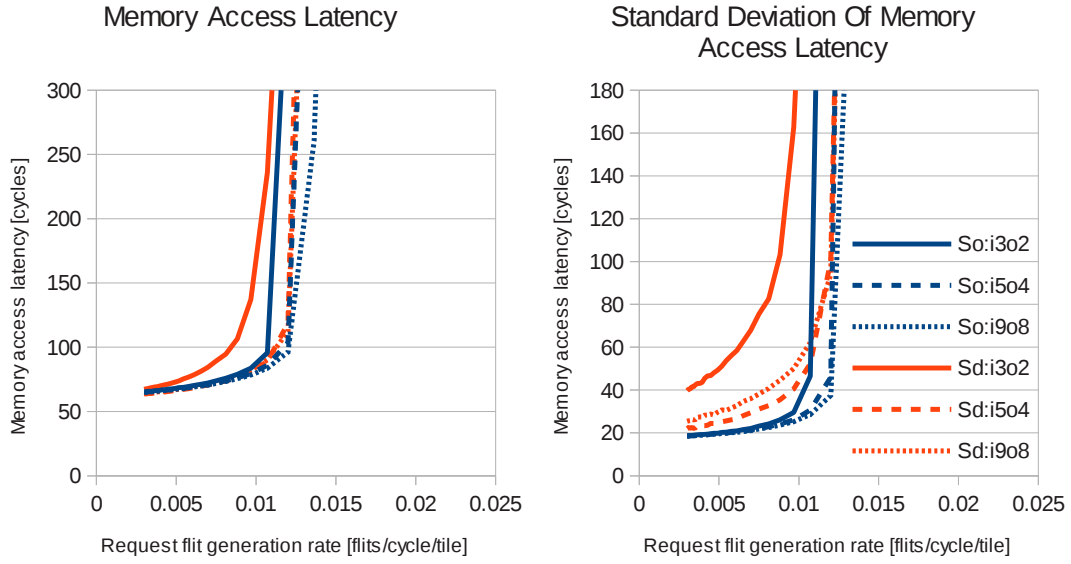

FIGURE 3.42: Average memory access latency (left) and its standard deviation (right) for different router queue sizes and uniformly distributed memory access traffic.

TABLE 3.1: Comparison of total router buffer space for different networks.

Total router buffer space [flits]	**Strict Ordering (So)**	**SdNoC (Sd)**
20	–	Sd:i3o2
40	So:i3o2	Sd:i5o4
80	So:i5o4	Sd:i9o8
160	So:i9o8	–

Table 3.1 shows the total buffer space requirements per router for each of the two concepts. This table also shows for which router queue lengths sdNoC and strict ordering have equal buffer space requirements: For example, So:i3o2 and Sd:i5o4 have the same router or network buffer requirements (not considering the network adapters). Detailed investigations in terms of total buffer space and chip area requirements are given in Section 3.2.3 and 4.3.

Figures 3.41 and 3.42 show the simulation results of the networks based on sdNoC and strict ordering (Sd and So) without localization of the memory access traffic. SdNoC is only able to profit significantly (5%) from the first increase of the router queue lengths from 2 flits to 4 flits (Sd:i5o4). Figure 3.42 also shows a major improvement in terms of average latency and its standard deviation. The average packet latencies are equal to strict ordering. The further increase of the router queues to a length of 8 flits (Sd:i9o8) provides only minimal additional throughput and improvement in terms of latency for the sdNoC. The strict ordering based network is able to increase its effective memory throughput by approximately 6% and 4% with the increase of queue lengths to 4 flits (So:i5o4) and to 8 flits (So:i9o8), as we also saw in the investigation of progressive deadlock recovery concept. This increase in throughput can also be seen in the diagram showing the inter processor throughput, in which the inter processor traffic is only squeezed out at higher request generation rates.

Figure 3.43 shows the same simulations, with the exception of localized memory access traffic (localization factor 100, see Section 3.1.2.5 for further details on localization). Also with localized memory traffic, sdNoC can neither profit significantly from the increase of the router queue length. Nevertheless, in this case it significantly outperforms strict ordering in terms of network throughput independent of the router queue length.

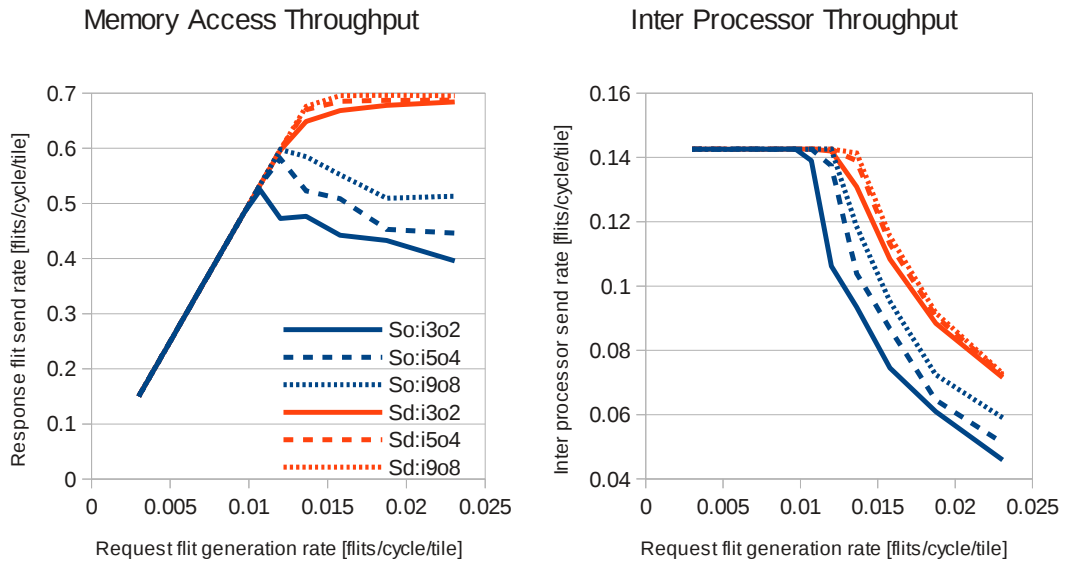

FIGURE 3.43: Memory response send rate (left) and inter-processor traffic send rates (right) for different router queue sizes and localized memory access traffic.

3.2.2.7 Adaptive Discard Threshold

Even if applications are not latency sensitive it is desirable that packets do not have to be retransmitted too many times until they reach their destination, even if this only applies to very few packets. Figure 3.44 shows the distribution of the required retransmissions for a sdNoC with a discard threshold of 15 cycles (Sd:d15), as it was simulated above at a memory send rate of 60% of its maximum. In the simulations here, approximately 2 million packets were transferred. Approximately 40 thousand packets had to be retransmitted once ($r = 1$,cut off in the figure). The occurrence of the higher retransmissions ($r \geq 2$) are shown in the figure.

As described in Section 3.2.1.1, the probability to drop packets that have already been retransmitted can be reduced by increasing their discard threshold. In the networks evaluated for this section, (Sd:d15ax) this is done by increasing the discard threshold by x% multiplied by the number of retransmissions r of a packet. The discard threshold in the given simulations is then calculated as follows: $dt = 15 + 15 * x * r$.

Figure 3.44 shows the results of four sdNoC with adaptive discard threshold (Sd:d15ax with $x = 25\%, 50\%, 75\%, 100\%$). While the discard rate stays constant (not shown), the occurrence of retransmission $r = 2$ goes down from ca. 1400 for $x = 0$ (Sd:d15) to ca.

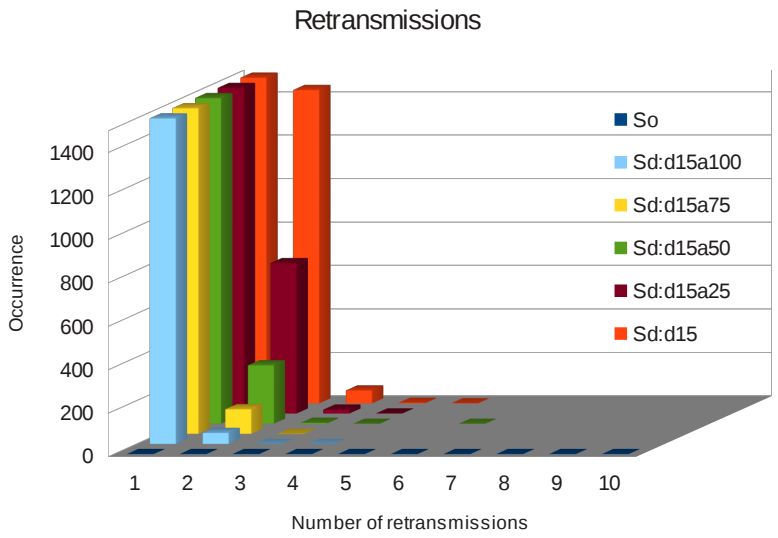

FIGURE 3.44: Distribution of the required retransmissions dependent on the adaptation of the discard threshold. Occurrence of 1 retransmission is cut off, since values are around 40,000.

110 for $x = 100$ (Sd:d15a100). The occurrence of $r > 1$ decreases with rising x. As the discard rate does not change, the occurrence of one retransmission ($r = 1$) goes slightly up. This cannot be seen in the figure, since the occurrence of one retransmission is slightly above 40 thousand packets, for 2 million transferred packets in this system.

However, neither the average packet latency nor its standard deviation change noticeably with this adaptation. The reason is simply that the occurrence of $r > 1$ is without adaptation of the threshold already very low in comparison to the occurrence of $r = 1$. Thus, reducing occurrence of $r > 1$ does have almost no influence on the latency metrics.

3.2.2.8 Virtual Channels in sdNoC

Section 3.2.2.6 investigated the performance of sdNoC in case additional buffer space is allocated in the network by extending the router queue lengths. However, additional buffer space could also be invested into realizing virtual channels in the sdNoC routers. In contrast to just extending the router queue lengths, this leads to more complexity, since more arbiters and a more complex switch unit are required. In addition, the link interface has to support the according signaling for virtual channels. Independent of these additional costs, the advantage of virtual channels in a sdNoC compared to strict

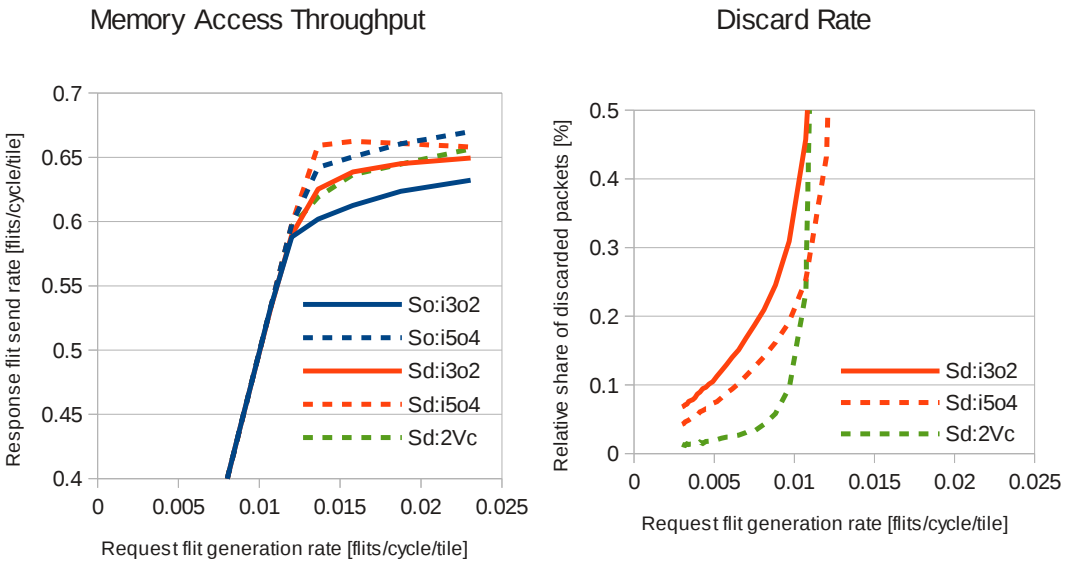

FIGURE 3.45: Memory response send rate (left) and share of discarded packets (right) for different router queue sizes and numbers of virtual channels.

ordering would be, that the channels can be freely used by all packets and are not statically assigned to message types.

However, Figure 3.45 shows that this additional effort is not paid back by better performance of the network. The sdNoC with 2 virtual channels (Sd:2Vc) is only able to provide the same throughput as the sdNoC without virtual channels (Sd:i3o2) that allocates only half the buffer space in the network. The sdNoC with 2 virtual channels (Sd:2Vc) is only able to provide lower average packet latencies and less latency variability (both not shown in diagrams) compared to the standard sdNoC (Sd:i3o2). Nevertheless, the sdNoC with 2 virtual channels cannot compete with the sdNoC that allocates the same buffer space in the network by extending the router queues (Sd:i5o4). The sdNoC with extended router queues (Sd:i5o4) provides better throughput, better latency values and its router architecture is less complex compared to the one with virtual channels (Sd:2Vc).

The reason why sdNoC does not profit from virtual channels as it profits from an extension of the queue length (up to a certain extent, see Section 3.2.2.6) is that the timer based discard criterion does not work well with virtual channels. In a network with virtual channels, packets can advance alternately on the different channels, which leads to less temporary congestions. Then, the criterion for the discard of a packet only becomes

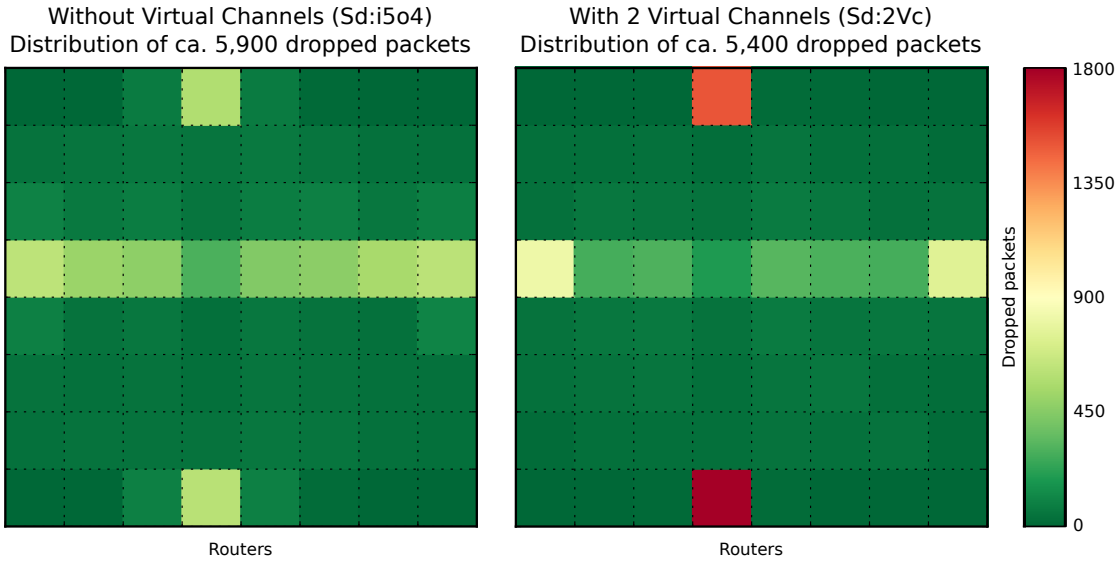

FIGURE 3.46: Distribution of dropped packets for sdNoC without virtual channels (Sd:i5o4) and sdNoC with two virtual channels Sd:2Vc.

true in case of message dependent deadlocks and of blockages directly at the hot-spots of the network. Otherwise the criterion is fulfilled only rarely. Here, the hot-spots are the memory tiles.

Figure 3.46 depicting the drop map of the sdNoC with 2 virtual channels (Sd:2Vc) on the right, shows exactly this behavior. The drop map depicts all routers of the network as colored blocks and the color is chosen according to the number of packets that were dropped by this router. Both drop maps are taken at memory injection rates of 82% of their maximum rate of 0.65 flits/cycle/tile. In both networks, the same number of packets was transferred within the same time. The sdNoC with extended router queues (Sd:i5o4) also drops most of its packets near the hot-spots (left drop map), nevertheless the drop map is more evenly distributed than that of the sdNoC with virtual channels (Sd:2Vc) on the right. The sdNoC with extended router queues (Sd:i5o4) even drops 10% more pockets than the one with 2 VCs (Sd:2Vc). Shorter discard timings are neither able to improve the performance of the sdNoC with virtual channels.

3.2.2.9 Increased Memory Bandwidth

In the following simulations, the theoretical memory bandwidth is increased by implementing 4 memory tiles at each border of the network. This memory mapping equals the mapping of Tilera's Tile64 processor [37]. The changed system architecture is denoted

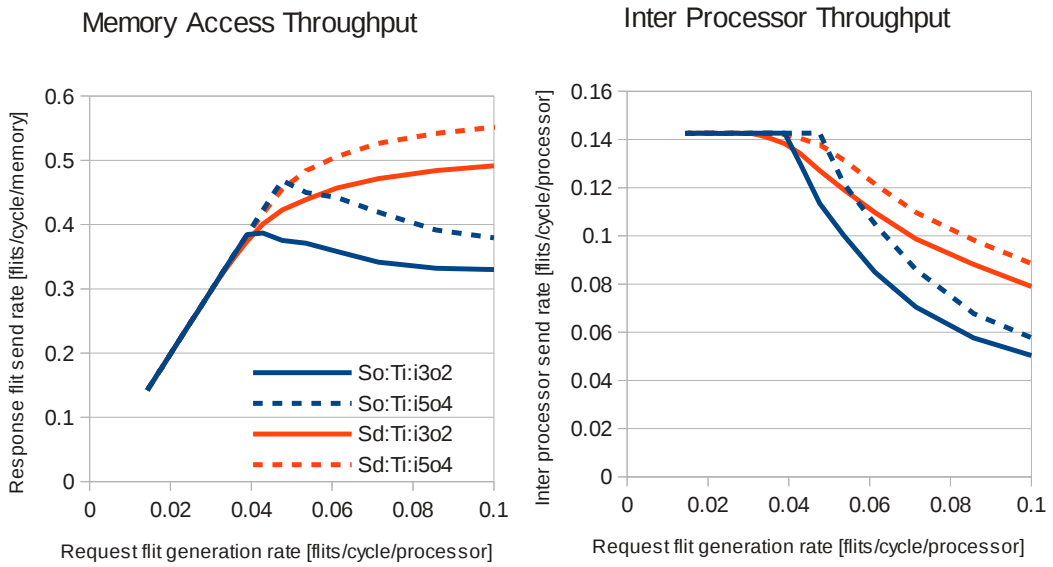

FIGURE 3.47: Memory response send rate (left) and inter-processor traffic send rates (right) for different router queue sizes and increased memory bandwidth.

in the network identifiers with the label Ti, e.g. So:Ti:i3o2. Now the system consists no longer of 60 CPU tiles and 4 memories, but of 48 CPU tiles and 16 memories. In consequence, the compute tiles will be able to send a significantly higher rate of read requests to the memories, as in the simulations before. The memory access traffic is localized with a factor of 100 (see Section 3.1.2.5 for definition).

SdNoC can compete with strict ordering in terms of memory throughput with half the buffer space in the network as the left diagram in Figure 3.47 shows. However, this is only possible since the inter processor traffic is squeezed out earlier than in the NoC based on strict ordering (see Figure 3.47 right). The latency diagrams in Figure 3.48 also show that in this system configuration, the sdNoC (Sd:Ti:i5o4) has to allocate the same buffer space in the network as strict ordering (So:Ti:i3o2) in order to be able to compete. Then, sdNoC suffers just from a higher latency variability at higher loads. The reason for this change in performance relative to strict ordering in the previous simulations is, that with higher memory bandwidth, the network is increasingly becoming the bottleneck.

For these simulations the retransmission buffer size of the sdNoC is set to 4 packets, and the resend period is again 400 cycles. However, the discard criterion is changed to 30 cycles, in contrast to the 15 cycles that were applied in the simulations before. This results in a better throughput of the network and significantly better latency values. The

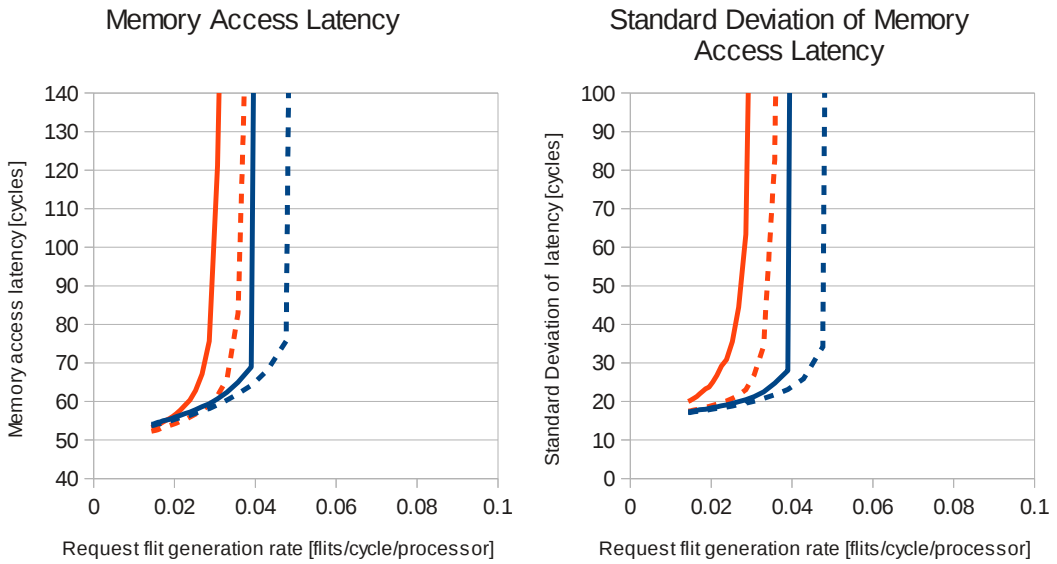

FIGURE 3.48: Average memory access latency (left) and its standard deviation (right) for different router queue sizes and increased memory bandwidth.

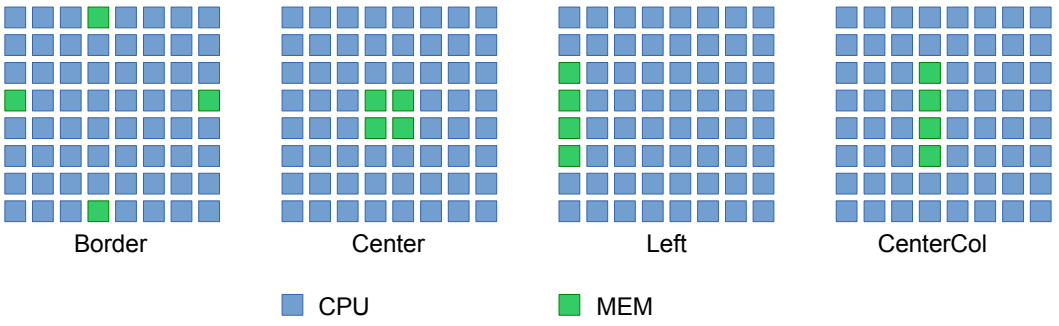

FIGURE 3.49: Overview of the evaluated memory mappings.

reason for the required change of the discard threshold is the adapted average length of the packets in the network. With the theoretically quadrupled memory bandwidth the share of response packets, that are 10 flits in size has also increased. This shows that the adequate discard threshold is closely related to the average packet length in the network. This also means that the maximum length of packets should not be too large and that the length should not differ too much between the packets.

3.2.2.10 Evaluation of Memory Mappings

Although sdNoC is much less sensitive to the occurrence of message dependent deadlocks than the progressive deadlock recovery for NoCs (see Section 3.1.2.9 for the equivalent investigation), its performance dependent on the mapping of the memories should also be investigated. The simulated systems contain 4 memory tiles and the memory access traffic is uniformly distributed, i.e. it is not localized. The applied traffic scenario is not changed. Inter processor traffic serving as background traffic and memory access traffic are applied to the network. The mappings shown in Figure 3.49 are investigated, not only for sdNoC but also for strict ordering:

- Border: Like in all the previous simulations of sdNoC's investigation, the memories are mapped in the middle of the four borders of the network (also see Figure 3.6 left).
- Center: The four memories are mapped in form of a square in the middle of the network (also see Figure 3.7 left).
- CenterCol: The memory tiles are placed in the middle of a column at the center of the network (also see Figure 3.7 right).
- Left: All memories are mapped in the middle of the network's left border.

The retransmission buffer size of the sdNoCs is again 4 packets and the discard threshold is 15 cycles. For the mappings Border and Left the resend period is 400 cycles, for the mappings Center and CenterCol it is decreased to 300 cycles due to the reduced average network distance of the communication partners.

Figure 3.50 shows the memory access throughput and the average memory access latency for the different networks and mappings. The two diagrams of this figure show that sdNoC provides approximately a 5% higher memory injection rate than strict ordering, if corresponding mappings are compared. Apart from this, both networks show basically the same dependence of throughput and latency on the mappings. Placing the memory tiles at the border or at the left provides higher throughput than putting them in the center. These mappings distribute the memory access traffic better over the network. While the mapping Center draws additional memory traffic to the center, where the

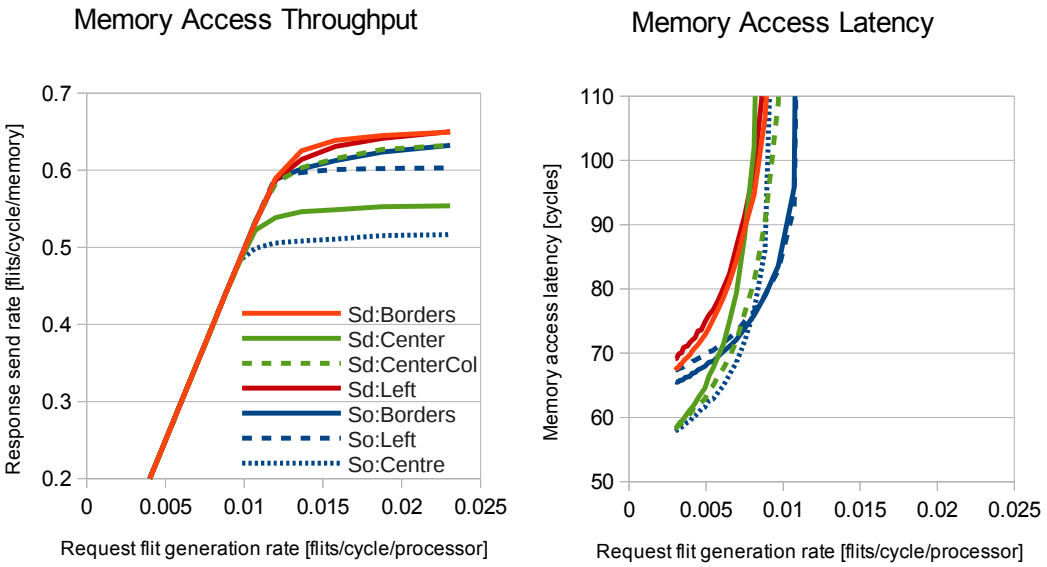

FIGURE 3.50: Memory response send rate (left) and average memory access latency (right) in dependence of different memory mappings.

TABLE 3.2: Best values of sdNoC parameters for the different traffic patterns.

Traffic Pattern	**Retransmission Buffer Size [packets]**	**Resend Period [cycles]**
Uniform	5	400
Transpose	4	300
Neighbor	7	300
Tornado	4	300

background traffic already creates a higher load due to the XY-routing. Correspondingly, the inter processor traffic throughput (not shown) of these mappings also suffers from this.

In terms of latency (see 3.50 right) the situation is the opposite. Placing the memories in the center reduces the total average network distance of all CPU tiles to the memories. Thus, these mappings lead to significantly lower (15% less) latency values at low network load.

3.2.2.11 Pseudo Synthetic Traffic Patterns

The focus of the investigations shown above was the performance of sdNoC in the presence of message dependent deadlocks. However, the fault tolerance mechanisms

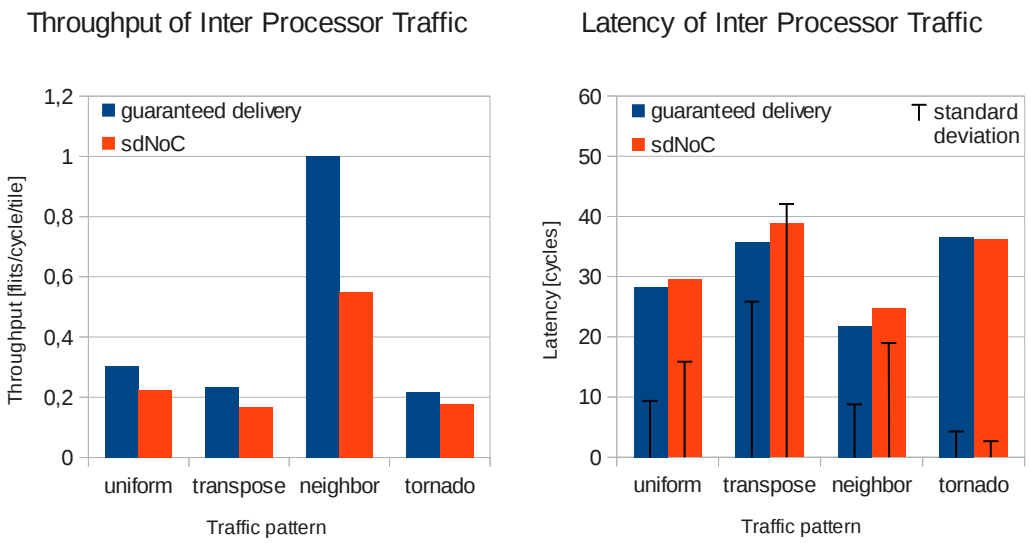

FIGURE 3.51: Average tile throughput (left) and average latency with its standard deviation (right) for different traffic scenarios without message dependencies and deadlocks.

of sdNoC, the end-to-end retransmission can also be used in systems without different message types, that are thus without message dependent deadlocks. In the following simulations, the sdNoC concept will be explored for pseudo synthetic traffic patterns, that are popular in the NoC community to compare networks. These directed traffic patterns are deduced from traffic that is caused by matrix operations or fluid dynamics simulations in parallel processors [19]. The following traffic patterns are applied to the modeled networks: transpose, neighbor and tornado traffic. In addition, an undirected traffic scenario with uniform traffic is also simulated. Since there are no message dependencies, sdNoC is compared to a standard guaranteed delivery NoC without virtual channels. The networks consist of 64 CPU tiles and are organized in a 8×8 mesh. The input buffers of the routers are set to 3 flits in size and the output buffer to 2 flits. The packet length is 5 flits again, as the inter processor traffic in the simulations above.

All sdNoCs are again modeled with a discard threshold of 15 cycles. The best values for the two other parameters, retransmission buffer size and resend period, are specific to the different traffic patterns. They are given in Table 3.2. These values were determined by exploration.

Figure 3.51 (left) shows the maximum injection rate per tile of the two network architectures and the different traffic patterns. The throughput of sdNoC is 20% to 40% lower than that of the guaranteed delivery NoC. This is due to the additional acknowledgement traffic which is exactly 20% of the original traffic, if retransmissions are neglected. For each successfully transfered 5 flit packet a one flit acknowledgement is sent in return. The difference in throughput in case of the neighbor traffic is especially noticeable. In the guaranteed delivery NoC there is no contention for link bandwidth from any packets. All paths from all tiles to their destinations do not overlap, so links do not have to be shared. In the sdNoC, this no longer holds true, since the acknowledgement traffic has to share links with the normal traffic.

On the right, Figure 3.51 shows the average packet latency and its standard deviation at a load of 50% of the maximum injection rate. In terms of average latency sdNoC is quite close to the standard NoC. In the sdNoC, the packet latency is measured as the time by which the packet that actually should be transferred reaches its destination. The time for the acknowledgement to reach the source is not taken into account, since the transfer of data from source to destination is the actual target. If the acknowledgement gets dropped and the packet has to be transfered again, this does not change the delay of the first successfully transferred packet, but delays the following packets since they have to wait longer for free space in the retransmission buffer. While the average latency values of the two network architectures only differ slightly, the variability, i.e. the standard deviation, is significantly worse in sdNoC for most traffic patterns.

3.2.3 Evaluation of NoC Wide Buffer Space Requirements

The sdNoC concept allows to save significant amounts of buffer space in the routers, if the system applies traffic with message dependencies on the network or requires fault tolerance to soft errors. Due to sdNoC's ability to solve deadlocks and provide retransmission, costly measures to avoid deadlocks or implement fault tolerance in the routers, are unnecessary. An sdNoC based router requires only half the buffer space than a router implementing strict ordering with 2 virtual channels. It requires even less buffer space than the progressive deadlock recovery concept that was proposed in Section 3.1, since it does not implement an escape channel for deadlocked packets. As depicted in Figure 3.52 (left) the buffer space savings are 50% compared to strict ordering for a system

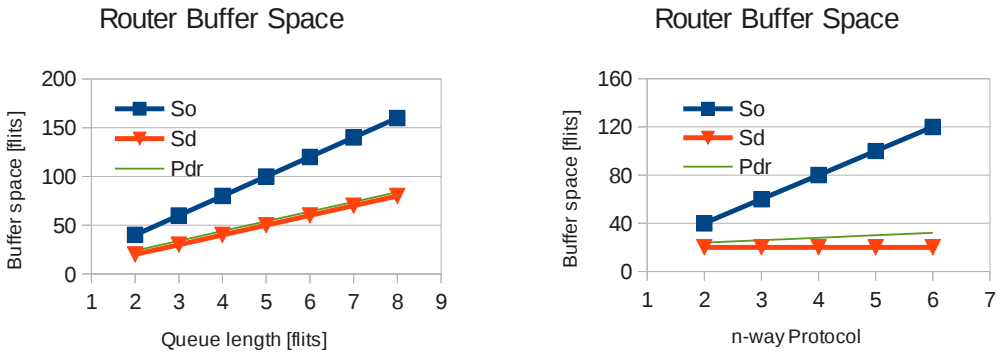

FIGURE 3.52: Buffer space requirement of a 5 port router depending on the queue length for 2 dependent messages (left) and for a varying number of dependent messages and a queue length of 2.

with 2 dependent messages independent of the length of the router's input and output buffers. In addition, the complete router architecture and also its buffer requirements are totally independent of the number of dependent messages in the system (see Figure 3.52 right). In a router based on progressive recovery the number of virtual channels of the deadlock recovery channel has to be adapted. This leads to a slight increase of the buffer requirements with rising n of the communication protocols (n-way protocol). In contrast, the buffer requirements of a router implementing strict ordering with virtual channels rises linearly with n.

Evaluating the total system in terms of buffer space requires a differentiation according to the realization of the retransmission functionality. In case the retransmission buffer is realized in the local memory of the tile, the network adapter just requires an input and an output queue. For the comparisons in Figures 3.53 and 3.54 10 flit queues are assumed. This architecture of the network adapter (Sd:locMem) also saves buffer space in comparison to the network adapter realizing strict ordering, which has to support the virtual channels of the network. Independent of the network size, this type of the sdNoC based system requires the least total buffer space (see Figure 3.53), only half of strict ordering and also significantly less than progressive deadlock recovery (from Section 3.1, labeled Pdr here). Of course, sdNoC's network adapter will also require additional logic to implement the retransmission from the local memory, but this will be evaluated in Section 4.3.

Assuming an implementation of the retransmission buffer directly in the network adapter

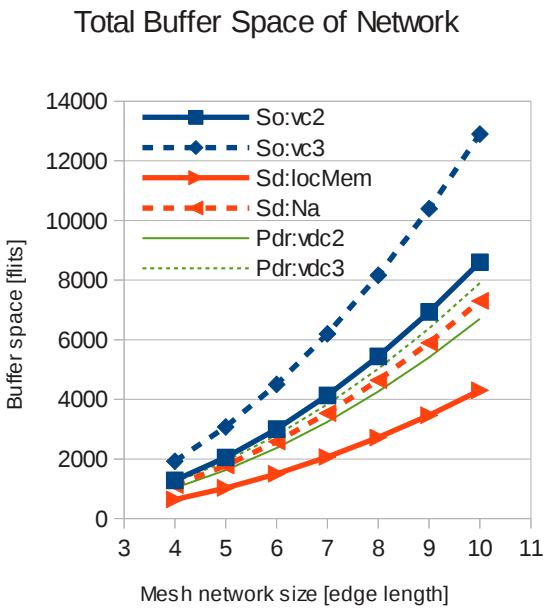

FIGURE 3.53: Comparison of total buffer space requirements for different network architectures and different network sizes.

(Sd:Na) the required buffer space rises. For the evaluation in Figure 3.53 a retransmission buffer of 4 queues each of 10 flits length is assumed. In addition, each network adapter is equipped with an input queue of 10 flits in size. In this case, the total buffer space requirements of the sdNoC based system is still lower than that of strict ordering. However, in case the system only has to support 2-way protocols, the progressive deadlock recovery concepts buffer requirements are slightly lower.

However, it has to be noted that the buffer requirements of strict ordering and the progressive deadlock recovery concept rise with increasing numbers of dependent messages in the system. In contrast, sdNoC's buffer requirements and the architecture of the network adapter and router are independent of the message dependencies present in the system.

Figure 3.54 shows the total buffer count of the systems simulated in the performance evaluation above. This figure compares the following systems for different router queue lengths: strict ordering (So), sdNoC with the local memory (Sd:locMem) and without (Sd:Na), and progressive deadlock recovery for NoCs (Pdr). The sdNoC based system with the retransmission buffer realized in the tile's local memory requires still less buffer

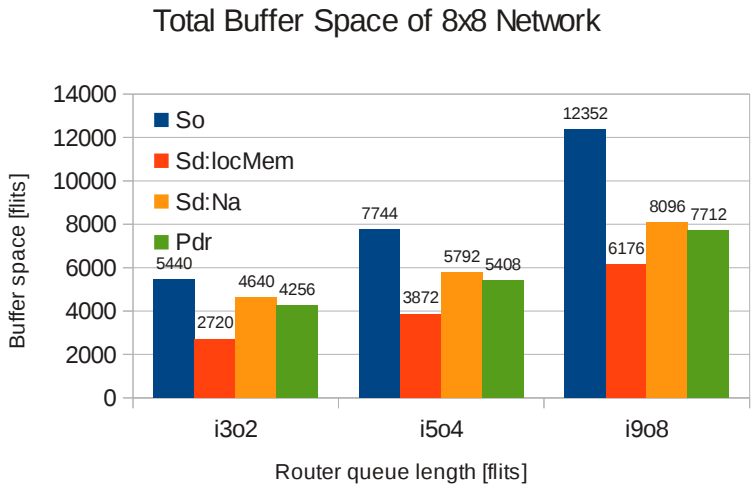

FIGURE 3.54: Comparison of total buffer space requirements for different network architectures and different router queue sizes.

space than a strict ordering based system. Even if the sdNoC's router queues are doubled in terms of length (Sd:locMem:i5o4 vs. So:i3o2). SdNoC's buffer requirements are also significantly less than that of the progressive deadlock recovery concept if the retransmission buffer is implemented in a tile local memory.

In case the retransmission buffer has to be implemented in the network adapter sdNoC requires slightly more buffer space than progressive deadlock recovery for equal router buffer configurations. However, it is still better than strict ordering for equal router buffer configurations and has only 7% more buffers for doubled router queue lengths in sdNoC (Sd:Na:i5o4 vs So:i3o2). These considerations already show very good buffer savings for the sdNoC concept for systems that have to support 2-way protocols. With every additional dependent message in the system the savings of sdNoC increase, since the buffer requirements rise only for the competing concepts.

3.2.4 Summary and General Evaluation

The sdNoC concept proposed in this dissertation provides advantages for NoC based systems on three different levels:

- Functionality: By discarding packets, the network is able to solve all types of deadlocks. Since routing dependent deadlocks can be easily solved without additional

costs, this feature is only applied to message dependent deadlocks. Due to its retransmission capabilities, sdNoC also provides fault tolerance to transient errors corrupting the packet data or even the router functionality. These fault tolerance features will become more and more significant in the future.

- Performance: The simulation results show that selectively discarding packets helps to solve temporary local congestions and thus improves the effective throughput of the network. Depending on the traffic scenarios it is even able to outperform strict ordering based networks that allocate significantly more buffer space. In comparison to the progressive deadlock recovery concept, sdNoC is clearly superior. This is due to the removed restriction of a limited bandwidth of the recovery channel when deadlocks should be resolved.

- Costs: SdNoC enables massive router buffer space savings in the network in comparison to strict ordering. If the network adapters are also taken into account, a differentiation between the implementation possibilities of the retransmission buffer has to be made. Only the implementation of this buffer in a local memory of the tile enables significant buffer savings in comparison to strict ordering. However, since the concept transfers the complexity from the routers into the network adapters, it remains to be seen whether the buffer space savings also translate into area savings (see following chapter).

The simulations showed that the throughput of sdNoC can be improved by extending the router queues. In this way, sdNoC is also highly adaptable as saved buffer space can be traded in for more performance according to the demands of the application. Nevertheless, all these advantages have to be paid by a higher variability of packet latencies and thus also higher worst case latencies.

The simulations also showed that there are no universal values for the parameters of sdNoC (discard threshold, retransmission buffer size, resend period) that provide optimum performance for all systems and applications. Instead, the parameters lead to trade-offs in terms of throughput and transfer latencies of the network. This means, sdNoC's parameters have to be adapted to the network architecture and the application, and the restrictions on the packet sizes have to be considered in order to achieve good network performance.

Chapter 4

Hardware Realization and Cost Evaluation

The evaluation in terms of buffer requirements presented above does not enable an exact comparison of sdNoC and strict ordering in terms of required chip area. This holds especially true, since sdNoC saves a lot of buffer space at the cost of adding complexity to the routers and especially to the network adapters. In order to perform an exact comparison of sdNoC and strict ordering in terms of area cost, a hardware model has to be developed. This hardware model is not used for further performance evaluations, since the OMNeT++ model, which was used for the performance evaluations presented above, is already flit accurate. Since the presented NoCs assume that one flit consists of one word, the OMNeT++ model is also cycle accurate.

The sdNoC hardware implementation presented in the following, is not developed from the scratch. Instead, LISNoC [94], a network-on-chip library providing NoC base components, was used as a basis and the according components were replaced, adapted or extended.

4.1 LISNoC

LISNoC is a modular network-on-chip library written in Verilog. It provides router and network adapter implementations to build wormhole forwarding based, guaranteed delivery NoCs. The architecture of all components is modular and highly configurable.

The basic properties of the router can be set by parameters: For example, the number of ports and virtual channels and the data width. Thus, all kinds of topologies can be built, by adapting these parameters and the routing table of the routers. To connect the tile to the routers, different variants of network adapters are available in LISNoC. The basic network adapter provides just the basic functionality of sending and receiving packets. More advanced adapters are equipped with DMA capabilities or even message passing functionality. For this kind of functionality a local memory has to be allocated in the tile.

The following Sections 4.1.1 to 4.1.3 give a short overview of the basics of the LISNoC library. It presents the network interface, the packet format and the architecture of the routers and the network adapters.

4.1.1 Interfaces and Packet Format

LISNoC's network interface enables the transfer of one flit per cycle over a link between routers or the network adapter. The width of the interface depends on the networks data width which is a parameter of the network and can be set according to the requirements of the application. Besides wires for the flit data, two additional wires are allocated to carry the flit type information. The flit transfer is controlled by a handshaking protocol, implemented by a ready and a valid signal per virtual channel. The ready signal of a virtual channel is controlled by the downstream router and signals the upstream router, in which virtual channels there is space for at least one flit. The valid signal of a virtual channel is set by the upstream router and shows the downstream if a valid flit is lying at the flit data wires and on which virtual channel the flit is transferred. This means, that a transfer occurs if both a ready and a valid signal for a virtual channel are high. Since only one flit can be transferred over a link per cycle, only one valid signal of a link can be high in a cycle on a network interface.

Packets can consist of one or multiple flits. The first flit of a packet, the header flits, contains the source address, the destination address and other fields relevant for the network or the communication between the tiles. The header flit can be followed by an arbitrary number of data flits, up to a specified maximum number. All packets, except single flit packets, are trailed by a tail flit, that represents the last flit of a packet. In case of single flit packets, the header flit also serves as tail flit. The type of a flit is

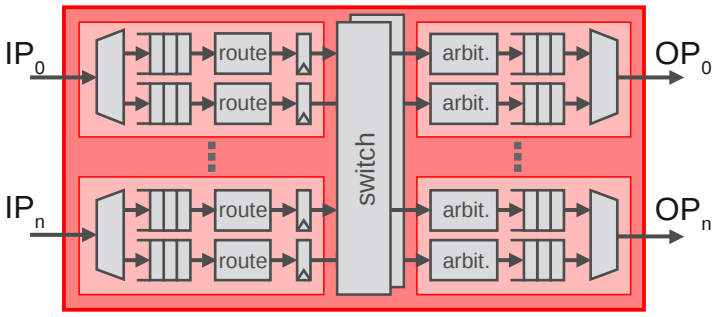

FIGURE 4.1: Architecture of router implementing strict ordering with 2 virtual channels.

specified by a two bit wide type field that is part of every flit in addition to the carried data.

4.1.2 Router Architecture

The basic architecture of LISNoC's router is shown in Figure 4.1. The architecture is based on input and output queues. The number of ports as well as the number of virtual channels can easily be adapted by parameters. The switch depicted in the figure realizes strict ordering, as it just interconnects channels of the same index. The lighter red filled areas mark the input and the output modules. These modules comprise all components required at the receiving or sending side of an input or output port. Thus, the number of ports of the router is easily adaptable by generating the according number of input and output modules.

The input module contains a de-multiplexer, input queues, routing units (route) and flit registers. The last three are allocated per virtual channel. The de-multiplexer is controlled by the valid signal of the upstream router, in order to put the incoming flit into the correct channel queue. The routing unit becomes active in case there is a header flit in the first position of the queue. Then it determines the output port that this packet must be forwarded to. The following flit register separates the routing logic from the following switching and arbitration functionality. Thus, the critical path in the logic is reduced and higher clock frequencies can be achieved.

The output module contains an arbiter, an output queue and an output arbiter. The first two components are again allocated per virtual channel. In case an arbiter is currently

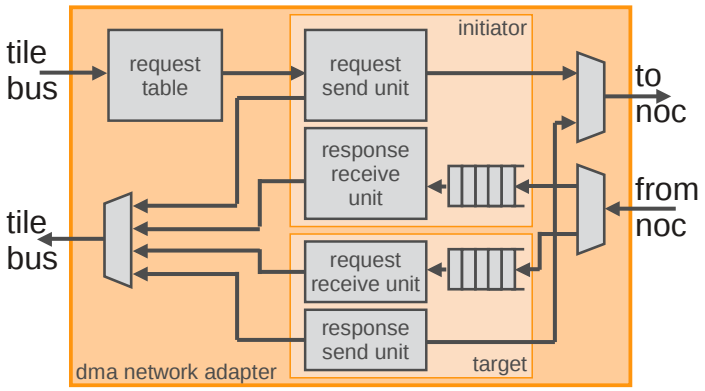

FIGURE 4.2: Architecture of network adapter supporting DMA functionality and strict ordering.

not transferring flits of a packet, it becomes active when header flits are in the flit registers at the input side that are destined to the output channel of this arbiter. In case there are multiple packets, the arbiter decides according to a round robin scheme which packet is forwarded first. The output queue separates this arbitration from the output or link arbitrations and the link delay. The output or link arbiter manages the access of flits in the output queues to the link if virtual channels are implemented. Otherwise, flits of the output queue only have to consider the ready signal of the downstream router when they access the link.

4.1.3 Network Adapter

The architecture of LISNoC's network adapter with DMA support is depicted in Figure 4.2. The network adapter is connected to the tile's local bus by a master and a slave interface. Via the slave interface a processor core of the tile can write DMA transfers into the request table. Table 4.1 shows the contents of this request table. The number of entries of this table has to be chosen so that the processor core does not stall because of a full request table. One entry of this table requires 85 bits. Beside the input queues, this table is one of the main contributors to the buffer requirements of the network adapter.

The initiator module is responsible for handling all transfers that are initiated by this network adapter. Thus, the request send unit generates the read and write requests according to the specifications in the request table and sends them into the network. Read requests can be generated with the data of the request table. In order to fetch

TABLE 4.1: Information stored on a DMA transfer in the request table of LISNoC's network adapter.

Valid	**Direction**	**Remote address**	**Remote tile**	**Size**	**Local address**
0/1	read/write	address in memory of remote tile	tile address	transfer size	address in local memory
1 bit	1 bit	32 bit	5 bit	14 bit	32 bit

the data for a write request from the local memory, the request send unit has access to the bus master interface. It also has a small 3 flit send queue allocated internally. The response receive unit handles all responses originating from requests of the request send unit. It reports the completion of requests to the request table and transfers the arriving data over the bus into the tile local memory. Between the response receive unit and the NoC itself an input queue is connected to receive incoming packets.

The target module is the counterpart to the initiator module. It receives requests from an initiator module of another tile and answers them. The requests are received over the target module's input queue. The data of write requests is stored away by the request receive unit over the local bus master interface. Internally, it is equipped with a 3 flit output queue. Read requests are answered by the response send unit by fetching the data from the tile local memory.

4.2 SdNoC

The following sections describe the adaptations to the standard LISNoC routers and network adapters that were made in order to realize a sdNoC based system. The fault tolerance mechanisms of sdNoC described above, are not implemented in the hardware model, i.e. there is no CRC check in the network adapter. Only, the retransmission capabilities and the packet discard in the network are implemented.

4.2.1 Router

In order to realize a sdNoC router, a standard guaranteed delivery router has to be equipped with the discard capabilities described in the sdNoC concept. First of all, the input and output queues have to be extended by this functionality. These queues

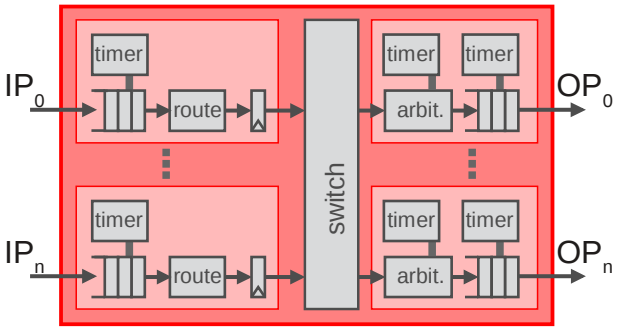

FIGURE 4.3: Architecture of sdNoC router.

are implemented by shift register based FIFOs. These FIFOs require functionality to drop the packet that entered last, from the header flit on. Therefore, the position of the header flit that entered last is stored. If this packet has to be discarded, then the FIFO's write pointer is simply set to the position of this header flit. In addition, a signal is required that stores whether the tail flit of this packet had already entered the queue or not. If so, the discarding of the packet is complete. If not the write pointer is not changed during the arrival of the remaining flits of the packet until the tail flit arrives. All this functionality can be implemented in the control logic of the FIFO by adding more cases to a select case block but without increasing its critical path. Thus, the FIFO with drop capabilities does not suffer from a reduction of its maximum clock frequency compared to the standard FIFO.

Every queue is equipped also with a timer, that can issue the discard command. The timer is started or reset with the entrance of a header flit. It stops counting if there is no header flit in the associated queue. If it reaches its specified discard threshold it is reset, stops counting and gives the drop command to the queue. Compared to the buffer space required for a flit the timers are relatively small. In order to realize a discard threshold of 15 cycles, the timer needs to be 4 bits wide. Even discard thresholds of up to a hundred cycles would not increase the size of the counter above 7 bits.

Besides the input and output queues, the router must also be able to drop packets of which the header flit is currently stored in the flit register between routing unit and switch. Connecting this functionality directly to this register leads to increase of critical paths since the discard of this packet has to be coordinated with the arbiter. Otherwise it could happen that the timer signals a discard to the flit register and the arbiter transfers this flit at the same time. A solution without these problems is to connect the

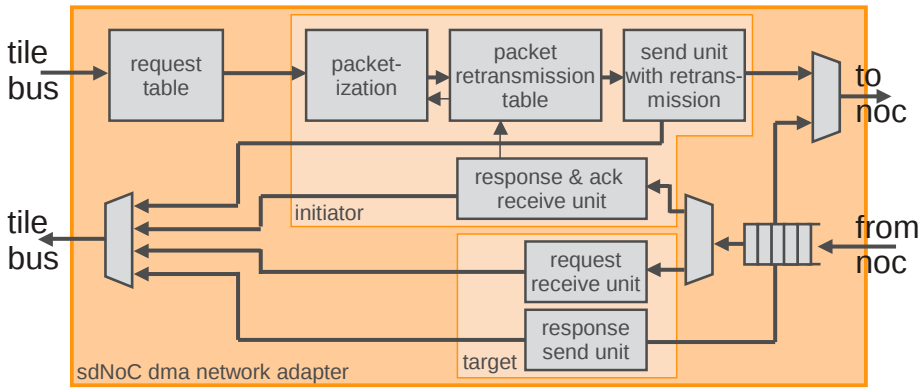

FIGURE 4.4: Architecture of sdNoC's network adapter supporting retransmission from a local memory and DMA functionality.

timer realizing the discard criterion directly to the arbiter, since the arbiter controls the read signal of the flit register. In case a packet should be dropped, the arbiter signals a read to the register but does not set the write signal to the following queue to high. The arbiter only changes the signals after a tail flit arrives or in the next cycle if the packet is a single flit. Thus, the packet gets dropped.

This implementation of the discard capabilities for the flit register again only adds some additional cases to the FSM of the arbiter, but does not increase any critical paths in the logic. Since the queues were also not slowed down by the additional functionality, the complete router is able to operate at the same maximum clock frequency as the standard guaranteed delivery router. In a synthesis done in a 65 nm TSMC low power library the router reached a maximum clock frequency of 1.15 GHz.

4.2.2 Network Adapter

Figure 4.4 shows the architecture of sdNoC's network adapter. The adaptations required to the standard network adapter with support of DMA are limited to the initiator module. The target module, the request table and the bus master interface can stay unchanged. The local character of the adaptations proves easy integration of the sdNoC concept in existing network adapters and enables reuse of existing components.

The packetization module segments the DMA transfers, which can have transfer sizes higher than the maximum packet size of the network, into packets considering the maximum packet size. That means, the specification of a request in the request table is

TABLE 4.2: Information stored in the retransmission table of the sdNoC network adapter.

Valid	**Transfer id**	**Request id**	**Direction**	**Remote address**	**Remote tile**	**Size**	**Local address**
0/1	unique number	id of original request	read/ write	address in memory of remote tile	tile address	transfer size	address in local memory
1 bit	4 bit	2 bit	1 bit	32 bit	5 bit	14 bit	32 bit

partitioned by the packetization module to one (or multiple) packet(s) and written into the retransmission table. If a request requires more packets than the retransmission table can take at once, the packetization module waits and progresses with the partition as the previous packets are acknowledged. Both write and read requests are partitioned equally by the packetization module. Segmentation of read requests means that the requested data is limited to the maximum packets size. Due to this, the DMA target module at the destination does not have to care about packetization and the according retransmission. Instead, the standard DMA target module can be re-used almost untouched. In general, the segmentation process requires the adaptation of the remote and local addresses of the transfers. The size field of the packet specification is set to the maximum packet size. It only has to be adapted for the last packet of a transfer.

The retransmission table administrates the specification of the packets that should be sent out or retransmitted by the send unit of the initiator module. Therefore, it generates the transfer id for the packet specifications written to the table and also stores the id of the request in the request table from which the packet entry originates. With the request id it can report acknowledged packets back to the packetization module, so that completely finished DMA transfers can be labeled as completed in the request table. The number of entries of the retransmission table is equal to the size of the retransmission buffer determined in the simulations above (3.2.2.3). Each entry of this table requires 91 bits to be stored. This is a little less than the size of 3 flits. Thus, storing the specification of a packet requires significantly less buffer than storing a complete packet, which can consist of 10 to 20 flits.

The send unit checks the retransmission table for new entries of packets. In case of a read request, it can generate the read request packet directly from the information of the table. In case of a write request, the according data is fetched from the local memory

over the bus interface. The send unit has a small, internal output queue of 3 flits length. It decouples the work of the send unit from the network to some extent. This means the send unit can already start the generation of the first flits and the bus transfer can already be requested, even if the network can not accept the flits yet. Instead, the flits are put into the output queue first. If the network is still busy after more than 3 cycles, the send unit will need to halt its operation. As soon as both the network interface and the bus are available for transfer of flits, the flits are sent into the network.

Together with the (re)transmission of a packet, the send unit also starts a timer, that triggers the retransmission if it reaches the predefined threshold (resend period) before the packet is not acknowledged. Accordingly, one timer must be allocated for every entry of the retransmission table. To enable retransmission periods of a couple of hundred cycles, as found appropriate in the simulations, each timer has to be 9 bits wide.

The response and acknowledgement receive unit is only slightly extended in comparison to the receive unit of the standard network adapter. It has to be extended to handle the incoming packets in terms of acknowledgements. It takes the transfer id from the the packets in the input queue and reports the acknowledgement to the packet retransmission table. Write acknowledgement packets can then be discarded, whereas, the data of a read response has to be stored in the tile local memory over the local bus. Since this network adapter does not have to implement strict ordering, it does not require two different input buffer queues to support the virtual channels of the network. Figure 4.4 shows that in the sdNoC network adapter the request receive module and the response receive module share one input buffer queue.

In a synthesis done with a 65 nm TSMC low power library both network adapters, the one for sdNoC and the one implementing strict ordering are able to reach a maximum clock frequency of 1 GHz. This means, both, router and network adapter for the sdNoC, do not lead to a reduction of network throughput because of decreased maximum clock frequencies compared to a standard, guaranteed delivery NoC.

4.3 Cost Evaluation

The hardware models described above were synthesized with a 65 nm TSMC low power library in order to enable an exact comparison of the two competing concepts, sdNoC

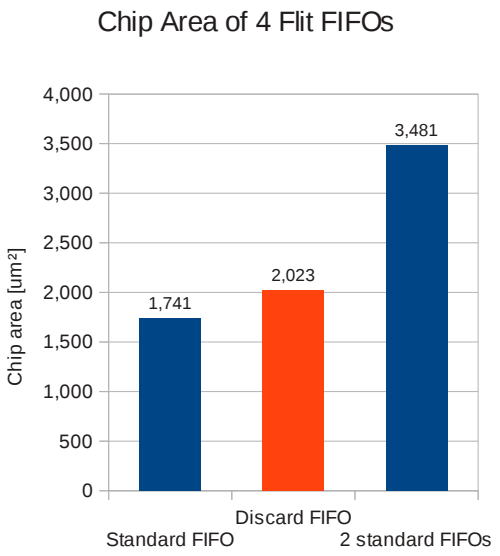

FIGURE 4.5: Chip area requirements of standard FIFO, sdNoC's FIFO and 2 standard FIFOs as allocated in strict ordering.

and strict ordering, in terms of chip area requirements. If not stated otherwise, all results given below originate from the hardware models synthesized for 1 GHz.

Figure 4.5 shows the chip area of the different FIFOs architectures. All FIFOs are based on shift registers, 4 flits long with each flit 34 bits (32 bits data width + 2 bits flit type). The standard FIFO implements just the basic functionality, pushing in flits at the back, taking them out at the front, handling ready and valid signals and managing the corresponding internal read and write pointers. This standard FIFO requires 1,741 μm^2 chip area.

The sdNoC FIFO adds the timer based discard criterion and the packet discard capabilities to the functionality of the standard FIFO. These features add up to 282 μm^2, resulting in a chip area of 2023 μm^2 for the sdNoC FIFO. This is 16% more area than the standard 4 flit FIFO. But, it is 42% less area than two standard FIFOs like they are required per port to realize strict ordering.

The complete sdNoC router with 5 ports, 4 flit input and output queues and a data width of 32 bits (+ 2 bits flit type) requires 24,530 μm^2 of chip area synthesized for 1 GHz (see Figure 4.6). With the same configuration, the router realizing strict ordering has a chip

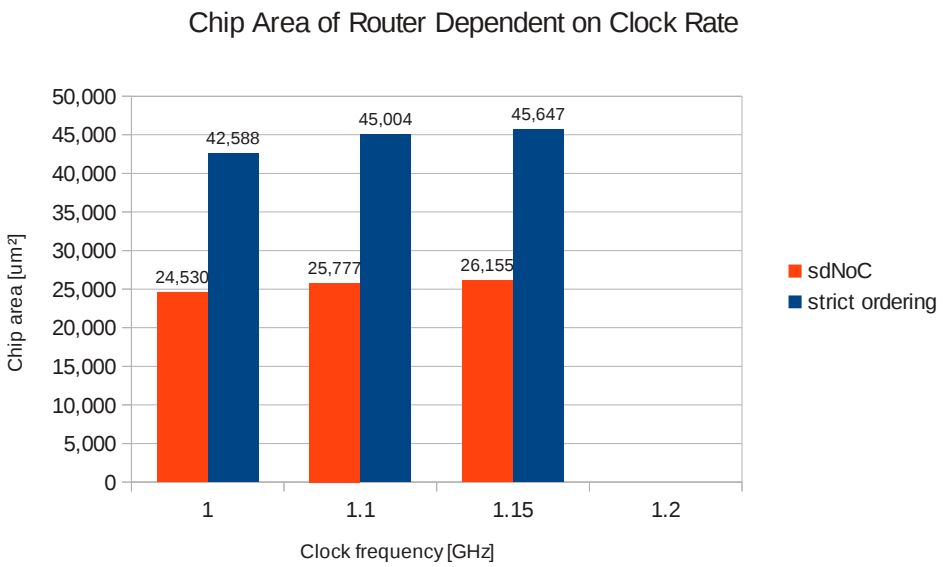

FIGURE 4.6: Chip area requirements of the router architectures for different clock frequencies.

footprint of 42,588 μm^2. Thus, the sdNoC router saves 42% of area in comparison to a router realizing strict ordering.

The absolute area savings of the sdNoC router compared to the router applying strict ordering will be lower, if the length of the buffer queues is reduced to 2 flits, or if only input buffering is applied. Nevertheless, the relative savings will not decrease, since the relative savings of the complete router are even higher than that of a sdNoC FIFO compared to 2 standard FIFOs (44% > 42%). This is caused by all the additional logic that strict ordering requires in a router. Two times the switch fabric of the sdNoC router, the double number of arbiters and in addition link arbiters at the output ports. Figure 4.6 also shows that both router architectures reach a maximum clock frequency of 1.15 GHz. This means that the sdNoC architecture does not extend the critical paths in the router, and that it is valid to perform the simulations of the competing network architectures with equal clock rates.

The cost evaluation of the network adapters assumes a flit width of 34 bits (data width 32 bit), a maximum packet size of 16 flits, and a maximum of 4 entries in the DMA request table. The input buffers instantiated in the NAs can receive a complete maximum sized packet from the network. For the parameters given above, this means one input buffer requires 7,792 μm^2. The complete network adapter implementing strict ordering

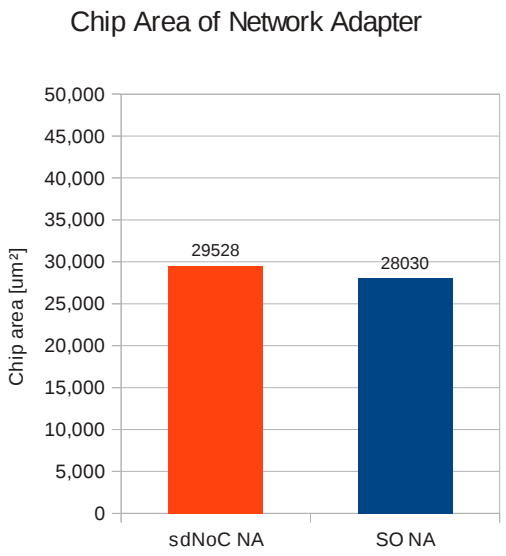

FIGURE 4.7: Comparison of chip area requirements of network adapter architectures.

and DMA capabilities occupies 28,030 μm^2. This includes two input buffers to separate requests and responses according to strict ordering.

The chip area requirements of the sdNoC network adapter are 29,528 μm^2. This is 10% more than that of strict ordering, although the sdNoC NA requires only one packet input buffer. However, this NA has to allocate the packet retransmission table, a table for the transfer ids and the complete retransmission functionality.

Nevertheless, sdNoC saves chip area in comparison to strict ordering. Assuming a direct network topology with one network adapter per router, the area savings are still ca. 16,560 μm^2 per tile. This means an sdNoC based system is able to save 23% chip area in comparison to an equal system using a network based on strict ordering. Note that sdNoC is even able to provide fault tolerance at the same time.

The extensions made to the standard network adapter to support retransmission do not increase any critical paths. The sdNoC network adapter reaches even a slightly higher clock (1.15 GHz) than the network adapter implementing strict ordering (1.1 GHz). In addition to proving the area savings, the synthesis of the hardware model shows that the features of sdNoC do not have to be bought by reduced clock frequency of the involved components. The sdNoC concept does not lead to increased complexity in the architectures of routers and network adapters.

Chapter 5

Multi-Topology NoC Optimized for Shared Resource Access

Many published NoCs [38, 42, 95] or NoC based systems [16, 50] are based on flat and regular topologies. Examples of flat and regular topologies are trees, rings, meshes, etc. Detailed knowledge on all kinds of aspects of NoCs is available for these topologies. For example, routing algorithms, which provide deadlock freedom, exist or also corresponding virtual channel configuration if necessary. In addition, flat topologies can easily be mapped on the chip during later stages of the design process.

The 2D mesh is probably the most popular topology in the field of NoCs. It provides a high degree of connectivity, thus delivering good performance. It can be easily mapped, which is especially true for homogeneous SoCs that consist of equally large processing cores. This thesis proposes an alternative to flat and completely regular topologies: Hybrid, hierarchical NoCs [1, 96]. A hierarchical network can be considered as an interconnection of multiple sub-networks on several hierarchy layers. Sub-networks on the lower levels are interconnected by the networks on the upper hierarchy levels. Hierarchical networks can provide several advantages compared to flat and regular topologies. This applies to homogeneous as well as to heterogeneous SoCs. Hierarchical topologies allow to decrease the maximum network distance of communication partners in comparison to flat topologies. For certain traffic scenarios even the average hop count and the network latency can be reduced. This is especially valid for traffic destined to central

or communication-centric tiles, such as I/Os or memories. These tiles are central to the system, i.e. they are accessed by many of the other cores of the system.

Many applications still rely on shared resources. In addition, traffic between the off- and on-chip domain has to be directed through a limited number of I/O interfaces, since the number of pads of a chip is limited, too. Thus, SoCs will still be built around such communication centric tiles in the medium-term future. In these systems, a high share of traffic needs to be passed through the network interfaces of the central tiles. By connecting these tiles to the upper hierarchy levels of the hierarchical NoCs, as proposed by this thesis, the communication cost of traffic destined to these tiles can be reduced.

5.1 Connecting Shared Resources in Hierarchical NoCs

One of the main targets of NoC research is to reduce communication costs for systems-on-chip. This applies especially to communication latencies, area and power consumption. This thesis proposes the usage of hierarchical network topologies to achieve this aim. Network latencies and hop counts, i.e. the network distance between communication partners, can be reduced, especially for tiles that are central for the corresponding system, and thus are accessed by many other tiles of the system. Such communication centric tiles can be I/O interfaces, shared resources like memories or any other cores that are accessed by many of the system's cores and attract a lot of traffic. For example, an I/O interface to an external memory is usually highly loaded, since internal memory is expensive and thus kept to a minimum. Hierarchical network architectures allow to move these communication centric tiles logically into the center of the network without necessarily being physically in the center of the chip. Hereby, the network distance between the central tiles and all their accessing tiles is reduced.

Many of today's SoC architectures are still built around those centric tiles. Some of the internal, shared resources could possibly be replaced with multiple, distributed resources by using different techniques or architectural approaches. However, this will not be possible with I/O interfaces, as the number of pads of a chip is limited. Thus, the number of I/O interfaces cannot be arbitrarily increased in the future. Although some packaging techniques allow to increase the number of pads of a chip, this is only possible at higher costs per unit.

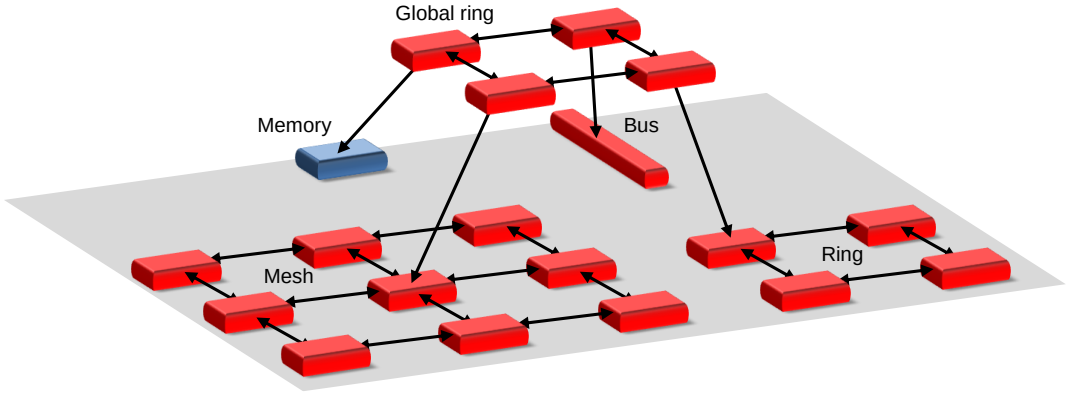

FIGURE 5.1: Hierarchical multi-topology network.

In general, the communication distance of external I/O interfaces or shared resources to all accessing tiles is usually very low in bus or crossbar based SoCs. In case of simple buses or crossbars it is one hop, if hierarchical bus or crossbar architectures are applied the distance can also be 2 or 3 hops. In contrast, the communication distance in NoCs can be much higher. It increases together with the increasing number of interconnected tiles. For example, the maximum hop count in a 8×8 2D mesh is 15 hops. Even tiles mapped to the center of this mesh network, are up to 8 hops away from the other tiles.

This thesis proposes to use hierarchical network topologies in order to efficiently interconnect communication centric tiles with their communication partners. By connecting these tiles, whether these are external interfaces or shared resources, to the upper layers of the hierarchical network, the network distance and thus the communication costs can be reduced for all accessing tiles. Figure 5.1 depicts this kind of interconnection for a memory tile. If the central tiles are accessed by tiles from all sub-networks of the hierarchical NoC, then they should be connected to the network of the highest hierarchy, also called the global network. Hereby, a reduction of the network distance compared to a flat network topology can be achieved.

In addition, a hierarchical network topology provides more freedom in placing the tiles on a chip. This means, that a module connected logically to the center of the network, like it is important for central tiles, can still be placed physically at the border of the chip. In a 2D mesh the logical mapping of the tiles is generally identical to the physical mapping of the system on the chip. This means, a tile that is logically mapped to the center of the NoC will in general also be in the center of the chip. Correspondingly, tiles

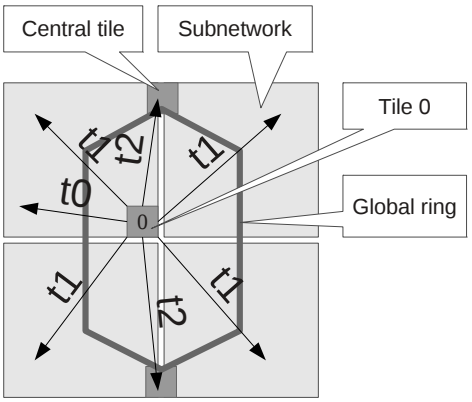

FIGURE 5.2: Classification of traffic in hierarchical network.

that have to be mapped at the border of the chip, will also be mapped to the border of the network. This is especially disadvantageous for communication centric tiles, that have to placed at the border or even in the corner of a chip. Depending on the chip package, this can be the case for external interfaces that require a lot of pads. This is the worst case scenario for a 2D mesh, since the average network distance for a tile mapped in the corner is maximal. A possible solution could be to increase the number of external interface and distribute them around the border of the network and the chip. However, this is only possible if all pads of the chip package are not allocated yet. In addition, the use of chip packaging techniques that allow a higher numbers of pads results in higher costs.

The traffic in a hierarchical network does not only consists of traffic that is addressed to shared resources or external interfaces. Figure 5.2 categorizes the traffic in hierarchical networks into 3 classes:

1. Traffic t0 stays in its sub-network.
2. Traffic t1, also called global traffic, consists of packets that are addressed to any other tile in the network excluding the central tiles.
3. Traffic t2 denotes the traffic going to the communication centric tiles, like I/O interfaces and shared resources.

Hierarchical networks can significantly reduce the maximum hop count for traffic t1 and t2. However, in case of traffic t1 there can also be communicating tiles in a hierarchical

NoC for which the network distance is increased in comparison to a 2D mesh network. Depending on the traffic scenario, this can even lead to an increase of the average hop count in the NoC (see Figure 5.6 left diagram), although the maximum network distance is decreased. In case the networks on the higher layers of the hierarchical network run at a higher clock, the communication latencies can still be reduced.

5.2 Hierarchical Multi-Topology NoCs

Many hierarchical networks use only one kind of topology for all sub-networks on all hierarchy layers [54–56]. The most popular networks of this kind are purely based on the ring topology. Hierarchical ring networks have been proposed for parallel computer networks and are widely adopted in wide area networks (WAN). SDH networks (Synchronous Digital Hierarchy) employed in telecommunication networks are purely built on hierarchical ring topologies.

Hierarchical NoCs that differ from this, follow another strict rule for choosing the topologies on their hierarchy levels, as Section 2.3.2 already showed. On each hierarchy level only one topology is used. For example, the authors of [57] use mesh topologies on the lowest hierarchy level and ring topologies on the upper levels.

This thesis proposes to break with this rule and choose the topology of all sub-networks of a hierarchical layer freely. The topologies should be selected according to the requirements of the application, the cores allocated to the sub-network and the thereof originating traffic. If SoCs are not completely homogeneous, like the Intel SCC [50], this will result in hierarchical, multi-topology NoCs as shown in Figure 5.1.

The concept of hierarchical multi-topology networks provides a high capability to adapt the interconnection infrastructure of a SoC to the application's requirements. In contrast, for a completely regular topology like a 2D mesh the adaptation is limited to the mapping of the cores. In addition, hierarchical, multi-topology NoCs enable the reuse of complete SoC partitions. The legacy interconnects of reused partitions, like buses or crossbars, can also be reused. This avoids the re-implementation of the modules' interfaces of a legacy SoC partition. It also eases the adoption of NoCs as it becomes less costly. This way, an evolutionary path to NoCs is provided. The aspect of employing different types of topologies for the sub-networks is also important for partitions of a

SoC that do not require high communication bandwidth. These partitions can not only profit from reduced area requirements of a bus or crossbar, but also from lower latencies compared to a network.

When choosing the topologies of the sub-networks, the main criteria are throughput and latency capabilities. Table 5.1 gives a rough evaluation of the performance values of different topologies. A higher degree of connectivity generally provides higher network throughput: For example a mesh or a spidergon topology (see [95], ring with diagonal links) has a significantly higher throughput than a ring. This cannot only be determined by simulation, but becomes also obvious when comparing the formulas for the aggregated link bandwidth of a ring (see Equation 5.1) and a mesh (see Equation 5.2). While the aggregated link bandwidth is not identical to the throughput of a network, it can give a first hint when comparing different topologies. A lower network distance, i.e. number of hops, between communication partners generally results in lower communication latencies, especially for crossbars or buses as long as the clock rates are not reduced too much, because of the interconnect's physical size.

$$BW_{agg,ring} = N \cdot BW_{link}, \quad N\text{... number of tiles} \tag{5.1}$$

$$BW_{agg,mesh} = 4(N - \sqrt{N}) \cdot BW_{link}, \quad N\text{... number of tiles} \tag{5.2}$$

In addition, many other aspects have to be considered in the topology selection. For example, the routing algorithm, anti-deadlock strategies, the router architecture, achievable router clocks etc. While ring topologies are very popular in wide area networks, they are not that popular in NoCs. A problem is the low scalability of a simple ring topology. However, it also provides advantages. Due to their low number of ports, the router architecture is very simple. First of all, this enables fast routing decisions. Second, this results in simple arbitration modules at the output ports that solve contentions of the input ports. These architectural aspects of ring routers lead to high clock rates [54, 96] and less chip area per router, as stated in [96]. This can compensate for the low throughput in flits per cycle to some extent. For example, the arbitration functionality of the routers in [96] is implemented by hierarchical round robin stages to achieve high

TABLE 5.1: General evaluation of network topologies.

Topology	**Throughput**	**Latency**
Mesh	+	-
Ring	0	-
Ring	0	+
Ring	-	+

clock speeds. Each stage evaluates only 2 input channels, thus 4 ports require 2 hierarchical stages. As a result the ring routers achieve a 30% higher maximum clock than mesh routers, which require more stages in the hierarchical round robin scheme for the arbitration functionality.

However, the simple network architecture of rings with its low degree of interconnectivity leads to deadlock problems, which cannot be solved by simply adapting the routing function as it is done in mesh networks (e.g. XY-routing). To break the routing cycle in a ring network as shown in [59], virtual channels have to be added. This measure increases the total buffer space in the ring routers, and thus also the chip area requirements. However, the complexity of the routing and arbitration functionality is not increased, since the paths between the virtual channels are not dynamically chosen in the routers. Instead, the paths between the virtual channels are static. Each router routes the virtual channel 0 (1) of the input port to the virtual channel 0 (1) of the output port. There is only one exception, one router in the ring breaks the routing cycle by forwarding virtual channel 0 of the input port to virtual channel 1 of the output port. Due to this, the maximum router clock rates are not limited by these measures.

Many applications show some kind of locality. Assuming an optimized mapping of tiles in the hierarchical network, this results in a high share of local traffic in the sub-networks at the lowest hierarchy. This local traffic does not have to travel to the upper hierarchies. Thus, the low scalability of the ring topology, in comparison to a mesh, becomes a problem when throughput is relevant or many tiles have to be interconnected. In such a case, choosing rings as communication infrastructure on the lowest hierarchy level of the hierarchical NoC is not the best choice. As mentioned above, the low degree of connectivity of a ring and the hereby caused low throughput can be compensated by a higher router clock rate, but only to a certain extent and only if the number of interconnected tiles is not too high. The authors of [54] chose an alternate approach, they reduce the number of interconnected tiles per ring on the lowest hierarchy level.

Correspondingly, they have to increase the number of sub-rings and the number of hierarchy levels to interconnect all the sub-networks. In contrast, this thesis proposes to choose the 2D mesh topology for sub-networks that require high throughput and the interconnection of more tiles than a crossbar can handle.

However, there can also be sub-partitions of a SoC that require less throughput. For example tiles that only have to handle configuration traffic from time to time do not have to be interconnected by a mesh network. Such sub-partitions can be implemented more area efficient with a shared bus or a ring network.

According to the arguments presented above, the hierarchical NoC can be tailored to the application by adapting the sub-networks to the individual requirements of a SoC's sub-partitions. Partitions with many tiles and high throughput requirements can be interconnected by a mesh. If the number of tiles is lower, a crossbar can also be adequate. In case of lower throughput requirements and only few tiles, rings or buses can be a good choice.

Apart from regular topologies, irregular ones exist, too. Such networks are mostly application specific networks, in which the interconnecting links follow the application's data streams very closely. Hereby, one hop connections are provided for communication partners where possible. In addition, each router can be individually adapted to its place in the network. For example, by different numbers of ports, virtual channels, etc. However, such networks are generally only suited for SoCs tailored to only a single application, such as an MPEG dedocder. In contrast, hierarchical, multi-topology networks proposed by this thesis are adapted to the application, but have also regular elements. The sub-networks employ basic, regular topologies enabling the use of the corresponding available knowledge. Existing routing algorithms can be applied as well as well-known anti-deadlock strategies. For example, XY-routing solves the problem of deadlocks in mesh networks very cost efficient. Comparable anti-deadlock strategies exist for all basic topologies and thus can be applied in hierarchical NoCs.

5.3 Realization

Hierarchical networks optimized for shared resource access can be realized in different ways. This thesis proposes the hierarchical network architectures shown in Figures 5.3,

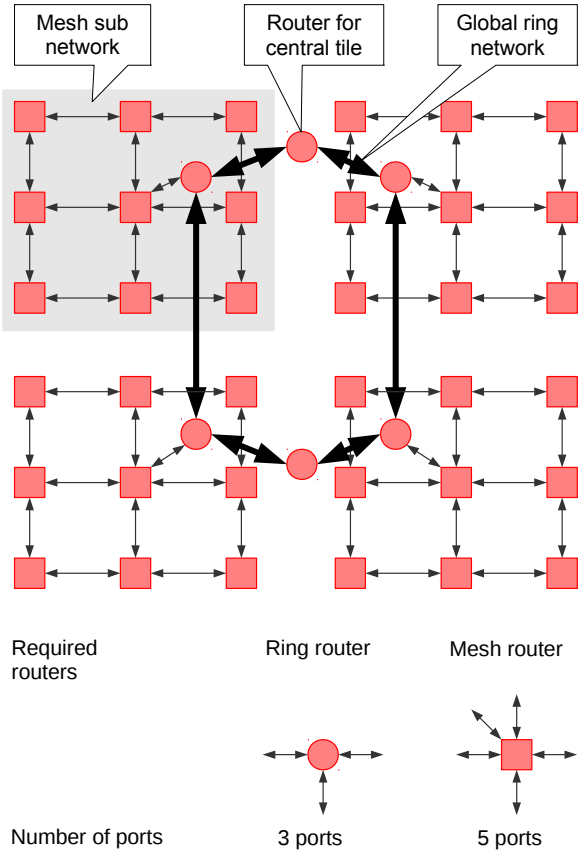

FIGURE 5.3: Hierarchical mesh network with separate ring routers for the global ring and the types of routers used for realization.

5.4 and 5.5 in order to connect central tiles to a system. These architectures are also evaluated in terms of performance in the next section.

The simplest approach of the hierarchical concept uses only standard mesh (5 ports) and ring routers (3 ports). As depicted in Figure 5.3 this results in sub-networks that are all built from its own routers. This means, no router is part of two sub-networks. The bidirectional, global ring network consists of 6 routers, of which 4 are used to connect the sub-networks to the ring and 2 interconnect the central tiles. Similarly, one router of each of the sub-meshes is not connected to a tile, but to the global ring. In the following, this hierarchical network architecture is referred to as hierarchical mesh network with separate ring routers.

An alternative to separate ring routers is to integrate the global ring into the routers of the sub-networks as shown in Figure 5.4. The path of the global ring is thus partially

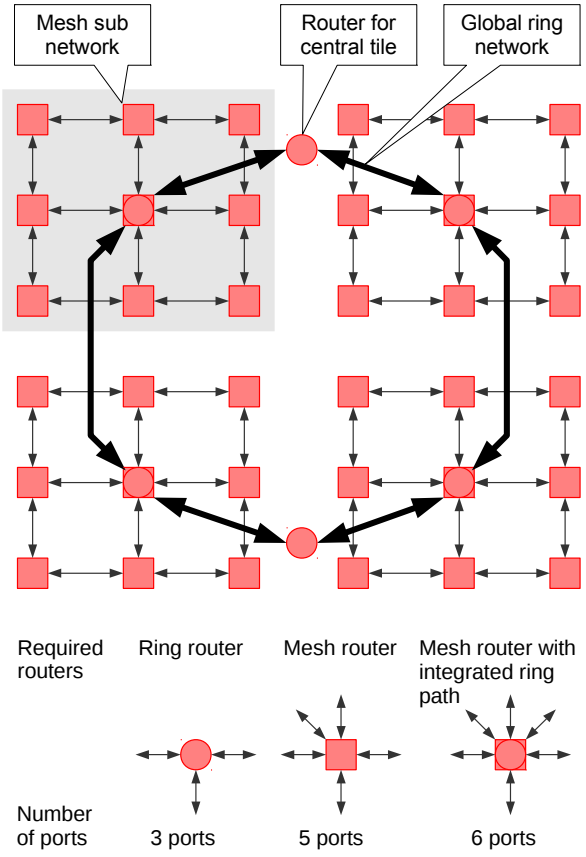

FIGURE 5.4: Hierarchical mesh network with integrated ring routers connecting the sub networks to the global ring and the types of routers used to implement this network architecture.

implemented by additional ports of the corresponding routers of the sub-meshes. Additional, separate routers for the global ring are only required for the interconnection of the central tiles. This architecture reduces the network distance to the central tiles by one and the distance to tiles of other sub-networks by 2 hops. In addition to the 3 and 5 port routers, this hierarchical network requires 6 port routers. This network is referred to as hierarchical mesh with integrated ring routers in the following.

A third way for the implementation of the hierarchical NoCs is shown in Figure 5.5. In this implementation, the network on the lower hierarchy level is not partitioned into sub-networks. Only the global ring on the upper hierarchy level is added to connect the central tiles to the rest of the system. This network architecture is called extended mesh in this thesis.

The performance and cost of a network is not only dependent on the topology, but also

on the network protocols. Thus, an adequate forwarding technique has to be selected for the networks. While slot based forwarding techniques allow higher clock rates due to simpler routing and switching functionalities, they are only well suited for ring topologies. In mesh networks, slot based forwarding can only be employed in combination with connection oriented communication [79]. However, connection oriented communication is only adequate for relatively static communication relations, in contrast to packet switched networks.

A possibility would be the usage of different forwarding protocols in sub-networks of different topologies. This would require the termination of the protocols and conversion of the data structures at the routers interconnecting the different domains. This adds complexity to these routers, leading to costs either in terms of performance or chip area. For example, a ring employing a slot based forwarding is connected to a wormhole based mesh router. On the ring each slot consists of only one or a few flits and the subsequent slots are statically destined to different targets. In case data requiring multiple slots is transferred from a tile of the ring into the mesh network, the router interconnecting the two topologies can either wait to receive all slots, buffer them and then send the complete data in form of one packet into the wormhole based network. This requires additional buffer space and adds latency. As an alternative, the interconnecting router could send each slot as a packet into the sub-mesh, resulting in a high overhead as each packet requires a header flit. These two examples are not the only possibilities for different forwarding technologies and their conversion. However, all of them add complexity to the converting router in one form or another.

Instead of using different forwarding techniques in the sub-networks, this thesis proposes to use wormhole forwarding [12] in the complete hierarchical NoC. Wormhole forwarding can be employed in any network topology. The only exception are buses, which cannot be implemented with wormhole forwarding, assuming a standard bus with standard bus interfaces is employed. In order to connect a bus based sub-partition to a wormhole based network, a gateway for protocol conversion is required. On the network side this gateway behaves like a network adapter, on the other side it has a bus interface. For a burst transfer from the bus to the network, the gateway has to generate a header flit from the information on the address bus send it into the network, and has to buffer the data on the data bus for at least one cycle. In case the network is free, the following bus words can be sent into the network with one cycle delay. In case the progress of the

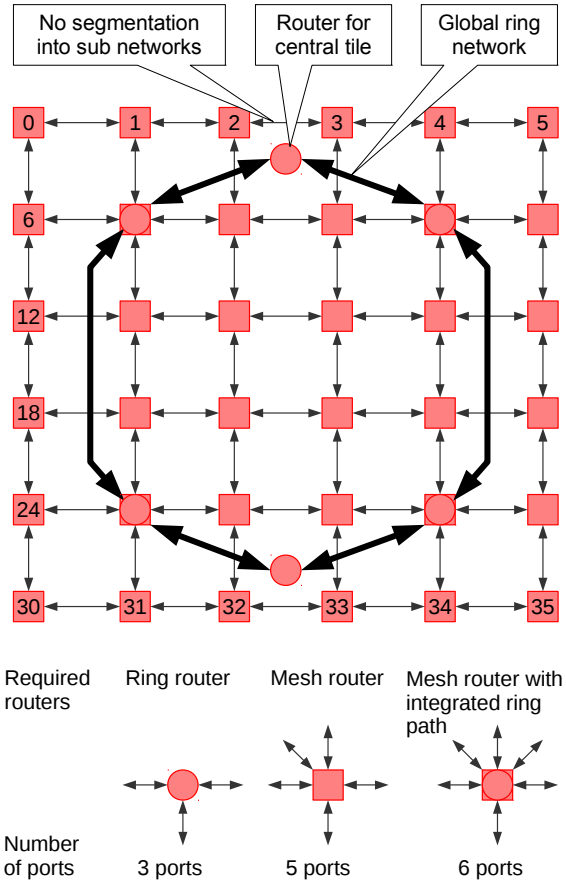

FIGURE 5.5: Hierarchical mesh network without partitioning the network of the lower hierarchy level together with the different kinds of router required for its implementation.

packet in the network is delayed, the gateway has to buffer the subsequent bus words or halt the bus.

Nevertheless, wormhole forwarding is a good choice for hierarchical, multi-topology networks. It provides low packet latencies, especially in comparison to store and forward, and high throughput. At the same time, it has low buffer space requirements and thus keeps the chip area costs low.

5.4 Performance Evaluation

In this section, the hierarchical network architectures that were presented above are evaluated in terms of performance and compared to a standard 2D mesh network architecture. The purpose of the simulations is to quantitatively evaluate the advantages of

hierarchical NoCs in connecting communication centric tiles. The network architectures are simulated with the help of OMNeT++ models already used in the evaluation of the deadlock recovery concepts above. Hop counts, packet latencies and throughput values are provided by the simulations.

5.4.1 Simulation Setup

The following list presents the network architectures that are compared in the simulations given below:

- **mesh**: The standard 2D mesh network consists of 6 × 6 tiles. The central tiles are connected to the routers 2 and 33, according to the numeration of routers in Figure 5.5. Tile 17 and 18, which have the highest network distance from the central tiles, are deactivated in order to achieve the same number of active tiles in all networks. In the hierarchical networks, this results in 32 actively sending tiles.
- **hMsR@2BW**: **H**ierarchical **m**esh with **s**eparate global ring **r**outers operating **at 2** × **b**and**w**idth. The hierarchical mesh consists of 4 sub meshes, with each of them built from 3 × 3 routers, and a global ring network that interconnects the 4 sub networks and the central tiles (see Figure 5.3). The global ring is realized by separate ring routers that operate at double clock frequency compared to the mesh routers . Each sub mesh interconnects only 8 tiles, since the central router of each 3 × 3 sub mesh is required to connect it to the global ring.
- **hMiR**: **H**ierarchical **m**esh with **i**ntegrated ring **r**outers (see Figure 5.4). Basically the same network architecture as the one described above with two exceptions. The routers of the global ring are not separate, but are integrated into the central routers of the sub meshes. Thus saving at least one hop for packets going over the global ring. In addition, all routers run at the same clock frequency.
- **hMiR@2BW**: **H**ierarchical **m**esh with **i**ntegrated ring **r**outers operating **at 2** × **b**and**w**idth. Hierarchical mesh with integrated ring routers like the network described above, except that the all router that are part of the global ring run at double clock frequency compared to the remaining mesh routers (see Figure 5.4).

- **eMesh**: **E**xtended **mesh** as depicted in Figure 5.5. This fully connected 2D mesh network is not partitioned into sub networks. It has an additional ring network that is exclusively used for traffic of type t2. Traffic t0 and t1 stay in the lower hierarchy level, i.e. the mesh.

The evaluations in terms of hop counts are performed and presented for different sizes of the 2D mesh $n \times n$, with n as edge length of the network. The size of the hierarchical networks are correspondingly equal, with the exception that they are partitioned into sub networks and the global ring network is added at a higher hierarchy level. The latency evaluations are only performed for networks with a size of 6×6 routers and the corresponding size of the hierarchical mesh networks.

For a fair comparison of all network architectures, the same number of tiles has to be active, i.e. send traffic, in each network. Since one router of each of the four sub network of the hierarchical networks is required to connect the sub network to the higher layer ring network, only 32 tiles are active in each of the modeled networks. Only these active tiles are sending traffic of types t0, t1 and t2 and are also receiving t0 and t1. In addition, two central tiles that receive traffic of type t2 are allocated in each network. These central tiles only receive traffic, e.g. write request packets, they do not send any packets in return to the senders of the incoming packets. This means, the simulated traffic scenarios do not contain any message dependencies. Thus, message dependent deadlocks are not considered in the following investigations.

Three types of traffic, already introduced above, are simulated. The following list gives details on the distributions of the destinations of the packets according to their traffic type.

- The local traffic t0 in the sub networks of the hierarchical networks is simply traffic uniformly distributed to all tiles of the sub network. Accordingly, the maximum hop count of this traffic in the 3×3 sub networks is 5 and the average hop count is 2.9. In order to get a fair comparison, traffic t0 has to be modeled in the standard and extended mesh network with the same average hop count. Therefore, a similar traffic model is used as in [57] to compare hierarchical and flat network topologies. According to this model traffic t0 is produced by generating uniformly distributed packets to all tiles within the corresponding maximum number of hops, here 5

hops, of a sending tile. However, this results in an average hop count of 3.6 for t0. So, in contrast to [57], the maximum distance for traffic t0 in the standard and extended mesh is reduced to 4, resulting in an average hop count of 3.0.

- The destinations of the packets of traffic type t1 are chosen according to a uniform distribution among all tiles of the network excluding the central tiles. The relation between the traffic types t0 and t1 in the simulations is: $\frac{t0}{t1} = \frac{3}{1}$. The generation rate of traffic $(t0 + t1)$ is increased during the simulations.

- Traffic t2 is targeted to the central tiles and its rate is chosen in such a way to almost fully load the network interface on the central tiles.

The inter packet times of all the traffic types are randomly generated according to the Poisson distribution. The lambda value of this distribution is chosen in such a way to get the desired traffic rate. The size of all generated packets is constant. Packets of the traffic types t0 and t1 are all 5 flits long, packets of type t2 that are destined to one of the central tiles are 10 flits long.

All routers of the evaluated networks are input and output buffered. The length of all router buffer queues is 2 flits. The mesh networks use XY-routing to avoid deadlocks and thus require no virtual channels. The global ring employs shortest path routing and the ring routers implement 2 virtual channels to avoid deadlocks. The routing in the 2 VCs of the global ring is done in a spiral fashion in order to break the routing cycle in the channel dependency graph.

The latency values presented below include the network access latency as well as the network latency. Thus, the measurement of a packet latency starts when it is put into the send queue, after it has been generated by the tile. The measurement ends after the packet has been completely received at the destination tile.

5.4.2 Hop Counts and Latency Evaluation

Figure 5.6 shows the hop count evaluations for the network architectures and different network sizes. The right diagram of this figure shows the hop count values for the traffic t2. There is a significant reduction of the average as well as the maximum hop count for this type of traffic in the hierarchical networks in comparison to the standard mesh

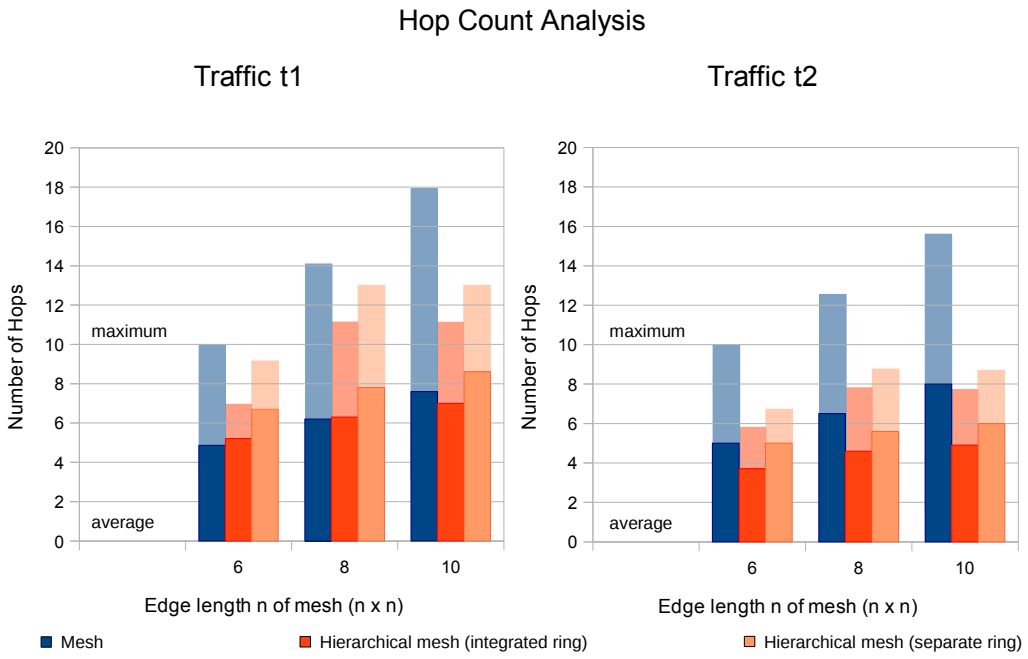

FIGURE 5.6: Hop count analysis of a standard 2D mesh and the presented hierarchical mesh architectures.

network. The decrease of the hop counts in comparison to the standard mesh network grows especially for larger network sizes. As expected, integrating the global ring routers into the sub mesh networks (hMiR) reduces the hop count by one for all packets of traffic t2 in comparison to hierarchical networks with separate ring routers (hMsR).

The left diagram of Figure 5.6 shows the hop count evaluation of traffic t1. The maximum hop counts of packets of traffic type t1 are also decreased by the hierarchical networks. In contrast to the hop count reductions of traffic t2, the hierarchical networks are not able to significantly reduce the average hop counts for traffic t1. The hierarchical networks with separate global ring routers are not able to provide savings at all. Only with the integrated ring routers, which save 2 hops for packets of traffic t1 in comparison to the separate routers, a reduction of average hop counts is possible, but only for larger networks.

The right diagram of Figure 5.7 supports the claim that the hierarchical network architectures are able to significantly reduce the packet latencies of traffic type t2 in comparison to the 2D mesh (mesh). However, Figure 5.7 also shows that the hierarchical network with the integrated global ring (hMiR) actually cannot profit from the decreased

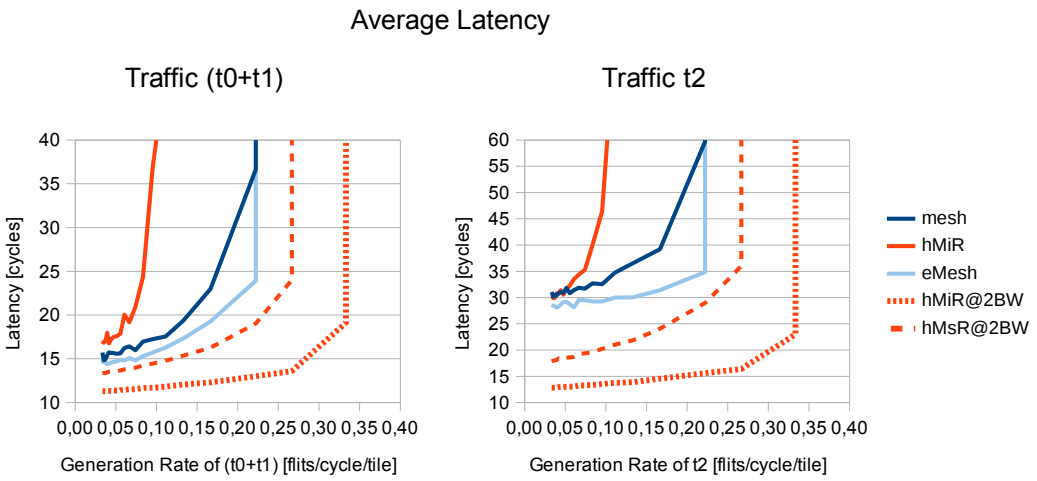

FIGURE 5.7: Average latencies of traffic classes t0+t1 and t2 [1].

average hop count (20%) in comparison to the standard mesh. The packet latencies of traffic t2 are not reduced in comparison to the mesh. The reason is a congestion in the global ring network that provides the access to the central tiles and interconnects the sub networks. This congestion occurs, because of the low bandwidth of the integrated global ring (hMiR), which is obviously overloaded by the applied traffic (t1 and t2). This can be seen in all latency diagrams of Figures 5.8 and 5.7. All these latency graphs of the hierarchical mesh with integrated ring routers (hMiR) show a sharp increase at a packet generation rate of approximately $0.02(t0 + t1)$. Again, the reason is the low bandwidth of the global ring of this hierarchical network. The throughput of this ring is not able to cope with the traffic t2 and t1 aggregating in the upper hierarchy level.

To improve the performance of the hierarchical network architectures the bottleneck of the global ring has to be removed. The bandwidth of the global ring is increased by doubling the clock frequencies of the routers that are part of the ring, which means of the upper hierarchy level of the two hierarchical networks with the postfix @2BW (hMiR@2BW and hMsR@2BW). Without the virtual channels the higher router clock would not help to accelerate the transfer of the packets over the global ring. This is due to the wormhole forwarding scheme employed in the networks. Since flits of different packets must not be mixed in the buffer queues of the routers, the problem would be that the flits of packets entering the global ring are only offered at the lower router clock by the sub-networks. However, because of the 2 virtual channels implemented in the

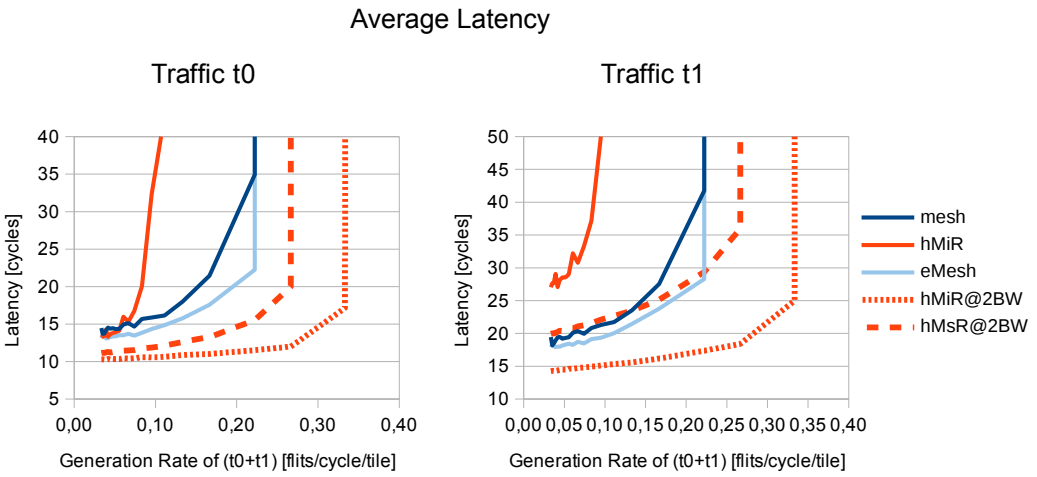

FIGURE 5.8: Average latencies of traffic classes t0 and t1 [1].

global ring, the faster ring routers can alternately serve flits of the two channels. In this way, this measure is able to significantly boost the throughput of the hierarchical networks. It becomes even higher than that of the mesh as Figures 5.8 and 5.7 show. Of course, increasing the clock frequencies of the global ring's routers also helps to reduce the packet latencies. The latencies of traffic t2 are almost halved for the hierarchical network with the integrated global ring (hMiR@2BW) in comparison to the mesh. The latencies of traffic t2 of the hierarchical network with separate ring routers (hMsR@2BW) are slightly higher than with the integrated routers (hMiR@2BW), due to the additional hop count for this type of traffic (see Figure 5.6 right diagram).

Comparing the t2 packet latencies of the extended and the standard mesh in the right diagram of Figure 5.7 shows that the reduced hop count has only a minor influence. The graph also shows that the latency of traffic t2 is decoupled from the rest of the traffic $(t0 + t1)$. The latency of packets of traffic type t2 increases only noticeably after the throughput of the extended mesh for $(t0 + t1)$ is reached. Both hierarchical networks (hMiR@2BW and hMsR@2BW) with the increased bandwidth of their global ring network outperform the other networks by length in terms of the latency of traffic t2.

In the right diagram of Figure 5.8 the latency results for traffic t1 are depicted. Again, the hierarchical network with the global ring operating at normal router clock (hMiR) suffers from the low bandwidth of the upper hierarchy which is also used by traffic of

type t1. The latencies are almost doubled in comparison to the standard mesh at low traffic rates. This network provides only a very low throughput, as the latency increases already sharply for packet generation rates of only $0.1(t0 + t1)$. The doubling of the bandwidth of the global ring also greatly affects the traffic t1. The network with the separate ring routers (hMsR@2BW) achieves latencies comparable to the standard mesh at low traffic rates. At higher traffic generation rates $(t0 + t1)$ it even outperforms the mesh. The best performance is again shown by the hierarchical network with integrated ring routers at double bandwidth (hMiR@2BW).

Although traffic of type t0 does not use the upper layers of the hierarchical networks, this traffic also profits from the introduction of the additional hierarchy level. The left diagram of Figure 5.8 shows how the latencies of traffic t0 of the extended mesh are reduced in comparison to the standard mesh, only due to the introduced global ring. However, the hierarchical network with the integrated routers (hMiR) does not profit from this measure. The problem is again the aggregation of traffic t1 and t2 in the global ring, which leads to a congestion. Packets of traffic t1 and t2 that try to enter the global ring and have to wait, congest also the sub networks. Thus, traffic of type t0 is also affected. By doubling the global ring's bandwidth in the other two hierarchical networks (hMiR@2BW and hMsR@2BW), this effect vanishes and the latencies of packets of type t0 are even decreased in comparison to the standard mesh. The reason is that traffic of types t1 and t2 leaves the local sub-networks after a few hops and uses the global ring to travel to its destinations and thus mostly leaves the local sub-networks to traffic t0.

The simulation results presented above show that the hierarchical networks can considerably lower the latencies of the applied traffic scenario. This applies especially for the traffic t2, but can also hold true for traffic of type t0 and t1. However, a prerequisite is that the network of the upper hierarchy provides sufficient bandwidth to cope with the aggregating traffic t1 and t2.

5.5 Summary and General Evaluation

Hierarchical, multi-topology networks enable a higher degree of adaptation to the requirements of the application and an optimized connection of heavily accessed tiles. These tiles, called central tiles here, can be connected to the higher or highest hierarchy

of the hierarchical network, to shorten their network distance from all accessing tiles. The network architecture can be adapted to the application by using (different) network topologies on the different sub-networks that fit the requirements best. In this way, the communication cost of transfers is greatly reduced in terms of hop counts and latency.

The improved connection of the central tiles is shown by the simulation results presented above. In addition, the results reveal how the performance of the proposed network architectures depends on the realizable routers and their throughput. When designing hierarchical networks, great care has to be taken to adequately implement the networks of the upper hierarchy layers. The traffic expected for the higher layers has to be estimated carefully and the bandwidth of the corresponding interconnect has to be designed accordingly.

Chapter 6

Conclusion

6.1 Summary of Scientific Contributions

Message dependent deadlocks are a problem that has to be considered in any system that is based on a packet switched NoC. Since most NoCs apply wormhole forwarding, this problem is relevant for most systems. The standard approach to message dependent deadlocks is deadlock avoidance and the most common concept is strict ordering. It reliably avoids the occurrence of message dependent deadlocks, but it approximately doubles the required chip area of the network. This is a quite costly measure for an actually rarely occurring type of deadlocks. While strict ordering is able to increase the network throughput compared to a NoC without virtual channels, it is not able to provide any additional functional advantages, like fault tolerance, despite its high costs.

This dissertation proves that deadlock recovery, especially regressive deadlock recovery, called sdNoC here, is an adequate solution to message dependent deadlocks. Regressive recovery is able to resolve all deadlocks. Here, it is only applied to message dependent deadlocks since routing dependent deadlocks can easily be resolved by corresponding routing algorithms. The investigations show that despite additional acknowledgement traffic, regressive recovery can provide equal or even better throughput than strict ordering depending on the traffic scenario.

In contrast to deadlock avoidance schemes, sdNoC even provides further functional advantages: The packet discard feature can solve temporary local congestions and hereby increase the total effective throughput of the network. In addition, the retransmission

capabilities of sdNoC provide fault tolerance to any transient error occurring in the network. This encompasses data bit flips of the flits on the links or in the buffers, but also errors in the control bits of the flits or of the router logic. As CMOS circuits become more and more sensitive to all forms of manufacturing and environmental variations, due to the constantly shrinking feature sizes, packet or data loss will have to be covered anyhow. SdNoC allows to use the measures implemented for fault tolerance to solve message dependent deadlocks at the same time. This is a major advantage over all other strategies to counter message dependent deadlocks.

SdNoC also achieves a complete independence of the network architecture from the number of dependent messages in the communication protocols. In a system based on strict ordering the complexity of the communication protocols is limited by the number of virtual channels or physically independent networks. This issue also has to be considered by application developers. They have to avoid creating communication between tiles that contains a higher number of message dependencies than the implemented NoC supports. In contrast, communication in sdNoC is not limited in regard to dependent messages in communication protocols.

Supposing that the retransmission buffer can be allocated in a tile local memory, sdNoC is able to provide all the above listed features with 23% (depending on router buffers) less chip area than strict ordering realized by virtual channels. In comparison to strict ordering, all these advantages have to be paid by a higher variability of transfer latencies, although the average latencies are equal or only slightly higher. Future work on sdNoC should focus on the implementation and evaluation of applications executed on a system based on sdNoC. This is necessary to evaluate the effect of sdNoC's latency variability on different types of applications. For example, how are cache coherency protocols or even time critical applications like audio or video decoders affected by retransmissions of packets.

The alternative deadlock recovery scheme, that was evaluated first in this dissertation, the progressive deadlock recovery for NoCs does not perform as good as sdNoC. Progressive recovery in which deadlocks are resolved by redirecting packets to a reserved channel, suffers from the limited bandwidth of this channel and the therefore limited capabilities to resolve deadlocks. The concept is not able to support higher network loads, it only works well at low to medium loaded networks. While the routers can save

almost half of the buffer space in comparison to strict ordering, the network adapters are not able to provide any savings. The progressive deadlock recovery scheme for NoCs presented in this dissertation does neither provide any additional functionality.

Not only the strategy used to counter the problem of message dependent deadlocks has a major effect on the performance and cost of a NoC, the topology is also significantly influencing these aspects. Instead of flat network topologies that are employed in most systems, this dissertation proposes hierarchical, multi-topology network architectures that offer major advantages. The sub networks of the hierarchical network architecture can be individually adapted to the needs of the application. This enables a higher degree of optimization of the total system in terms of cost and performance. Another aspect is the placement of communication centric tiles, such as shared memories or IOs, that are heavily accessed by many other tiles of the system. These tiles can be connected to the upper hierarchies of the network in order to reduce the network distance of all accessing tiles to a minimum. This architecture also eases the central interconnection of communication centric tiles in the network, even if they have to be placed at the border of the chip due to pin constraints.

6.2 Outlook

The hierarchical, multi-topology NoC, presented in Chapter 5, was evaluated with the help of partitioned, but still uniformly distributed traffics. Thus, the evaluation in the simulations were just targeted at effects of optimizing the interconnection of central tiles, such as shared resources or external interfaces. For a quantitative evaluation of the multi-topology property, based on simulations, parallelized applications for heterogeneous systems are required.

Depending on the application's traffic characteristics of such heterogeneous systems, it can even be advantageous to use different forwarding and switching schemes in the sub-partitions. For example, ring topologies could use TDMA to profit from higher clock rates and simpler router architectures. This can apply to ring topologies on lower as well as on higher hierarchy layers. Sub-meshes of the same systems can still be based on wormhole forwarding. Of course, this requires converters to be put between the domains

using different protocols. Whether the additional costs, independent of whether these are area or latency costs, are worth it, would have to be evaluated.

Both deadlock recovery systems presented can also profit from further evaluations performed with traffic scenarios derived from real, parallelized application. For this matter, adequate applications have to be found that can be parallelized on many cores and the traffic flows have to be determined and modeled by the abstract traffic generators of the NoC simulation model used for the evaluations of this thesis. In order to evaluate the execution time of applications by simulation, the behavior of the tiles has to be modeled in more detail, not just by the abstract traffic generators. This also holds true, for investigations on the use of selective packet discard in NoCs in combination with coherence protocols. Such investigations can determine the overhead, i.e. the added latency or delay, from discarded packets on the execution of applications that are based on coherence protocols.

The retransmission and discard mechanisms presented for sdNoC leaves also room for further research. Both mechanisms are more or less static, i.e. packets are retransmitted after a certain period (with a minimal random part to avoid livelocks) or discarded. In the evaluations, values for the corresponding parameters were explored for certain traffic scenarios. The evaluations showed that sdNoC provides some tolerance regarding the parameter values, i.e. small changes of the parameters, like the retransmission period, do not lead to dramatic changes of the performance. However, in case of significantly temporally changing traffic scenarios an adaptation of sdNoC's parameters would enable optimal performance at all times. For example, in the evaluations, the retransmission period was determined for a system with many actively sending tiles. Hereby, the retransmission also has a limiting effect on the send rate of the tiles. Assuming a change of the traffic scenario, due to the switch to another application for example, that only requires a few number of communicating cores. In this situation, the limiting effect on the send rate is only desired in reduced way, if it is desired at all.

However, dynamically adapting the parameters of sdNoC, as in a self-optimizing system, is not that easy to realize. The according decisions cannot be taken by the individual tiles with the limited knowledge that they have on the network's load situation from monitoring the status of the network interface a tile is connected to. For the distribution of the load information that is the basis for decisions of such kind, additional data has

to be transfered over the network either by piggybacking the information on the normal data packets or by additional packets. As an alternative, the load information could also be transported by an additional interconnect with reduced functionality and data width. Providing the tiles with comprehensive load information of the NoC enables the tiles to take their own decisions. In contrast to such a distributed approach, a solution in which a centralized entity takes these decisions on behalf of all tiles is also conceivable. This feedback loop, providing the sending tiles with load information of the network, would also enable self-learning mechanisms controlling and optimizing the sdNoC parameters and thus the send rate of the tiles. The effort for such solutions has to be determined with the help of hardware models and has to be weight in against the performance gains.

Bibliography

[1] A. Lankes, T. Wild, and A. Herkersdorf. Hierarchical nocs for optimized access to shared memory and io resources. In *Digital System Design, Architectures, Methods and Tools, 2009. DSD '09. 12th Euromicro Conference on*, pages 255 –262, August 2009. doi: 10.1109/DSD.2009.158.

[2] Gordon E. Moore. Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114 ff. *Solid-State Circuits Society Newsletter, IEEE*, 11(5):33–35, 2006. ISSN 1098-4232. doi: 10.1109/N-SSC.2006.4785860.

[3] W. Wolf, A.A. Jerraya, and G. Martin. Multiprocessor system-on-chip (mpsoc) technology. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(10):1701–1713, October 2008. ISSN 0278-0070. doi: 10.1109/TCAD.2008.923415.

[4] Shekhar Borkar. Thousand core chips: a technology perspective. In *Proceedings of the 44th annual Design Automation Conference*, DAC '07, pages 746–749, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-627-1. doi: 10.1145/1278480.1278667. URL http://doi.acm.org/10.1145/1278480.1278667.

[5] Shekhar Borkar and Andrew A. Chien. The future of microprocessors. *Commun. ACM*, 54(5):67–77, May 2011. ISSN 0001-0782. doi: 10.1145/1941487.1941507. URL http://doi.acm.org/10.1145/1941487.1941507.

[6] William J. Dally and Brian Towles. Route packets, not wires: On-chip inteconnection networks. In *Proceedings of the 38th Annual Design Automation Conference*, DAC '01, pages 684–689, New York, NY, USA, 2001. ACM. ISBN 1-58113-297-2. doi: 10.1145/378239.379048. URL http://doi.acm.org/10.1145/378239.379048.

[7] Luca Benini and Giovanni De Micheli. Networks on chips: A new soc paradigm. *Computer*, 35(1):70–78, January 2002. ISSN 0018-9162. doi: 10.1109/2.976921. URL http://dx.doi.org/10.1109/2.976921.

[8] Jose Duato, Sudhakar Yalamanchili, and Ni Lionel. *Interconnection Networks: An Engineering Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002. ISBN 1558608524.

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

[10] Axel Jantsch and Hannu Tenhunen et al. *Networks on Chip*. Kluwer Academic Publishers, Hingham, MA, USA, 2003. ISBN 1-4020-7392-5.

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

[12] Timo D. Hamalainen Erno Salminen, Ari Kulmala. Survey of network-on-chip proposals. *www.ocpip.org*, page 13, April 2008.

[13] C.L. Seitz and et al. Wormhole chip project report. 1985.

[14] J.D. Owens, W.J. Dally, R. Ho, D.N. Jayasimha, S.W. Keckler, and Li-Shiuan Peh. Research challenges for on-chip interconnection networks. *IEEE Micro*, 27(5): 96–108, Sept.-Oct. 2007. ISSN 0272-1732. doi: 10.1109/MM.2007.4378787.

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

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

[17] P. Gratz, Changkyu Kim, R. McDonald, S.W. Keckler, and D. Burger. Implementation and evaluation of on-chip network architectures. In *Computer Design, 2006. ICCD 2006. International Conference on*, pages 477 –484, Oct 2006. doi: 10.1109/ICCD.2006.4380859.

[18] Umit Y. Ogras, Jingcao Hu, and Radu Marculescu. Key research problems in noc design: a holistic perspective. In *Proceedings of the 3rd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, CODES+ISSS '05, pages 69–74, New York, NY, USA, 2005. ACM. ISBN 1-59593-161-9. doi: 10.1145/1084834.1084856. URL http://doi.acm.org/10.1145/1084834.1084856.

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

[20] Christopher J. Glass and Lionel M. Ni. The turn model for adaptive routing. In *Proceedings of the 19th annual international symposium on Computer architecture*, ISCA '92, pages 278–287, New York, NY, USA, 1992. ACM. ISBN 0-89791-509-7. doi: 10.1145/139669.140384. URL http://doi.acm.org/10.1145/139669.140384.

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

[22] Jingcao Hu and Radu Marculescu. Dyad: Smart routing for networks-on-chip. In *Proceedings of the 41st Annual Design Automation Conference*, DAC '04, pages 260–263, New York, NY, USA, 2004. ACM. ISBN 1-58113-828-8. doi: 10.1145/996566.996638. URL http://doi.acm.org/10.1145/996566.996638.

[23] Yong Ho Song and T. M. Pinkston. A progressive approach to handling message-dependent deadlock in parallel computer systems. *IEEE J PDS*, 14(3):259–275, March 2003. doi: 10.1109/TPDS.2003.1189584.

[24] Fernando Moraes, Ney Calazans, Aline Mello, Leandro Möller, and Luciano Ost. Hermes: an infrastructure for low area overhead packet-switching networks on chip.

Integr. VLSI J., 38(1):69–93, 2004. ISSN 0167-9260. doi: http://dx.doi.org/10.1016/j.vlsi.2004.03.003.

[25] Matteo Dall'Osso, Gianluca Biccari, Luca Giovannini, Davide Bertozzi, and Luca Benini. xpipes: a latency insensitive parameterized network-on-chip architecture for multi-processor socs. *Computer Design, International Conference on*, 0:536, 2003. ISSN 1063-6404. doi: http://doi.ieeecomputersociety.org/10.1109/ICCD.2003.1240952.

[26] Davide Bertozzi and Luca Benini. Xpipes: A network-on-chip architecture for gigascale systems-on-chip. *IEEE Circuits and Systems Magazine*, 4(2):18–31, 2004. doi: 10.1109/MCAS.2004.1330747.

[27] T. Bjerregaard. *The MANGO clockless network-on-chip: Concepts and implementation*. PhD thesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2005. URL http://www2.imm.dtu.dk/pubdb/p.php?4025. Supervised by Assoc. Prof. Jens Sparsø, IMM.

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

[29] S. Borkar. Designing reliable systems from unreliable components: the challenges of transistor variability and degradation. *Micro, IEEE*, 25(6):10 – 16, November-December 2005. ISSN 0272-1732. doi: 10.1109/MM.2005.110.

[30] Mitra S., Narayanan V., Spainhower L., and Xie Y. Robust system design from unreliable components. Technical report, International Symposium on Computer Architecture, 2005.

[31] Andreas Lankes, Thomas Wild, Andreas Herkersdorf, Soeren Sonntag, and Helmut Reinig. Comparison of deadlock recovery and avoidance mechanisms to approach message dependent deadlocks in on-chip networks. pages 17–24, 2010. doi: http://dx.doi.org/10.1109/NOCS.2010.11.

[32] Andreas Lankes, Thomas Wild, Stefan Wallentowitz, and Andreas Herkersdorf. Benefits of selective packet discard in networks-on-chip. *ACM Trans. Archit. Code Optim.*, 9(2):12:1–12:21, Jun. 2012. ISSN 1544-3566. doi: 10.1145/2207222.2207228. URL http://doi.acm.org/10.1145/2207222.2207228.

[33] W. J. Dally and H. Aoki. Deadlock-free adaptive routing in multicomputer networks using virtual channels. *IEEE Trans. Parallel Distrib. Syst.*, 4(4):466–475, Apr. 1993. ISSN 1045-9219. doi: 10.1109/71.219761. URL http://dx.doi.org/10.1109/71.219761.

[34] Robert Mullins, Andrew West, and Simon Moore. Low-latency virtual-channel routers for on-chip networks. In *ISCA '04: Proceedings of the 31st annual international symposium on Computer architecture*, page 188, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2143-6.

[35] Wan-Ting Su, Jih-Sheng Shen, and Pao-Ann Hsiung. Network-on-chip router design with buffer-stealing. In *Proceedings of the 16th Asia and South Pacific Design Automation Conference*, ASPDAC '11, pages 160–164, Piscataway, NJ, USA, 2011. IEEE Press. ISBN 978-1-4244-7516-2. URL http://dl.acm.org/citation.cfm?id=1950815.1950859.

[36] I. Walter, I. Cidon, R. Ginosar, and A. Kolodny. Access regulation to hot-modules in wormhole nocs. *Networks-on-Chip, 2007. NOCS 2007. First International Symposium on*, pages 137–148, 7-9 May 2007. doi: 10.1109/NOCS.2007.8.

[37] David Wentzlaff, Patrick Griffin, Henry Hoffmann, Liewei Bao, Bruce Edwards, Carl Ramey, Matthew Mattina, Chyi-Chang Miao, John F. Brown III, and Anant Agarwal. On-chip interconnection architecture of the tile processor. *IEEE Micro*, 27(5):15–31, Sept.-Oct. 2007. ISSN 0272-1732. doi: 10.1109/MM.2007.4378780.

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

[39] Parviz Kermani and Leonard Kleinrock. Virtual cut-through: a new computer communication switching technique. *Computer Networks*, 3:267–286, 1979.

[40] Mario Gerla, Prasasth Palnati, and Simon Walton. Multicasting protocols for high-speed, wormhole-routing local area networks. In *SIGCOMM '96: Conference proceedings on Applications, technologies, architectures, and protocols for computer communications*, pages 184–193, New York, NY, USA, 1996. ACM Press. ISBN 0-89791-790-1. doi: http://doi.acm.org/10.1145/248156.248173.

[41] Tobias Bjerregaard and Jens Sparso. A router architecture for connection-oriented service guarantees in the mango clockless network-on-chip. In *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, pages 1226–1231, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2288-2. doi: http://dx.doi.org/10.1109/DATE.2005.36.

[42] Aline Mello, Leonel Tedesco, Ney Calazans, and Fernando Moraes. Virtual channels in networks on chip: implementation and evaluation on hermes noc. In *SBCCI '05: Proceedings of the 18th annual symposium on Integrated circuits and system design*, pages 178–183, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-174-0. doi: http://doi.acm.org/10.1145/1081081.1081128.

[43] Kees Goossens, John Dielissen, and Andrei Radulescu. Aethereal network on chip: Concepts, architectures, and implementations. *IEEE Des. Test*, 22(5):414–421, 2005. ISSN 0740-7475. doi: http://dx.doi.org/10.1109/MDT.2005.99.

[44] M. Coppola, R. Locatelli, G. Maruccia, L. Pieralisi, and A. Scandurra. Spidergon: a novel on-chip communication network. In *System-on-Chip, 2004. Proceedings. 2004 International Symposium on*, pages 15–, November 2004. doi: 10.1109/ISSOC.2004.1411133.

[45] Srinivasan Murali and Giovanni De Micheli. Sunmap: a tool for automatic topology selection and generation for nocs. In *Proceedings of the 41st annual Design Automation Conference*, DAC '04, pages 914–919, New York, NY, USA, 2004. ACM. ISBN 1-58113-828-8. doi: 10.1145/996566.996809. URL http://doi.acm.org/10.1145/996566.996809.

[46] Srinivasan Murali, Paolo Meloni, Federico Angiolini, David Atienza, Salvatore Carta, Luca Benini, Giovanni De Micheli, and Luigi Raffo. Designing application-specific networks on chips with floorplan information. In *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*, ICCAD '06, pages

355–362, New York, NY, USA, 2006. ACM. ISBN 1-59593-389-1. doi: 10.1145/1233501.1233573. URL http://doi.acm.org/10.1145/1233501.1233573.

[47] Andreas Hansson, Kees Goossens, and Andrei Rădulescu. A unified approach to constrained mapping and routing on network-on-chip architectures. In *Proceedings of the 3rd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, CODES+ISSS '05, pages 75–80, New York, NY, USA, 2005. ACM. ISBN 1-59593-161-9. doi: 10.1145/1084834.1084857. URL http://doi.acm.org/10.1145/1084834.1084857.

[48] Jiang Xu, Wayne Wolf, Joerg Henkel, and Srimat Chakradhar. A design methodology for application-specific networks-on-chip. *Trans. on Embedded Computing Sys.*, 5(2):263–280, 2006. ISSN 1539-9087. doi: http://doi.acm.org/10.1145/1151074.1151076.

[49] Davide Bertozzi, Antoine Jalabert, Srinivasan Murali, Rutuparna Tamhankar, Stergios Stergiou, Luca Benini, and Giovanni De Micheli. Noc synthesis flow for customized domain specific multiprocessor systems-on-chip. *IEEE Trans. Parallel Distrib. Syst.*, 16(2):113–129, February 2005. ISSN 1045-9219. doi: 10.1109/TPDS.2005.22. URL http://dx.doi.org/10.1109/TPDS.2005.22.

[50] P. Gschwandtner, T. Fahringer, and R. Prodan. Performance analysis and benchmarking of the intel scc. In *Cluster Computing (CLUSTER), 2011 IEEE International Conference on*, pages 139–149, September 2011. doi: 10.1109/CLUSTER.2011.24.

[51] M. Coppola, M. D. Grammatikakis, R. Locatelli, G. Maruccia, and L. Pieralisi. *Design of Cost-Efficient Interconnect Processing Units: Spidergon STNoC*. CRC Press, Inc., Boca Raton, FL, USA, 2008.

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

[53] Francisco Gilabert, Simone Medardoni, Davide Bertozzi, Luca Benini, Maria Engracia Gomez, Pedro Lopez, and Jose Duato. Exploring high-dimensional topologies for noc design through an integrated analysis and synthesis framework. In *NOCS*

'08: Proceedings of the Second ACM/IEEE International Symposium on Networks-on-Chip (nocs 2008), pages 107–116, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3098-7.

[54] Govindan Ravindran and Michael Stumm. A performance comparison of hierarchical ring- and mesh- connected multiprocessor networks. In *HPCA '97: Proceedings of the 3rd IEEE Symposium on High-Performance Computer Architecture*, page 58, Washington, DC, USA, 1997. IEEE Computer Society. ISBN 0-8186-7764-3.

[55] V.C. Hamacher and Hong Jiang. Hierarchical ring network configuration and performance modeling. *Computers, IEEE Transactions on*, 50(1):1–12, Jan 2001. ISSN 0018-9340. doi: 10.1109/12.902749.

[56] S. Bourduas, B. Kuo, Z. Zilic, and N. Manjikian. Modeling and evaluation of an energy-efficient hierarchical ring interconnect for system-on-chip multiprocessors. *Circuits and Systems, 2006 IEEE North-East Workshop on*, pages 201–204, June 2006.

[57] S. Bourduas and Z. Zilic. A hybrid ring/mesh interconnect for network-on-chip using hierarchical rings for global routing. *Networks-on-Chip, 2007. NOCS 2007. First International Symposium on*, pages 195–204, 7-9 May 2007. doi: 10.1109/NOCS.2007.3.

[58] Donghyun Kim, Kwanho Kim, Joo-Young Kim, Seung-Jin Lee, and Hoi-Jim Yoo. Solutions for real chip implementation issues of noc and their application to memory-centric noc. *Networks-on-Chip, 2007. NOCS 2007. First International Symposium on*, pages 30–39, 7-9 May 2007. doi: 10.1109/NOCS.2007.40.

[59] W. J. Dally and C. L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Trans. Comput.*, 36(5):547–553, May 1987. ISSN 0018-9340. doi: 10.1109/TC.1987.1676939. URL http://dx.doi.org/10.1109/TC.1987.1676939.

[60] K. V. Anjan and Timothy Mark Pinkston. An efficient, fully adaptive deadlock recovery scheme: Disha. In *Proceedings of the 22Nd Annual International Symposium on Computer Architecture*, ISCA '95, pages 201–210, New York, NY, USA, 1995. ACM. ISBN 0-89791-698-0. doi: 10.1145/223982.224431. URL http://doi.acm.org/10.1145/223982.224431.

[61] M. Sfinghal. Deadlock detection in distributed systems. *Computer*, 22(11):37–48, Nov 1989. ISSN 0018-9162. doi: 10.1109/2.43525. URL http://dx.doi.org/10.1109/2.43525.

[62] Ajay D. Kshemkalyani and Mukesh Singhal. On characterization and correctness of distributed deadlock detection. *J. Parallel Distrib. Comput.*, 22(1):44–59, July 1994. ISSN 0743-7315. doi: 10.1006/jpdc.1994.1069. URL http://dx.doi.org/10.1006/jpdc.1994.1069.

[63] P. López and J. Duato. A very efficient distributed deadlock detection mechanism for wormhole networks. In *Proceedings of the 4th International Symposium on High-Performance Computer Architecture*, HPCA '98, pages 57–, Washington, DC, USA, 1998. IEEE Computer Society. ISBN 0-8186-8323-6. URL http://dl.acm.org/citation.cfm?id=822079.822719.

[64] Juan-Miguel Martínez Rubio, Pedro López, and José Duato. Fc3d: Flow control-based distributed deadlock detection mechanism for true fully adaptive routing in wormhole networks. *IEEE Trans. Parallel Distrib. Syst.*, 14(8):765–779, August 2003. ISSN 1045-9219. doi: 10.1109/TPDS.2003.1225056. URL http://dx.doi.org/10.1109/TPDS.2003.1225056.

[65] I. Saastamoinen, M. Alho, and J. Nurmi. Buffer implementation for proteo network-on-chip. volume 2, pages II–113 – II–116 vol.2, May 2003. doi: 10.1109/ISCAS.2003.1205906.

[66] J.H. Kim, Ziqiang Liu, and A.A. Chien. Compressionless routing: a framework for adaptive and fault-tolerant routing. In *Computer Architecture, 1994., Proceedings the 21st Annual International Symposium on*, pages 289 –300, Apr. 1994. doi: 10.1109/ISCA.1994.288141.

[67] Andreas Hansson, Kees Goossens, and Andrei Radulescu. Avoiding message-dependent deadlock in network-based systems on chip. *VLSI Design*, 2007, 2007. URL http://dblp.uni-trier.de/db/journals/vlsi/vlsi2007.html#HanssonGR07.

[68] Om Prakash Gangwal, Johan Janssen, Selliah Rathnam, Erwin Bellers, and Marc Duranton. Understanding video pixel processing applications for flexible implementations. In *Proceedings of the Euromicro Symposium on Digital Systems Design*,

DSD '03, pages 392–, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-2003-0. URL http://dl.acm.org/citation.cfm?id=942792.943104.

[69] Jingcao Hu, Umit Y. Ogras, and Radu Marculescu. System-level buffer allocation for application-specific networks-on-chip router design. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 25(12):2919 –2933, December 2006. ISSN 0278-0070. doi: 10.1109/TCAD.2006.882474.

[70] Terry Tao Ye, Luca Benini, and Giovanni De Micheli. Packetized on-chip interconnect communication analysis for mpsoc. In *Proceedings of the Conference on Design, Automation and Test in Europe - Volume 1*, DATE '03, pages 10344–, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1870-2. URL http://dl.acm.org/citation.cfm?id=789083.1022750.

[71] Frederic Petrot, Alain Greiner, and Pascal Gomez. On cache coherency and memory consistency issues in noc based shared memory multiprocessor soc architectures. In *Proceedings of the 9th EUROMICRO Conference on Digital System Design*, DSD '06, pages 53–60, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2609-8. doi: 10.1109/DSD.2006.73. URL http://dx.doi.org/10.1109/DSD.2006.73.

[72] Arteris. A comparison of network-on-chip and busses. *Design&Reuse*, 2005.

[73] Srinivasan Murali and Giovanni De Micheli. An application-specific design methodology for stbus crossbar generation. In *Proceedings of the conference on Design, Automation and Test in Europe - Volume 2*, DATE '05, pages 1176–1181, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2288-2. doi: 10.1109/DATE.2005.50. URL http://dx.doi.org/10.1109/DATE.2005.50.

[74] Young Jin Yoon, Nicola Concer, Michele Petracca, and Luca Carloni. Virtual channels vs. multiple physical networks: a comparative analysis. In *Proceedings of the 47th Design Automation Conference*, DAC '10, pages 162–165, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0002-5. doi: 10.1145/1837274.1837315. URL http://doi.acm.org/10.1145/1837274.1837315.

[75] Jin Liu and J.G. Delgado-Frias. A shared self-compacting buffer for network-on-chip systems. In *Circuits and Systems, 2006. MWSCAS '06. 49th IEEE International*

Midwest Symposium on, volume 2, pages 26 –30, August 2006. doi: 10.1109/MWSCAS.2006.382199.

[76] M. H. Neishaburi and Zeljko Zilic. Reliability aware noc router architecture using input channel buffer sharing. In *Proceedings of the 19th ACM Great Lakes symposium on VLSI*, GLSVLSI '09, pages 511–516, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-522-2. doi: http://doi.acm.org/10.1145/1531542.1531658. URL http://doi.acm.org/10.1145/1531542.1531658.

[77] Mingche Lai, Zhiying Wang, Lei Gao, Hongyi Lu, and Kui Dai. A dynamically-allocated virtual channel architecture with congestion awareness for on-chip routers. In *Proceedings of the 45th annual Design Automation Conference*, DAC '08, pages 630–633, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-115-6. doi: http://doi.acm.org/10.1145/1391469.1391630. URL http://doi.acm.org/10.1145/1391469.1391630.

[78] Andrei Radulescu, John Dielissen, Santiago Gonzalez Pestana, Om Prakash Gangwal, Edwin Rijpkema, Paul Wielage, and Kees Goossens. An efficient on-chip ni offering guaranteed services, shared-memory abstraction, and flexible network configuration. *IEEE Trans. on CAD of ICs and systems*, 24:2005, 2005.

[79] Mikael Millberg, Erland Nilsson, Rikard Thid, and Axel Jantsch. Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network on chip. In *Proceedings of the conference on Design, automation and test in Europe - Volume 2*, DATE '04, pages 20890–, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2085-5. URL http://dl.acm.org/citation.cfm?id=968879.969206.

[80] Y. Durand, C. Bernard, , and D Lattard. Faust: On-chip distributed architecture for a 4g baseband modem soc. *In Design & Reuse IP-SoC, IEEE Computer Society Press*, 2005.

[81] C. Gomez, M.E. Gomez, P. Lopez, and Duato J. Bps: A bufferless switching technique for nocs. In *Workshop on Interconnection Network Architectures: The 3rd International Conference on High-Performance Embedded Architectures and Compilers (HiPEAC)*, pages 43–50, January 2008.

[82] Mitchell Hayenga, Natalie Enright Jerger, and Mikko Lipasti. Scarab: a single cycle adaptive routing and bufferless network. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, pages 244–254, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-798-1. doi: http://doi.acm.org/10.1145/1669112.1669144. URL http://doi.acm.org/10.1145/1669112.1669144.

[83] Thomas Moscibroda and Onur Mutlu. A case for bufferless routing in on-chip networks. In *Proceedings of the 36th annual international symposium on Computer architecture*, ISCA '09, pages 196–207, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-526-0. doi: http://doi.acm.org/10.1145/1555754.1555781. URL http://doi.acm.org/10.1145/1555754.1555781.

[84] C. Fallin, C. Craik, and O. Mutlu. Chipper: A low-complexity bufferless deflection router. In *High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on*, pages 144–155, 2011. doi: 10.1109/HPCA.2011.5749724.

[85] Srinivasan Murali, Theocharis Theocharides, N. Vijaykrishnan, Mary Jane Irwin, Luca Benini, and Giovanni De Micheli. Analysis of error recovery schemes for networks on chips. *IEEE Des. Test*, 22(5):434–442, 2005. ISSN 0740-7475. doi: http://dx.doi.org/10.1109/MDT.2005.104.

[86] S. Borkar. The exascale challenge. In *VLSI Design Automation and Test (VLSI-DAT), 2010 International Symposium on*, pages 2 –3, april 2010. doi: 10.1109/VDAT.2010.5496640.

[87] Young Hoon Kang, Taek-Jun Kwon, and Jeffrey Draper. Fault-tolerant flow control in on-chip networks. In *NOCS '10: Proceedings of the 2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip*, pages 79–86, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4053-5. doi: http://dx.doi.org/10.1109/NOCS.2010.18.

[88] Dongkook Park, Chrysostomos Nicopoulos, Jongman Kim, N. Vijaykrishnan, and Chita R. Das. Exploring fault-tolerant network-on-chip architectures. In *DSN '06: Proceedings of the International Conference on Dependable Systems and Networks*, pages 93–104, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2607-1. doi: http://dx.doi.org/10.1109/DSN.2006.35.

[89] Daniel P. Siewiorek and Robert S. Swarz. *Reliable computer systems (3rd ed.): design and evaluation*. A. K. Peters, Ltd., Natick, MA, USA, 1998. ISBN 1-56881-092-X.

[90] Andras Varga. The omnet++ discrete event simulation system. *Proceedings of the European Simulation Multiconference (ESM'2001)*, June 2001.

[91] Andras Varga and Rudolf Hornig. An overview of the omnet++ simulation environment. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems*, Simutools '08, pages 60:1–60:10, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). ISBN 978-963-9799-20-2. URL http://dl.acm.org/citation.cfm?id=1416222.1416290.

[92] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.*, 1:397–413, August 1993. ISSN 1063-6692. doi: http://dx.doi.org/10.1109/90.251892. URL http://dx.doi.org/10.1109/90.251892.

[93] A.P. Frantz, M. Cassel, F.L. Kastensmidt, E. Cota, and L. Carro. Crosstalk- and seu-aware networks on chips. *Design Test of Computers, IEEE*, 24(4):340 –350, July-August 2007. ISSN 0740-7475. doi: 10.1109/MDT.2007.128.

[94] S. Wallentowitz, A. Lankes, A. Zaib, T. Wild, and A. Herkersdorf. A framework for open tiled manycore system-on-chip. In *22nd International Conference on Field Programmable Logic and Applications*, Oslo, Norway, August 2012.

[95] Luciano Bononi and Nicola Concer. Simulation and analysis of network on chip architectures: ring, spidergon and 2d mesh. In *DATE '06: Proceedings of the conference on Design, automation and test in Europe*, pages 154–159, 3001 Leuven, Belgium, Belgium, 2006. European Design and Automation Association. ISBN 3-9810801-0-6.

[96] A. Lankes, A. Herkersdorf, S. Sonntag, and H. Reinig. Noc topology exploration for mobile multimedia applications. In *Electronics, Circuits, and Systems, 2009. ICECS 2009. 16th IEEE International Conference on*, pages 707 –710, December 2009. doi: 10.1109/ICECS.2009.5410789.