Deadlock Free Routing in Mesh Networks on Chip with Regions

by

Rickard Holsmark

September 2009
ISBN 978-91-7393-559-3
Linköping Studies in Science and Technology
Thesis No. 1410
ISSN 0280-7971
LiU-Tek-Lic-2009:18

ABSTRACT

There is a seemingly endless miniaturization of electronic components, which has enabled designers to build sophisticated computing structures on silicon chips. Consequently, electronic systems are continuously improving with new and more advanced functionalities. Design complexity of these Systems on Chip (SoC) is reduced by the use of pre-designed *cores*. However, several problems related to the interconnection of cores remain. Network on Chip (NoC) is a new SoC design paradigm, which targets the interconnect problems using classical network concepts. Still, SoC cores show large variance in size and functionality, whereas several NoC benefits relate to regularity and homogeneity.

This thesis studies some network aspects which are characteristic to NoC systems. One is the issue of area wastage in NoC due to cores of various sizes. We elaborate on using oversized *regions* in regular mesh NoC and identify several new design possibilities. Adverse effects of regions on communication are outlined and evaluated by simulation.

Deadlock freedom is an important region issue, since it affects both the usability and performance of routing algorithms. The concept of faulty blocks, used in deadlock free fault-tolerant routing algorithms has similarities with rectangular regions. We have improved and adopted one such algorithm to provide deadlock free routing in NoC with regions. This work also offers a methodology for designing topology agnostic, deadlock free, highly adaptive *application specific* routing algorithms. The methodology exploits information about communication among tasks of an application. This is used in the analysis of deadlock freedom, such that fewer deadlock preventing routing restrictions are required.

A comparative study of the two proposed routing algorithms shows that the application specific algorithm gives significantly higher performance. But, the fault-tolerant algorithm may be preferred for systems requiring support for general communication. Several extensions to our work are proposed, for example in areas such as core mapping and efficient routing algorithms. The region concept can be extended for supporting reuse of a pre-designed NoC as a component in a larger hierarchical NoC.

This work has been supported by Jönköping University and the Swedish KK-Foundation.

Department of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping, Sweden

Deadlock Free Routing in Mesh Networks on Chip with Regions

Rickard Holsmark

Dept. of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping, Sweden

Dept. of Electronics and Computer Engineering
School of Engineering
Box 1026 Jönköping, Sweden

Linköping 2009

Deadlock Free Routing in Mesh Networks on Chip with Regions
by Rickard Holsmark

Linköping Studies in Science and Technology, No 1410

ISBN 978-91-7393-559-3
ISSN 0280-7971

Printed by LiU-Tryck, Linköping 2009

ABSTRACT

There is a seemingly endless miniaturization of electronic components, which has enabled designers to build sophisticated computing structures on silicon chips. Consequently, electronic systems are continuously improving with new and more advanced functionalities. Design complexity of these Systems on Chip (SoC) is reduced by the use of pre-designed *cores*. However, several problems related to the interconnection of cores remain. Network on Chip (NoC) is a new SoC design paradigm, which targets the interconnect problems using classical network concepts. Still, SoC cores show large variance in size and functionality, whereas several NoC benefits relate to regularity and homogeneity.

This thesis studies some network aspects which are characteristic to NoC systems. One is the issue of area wastage in NoC due to cores of various sizes. We elaborate on using oversized *regions* in regular mesh NoC and identify several new design possibilities. Adverse effects of regions on communication are outlined and evaluated by simulation.

Deadlock freedom is an important region issue, since it affects both the usability and performance of routing algorithms. The concept of faulty blocks, used in deadlock free fault-tolerant routing algorithms has similarities with rectangular regions. We have improved and adopted one such algorithm to provide deadlock free routing in NoC with regions. This work also offers a methodology for designing topology agnostic, deadlock free, highly adaptive *application specific* routing algorithms. The methodology exploits information about communication among tasks of an application. This is used in the analysis of deadlock freedom, such that fewer deadlock preventing routing restrictions are required.

A comparative study of the two proposed routing algorithms shows that the application specific algorithm gives significantly higher performance. But, the fault-tolerant algorithm may be preferred for systems requiring support for general communication. Several extensions to our work are proposed, for example in areas such as core mapping and efficient routing algorithms. The region concept can be extended for supporting reuse of a pre-designed NoC as a component in a larger hierarchical NoC.

ACKNOWLEDGEMENTS

During the work with this thesis, I have received large amounts of inspiration and support from colleagues, family and friends. I would like to explicitly express my gratitude to a few of those who have given the most.

I am forever grateful to my supervisor Shashi Kumar for his tremendous help and encouragement in my research. The positive spirit in our discussions is usually enough motivation for me to work.

Many warm thanks also to Maurizio Palesi, who is a bright and helpful person that I really like to work with. Each time we meet I learn something new.

I also highly appreciate the help and guidance from Petru Eles. He has given me loads of good advice in my studies and research activities.

My colleagues at School of Engineering, and especially those in Electronics and Computer Engineering, are great friends which I enjoy to work and converse with.

Finally, my wife and children: In spite of my shortcomings you have always enriched my life and shown great patience. Thank you - I love you.

TABLE OF CONTENTS

1

Introduction

The core of this thesis is communication. But, not communication among humans or other biological constructs. Instead the objects of interest are electronic components that, due to possibilities created by advancements in production technology, require new ways of communicating. More specifically, it is actually the communication among components *inside* a silicon chip.

The most famous silicon chip is probably the micro-processor in personal computers. Less known is perhaps that both processors and other types of electronic chips are found in almost every electronic device. Rapid improvements of chip manufacturing technology continuously provide higher transistor capacity of the chips. Higher capacity allows more components on the same chip, which in turn increases design complexity. This has led to the concept of Systems on Chip (SoC), where complete electronic systems are built by integrating components (cores) with well defined functionalities and interfaces.

As more and more cores can be used on a chip, the harder it becomes to design the wiring that interconnects the cores. This thesis is focused on some aspects of improving the interconnection of cores in a SoC. The work is performed within a recently proposed design paradigm, Networks on Chip (NoC), which views the SoC interconnect more as a network rather than an arbitrary interconnect structure.

1.1 Electronic Systems on Chip

Silicon chip capacity is increasing at an extremely high rate and the current, most advanced chips host billions of transistors. One sign of the increased capacity is the dramatic performance improvements of personal computers. The improved performance is due to several factors, but one of the most important and well-known is the increased capabilities of micro-processor chips, e.g. Intel Pentium and AMD Athlon. Although in the front-line of technology, these chips represent only a small fraction (less than 1% (Turley 2002)) of the total number of processor chips and an even smaller share of all electronic chips in the world.

Chip devices like processors, memories and various custom electronic circuits are found almost everywhere; cars, phones, ovens, TVs, airplanes and cameras are only a few examples. There is a large variation in capabilities among electronic chips and designing the most sophisticated chips is a complex task. To reduce complexity, the design of advanced electronic chips has turned towards a modular approach, where Systems on Chip (SoC) are formed by interconnecting *cores* or *IP-cores* (Intellectual Property) (Kucukcakar 1998). A core is a stand-alone component that has a specified, often advanced, functionality and some standard interface that facilitates system integration.

1.1.1 The System on Chip Interconnect Problem

Several of the existing core interconnect schemes in SoC are bus-based (Loghi et al. 2004). A bus is an interconnect technique where all cores share the wires of the interconnect. Access possibilities of each core to the bus decreases as the number of connected cores increases. Fulfilling the communication needs of a large number of cores, with varying communication requirements may be difficult using a bus-based interconnect. Therefore can solutions like point to point links and hierarchical buses also be seen in SoC interconnects (Goossens et al. 2004).

As the number of cores in a SoC increases, the higher will be the requirements on the interconnect. This not only results in wiring difficulties in new systems, but also in diminished possibilities to reuse and extend existing systems. It is often the case that interconnect is the performance bottleneck in SoC designs (Sylvester & Keutzer 1998) (Henkel et al. 2004).

1.2 NoC: A New Way to Design Complex Systems

The concept of Networks on Chip (NoC) emerged in the first years of the 21st century (Dally & Towles 2001) (Guerrier & Greiner 2000). Researchers argued that the traditional interconnection techniques were insufficient, in light of the escalating SoC interconnect problems and foreseen future communication requirements. Reuse of existing bus-based systems was also hindered by poor scalability. At the same time, use of dedicated connections was limited by physical on-chip realities.

The main idea of NoC is that inter-chip communication should be treated with a more network oriented view, and adopt concepts from the well established area of computer networks. A network usually exploits the fact that communication between nodes (computers) is not a constant activity. This makes it possible to organize communication such that the interconnect resources to some extent are shared. If resources are shared, cost of the total interconnect will be reduced.

Two commonly used SoC interconnect techniques can actually be seen as two extreme points in the network domain. The bus technique is sharing to the extreme - all nodes share the same communication resource. Point to point connections are the extreme at the other end - nothing is shared. Figure 1-1 illustrates two versions of a SoC; one with a bus-based interconnect and one with a network-based interconnect (NoC).

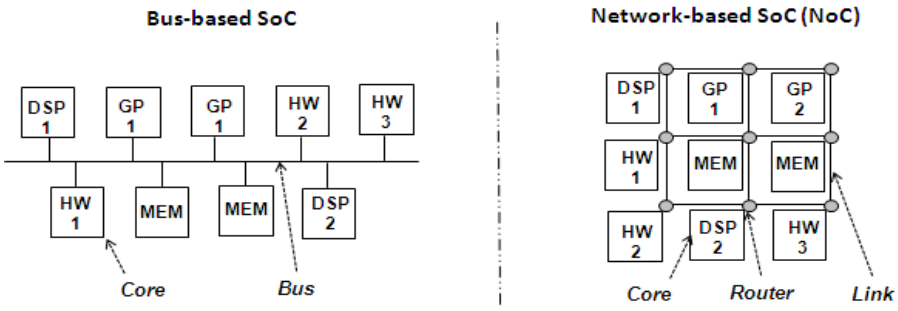


Figure 1-1: Bus-based and NoC-based Systems on Chip

The system consists of a number of cores of various types. Typical cores in a SoC are digital signal processing cores (DSP), general purpose processing (GP) cores, memories (MEM) and custom hardware (HW) cores. In the bus-based SoC, all cores are connected to a single bus. Once a message is sent on the bus, it is received directly by the destination core. In the network-based SoC, each core is connected to a router. A message from a core to another core is first sent to the router connected to the source core. Then it is forwarded by other network routers until it reaches the destination router, where it is delivered to the destination core.

Figure 1-1 hints at the main benefits and drawbacks of the two interconnects. Once a core is granted bus access, communication over the bus is very fast. But the more bus users (cores) there are, the less bus access-time will be available for each of them. This is why a bus is considered to have low scalability.

A message in the network-based system is not that fast delivered, as a transmission requires use of intermediate routers. Still, several messages can be sent simultaneously between different source and destination cores. As the addition of a core also implies an extra router and links, this organization is more scalable.

1.3 NoC Characteristics and Problem Area

Off-chip computer networks have been both implemented and researched on for many years. They are used in various applications; supercomputers, Internet and car electronics, are only a few examples. Several issues, like objectives and operational conditions affect the characteristics of a network. For example, the main objective for a supercomputer network is high performance. Supercomputers are mainly used for various computationally intensive applications and are placed in protected environments. Cost is of relatively less importance; these systems are, instead, mainly constrained by what is technologically possible (Brightwell et al. 2005). A CAN-bus on the other hand, is developed for other types of environments, like cars. It is designed for control oriented communication with lower data-rates. High importance is given to properties like real-time requirements, cost and modularity (Di Natale 2000).

It is likely that such types of application requirements will affect also characteristics of on-chip networks. But NoC also presents new types of network design constraints. The most fundamental difference between on-chip and off-chip networks is the constraints for resource usage. A chip has a certain amount of resources available and these can be allocated to either computation or communication tasks. Resources used for communication will inevitably reduce the resources available for computation. This is normally not the case for off-chip networks.

Besides this trade-off related to resources, there are several other aspects which are of high relevance to NoC. Two of these are of main interest for the work in this thesis:

1. Chip resource usage vs. design complexity
2. Application specific optimizations

The first aspect relates to a conflict between efficient use of chip area and the advantages with regular network structures. The reason is that efficient layout of size-varying SoC cores require customized irregular interconnect structures. Chip design complexity is on the other hand often reduced by regular interconnect structures.

The second aspect allows for the design of more optimized interconnect resources in NoCs as compared to off-chip networks. Off-chip networks are usually designed for arbitrary communication patterns. Several NoC applications though, may have more specific functionality. Knowledge of application communication can then be used to design more efficient network resources.

Both these aspects are briefly described in the following sub-sections.

1.3.1 NoC Interconnect Layout and Heterogeneity of Cores

Cores used in on-chip systems are heterogeneous with respect to size. This affects the possibilities of combining effective resource usage with a symmetric network structure. For example, study Figure 1-2 that depicts a set of cores of different sizes and two variants

of interconnect between them. The cores are connected to the router (white squares) in their upper right corner. The routers and the links between routers form the interconnect network.

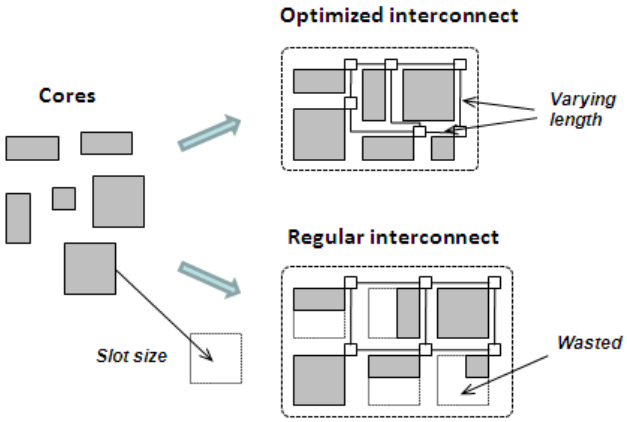


Figure 1-2: Optimized and regular interconnect with respect to core layout

The upper version has an optimized layout that does not require more resources than necessary to fit in the cores. This is effective regarding resource utilization but the structure of the interconnect is not symmetric. This non-symmetry influences other design parameters, where for example the varying wire lengths induce different transmission times between the routers. Electro-magnetical interference between interconnect and cores is also harder to estimate with an irregular wire structure. To conclude, non-symmetric interconnects have negative impact on design complexity.

The lower version of Figure 1-2, instead wastes resources by utilizing a symmetric interconnect. Symmetry in wire length requires that each core slot is of the same size, and consequently the core slot size must then be adjusted to the largest core. On the positive side is the possibility of more accurate estimation of electrical parameters, which in turn allows for higher interconnect performance, reduced design complexity and increased reusability.

Note that both versions are equal in one aspect; topology. In each of them, both the number of routers and links, as well as the connectivity between the routers is identical. Usually this organization is considered as a regular topology, called two-dimensional mesh. The symmetric property adds a new dimension to on-chip network topology, since off-chip networks are not constrained in the same way. For example, the mesh topology is usually depicted as a symmetric structure in network literature.

1.3.2 Application Specific NoC Communication

Another difference between NoC and off-chip networks is related to the amount of knowledge regarding applications that will use the network. Like other systems, a SoC

can be used for very different applications. Some are considered for general computations, like PC microprocessor chips, whereas others perform very specific tasks, like MPEG-4 H264 encoder/decoder chips. The last example falls within a computing area called embedded systems (Figure 1-3).

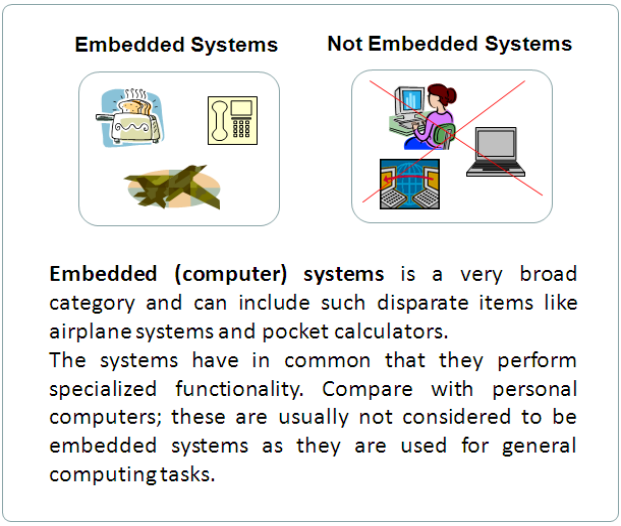


Figure 1-3: Embedded Systems

When these, also called *application specific* systems, are developed it is usually possible to have good knowledge about the communication among different components. Once the chip is manufactured it is unlikely that this communication pattern will change substantially. The communication knowledge can be used for optimizing network communication resources to a higher degree than what is possible for off-chip networks.

Recall Figure 1-2 with two different versions of core interconnects. It is clear that the non-symmetric version is optimized with respect to required silicon area. This type of optimization can be (but is often not) independent of the functionality of the system. Application specific optimization is in this respect of a different nature and can be applied to both types of interconnect. Consider for example the symmetric interconnect version in Figure 1-2. If the communication within the network is known, it is possible to plan the routes between cores such that contention is minimized. This way, network performance is increased without losing the benefits of a symmetric network.

1.4 NoC Characteristics and Deadlock Free Routing

Chip layout and application specific aspects affect design possibilities of NoC systems. Resource utilization advantages of customized interconnect layout and optimizations due to application knowledge are quite easily understood. One aspect which is more implicitly affected is the design space of efficient *deadlock-free* routing algorithms. A routing algorithm determines which routes or paths packets can take for each source - destination pair. Deadlock freedom is an important property for networks, since a packet deadlock may destroy possibilities of communication. A deadlock in this context refers to

a situation where packets are involved in a circular wait for resources. The overhead for resolving deadlocks can be costly and it is often desired that routing algorithms are deadlock free, i.e. guarantees that deadlocks cannot occur.

The requirement of deadlock freedom limits the available routes of a routing algorithm. In general, irregular interconnect and application specific optimization affect design of efficient deadlock-free routing algorithms in opposite ways:

Irregular interconnect: *Reduced* possibilities of efficient routing because of the need for more complex routing algorithms.

Application specific communication: *Increased* possibilities of efficient routing because of the opportunity to optimize routing algorithms.

There exist several deadlock-free routing algorithms for regular networks. These are relatively efficient with respect to cost and performance. But, new possibilities of deadlock may render them unusable for even the slightest change in topology. In such cases, a deadlock-free routing algorithm for an irregular topology is required. Irregular topology algorithms are generally more complex than those for regular topologies and a high price may be paid even for small irregularities. Therefore, customized interconnect layout may decrease possibilities of using the most efficient routing algorithms.

Even though basic regular topology algorithms may be relatively efficient, deadlock freedom requirements prohibit full utilization of available packet routes. Special techniques may increase the amount of available routes, but their implementation requires additional communication resources. General deadlock-free routing algorithms are designed assuming worst case communication patterns. However, in the case of NoC it is likely that more information about the communication is known. If not all possible communications are considered in the design of the routing algorithm, deadlock freedom requirements can be relaxed and more routes can be allowed without additional network resources.

1.5 Contributions

The framework for the work in this thesis is the Network in Chip (NoC) paradigm. There are mainly two issues that are studied. One of those is the consequences of introducing irregularity in symmetric topologies, such that size-varying cores are handled more efficiently. The second issue relates to optimization possibilities due to the application specific characteristic of on-chip communication. In particular the main contributions are:

- Analysis of the impact of allowing *regions* in regular networks. A region is an oversized core slot that enables use of large resources or encapsulation of components. A network with regions can be characterized as partially regular.

- Development of an improved version of a previously published fault-tolerant routing algorithm. This routing algorithm allows deadlock free routing in networks with regions.
- Development of a new application specific routing algorithm methodology for NoCs. The methodology targets performance loss incurred by requirements of deadlock free routing. By using knowledge of application communications, assumptions for deadlock freedom can be relaxed and routing performance improved.

Contributions are mainly based on work published in conference proceedings and journals (Holsmark & Kumar 2005) (Holsmark et al. 2006) (Palesi, Holsmark & Kumar 2006) (Holsmark et al. 2008) (Holsmark & Kumar 2007) (Palesi et al. 2009).

1.6 Thesis Layout

This thesis is organized as follows. Chapter 2 provides background knowledge to the work in the thesis. It includes topics like SoC, network concepts and related work. Chapter 3 presents an elaboration of the region concept. After this, Chapter 4 describes and analyzes two routing algorithms developed for routing in partially regular NoC.

The impact of regions on network performance is investigated in Chapter 5. A comparative evaluation of the two algorithms, presented in Chapter 4, is performed in Chapter 6. Conclusions and proposals of future work are given in Chapter 7. The thesis also includes an appendix that discusses and proposes equations for calculating average distance in mesh networks.

2

Background and Related Work

Network on chip (NoC) is a new design paradigm with a network oriented approach towards the interconnect problems in Systems on Chip (SoC). SoC interconnect has generally not been designed as networks. Instead can these interconnects to a large extent be characterized as ad-hoc structures, which have evolved closely with electronic system components. As the capacity of silicon chips has increased, the efficiency of the traditional SoC interconnect technologies has decreased.

This chapter begins with some historical notes on the evolution that formed the practices of electronic systems design. After this follows brief discussions on design complexity, core based design and the interconnect problems that motivated the initial proposals on NoC.

Knowledge of communication networks is an essential background for work in the NoC area. Consequently, this chapter provides an overview of important network concepts. The focus is set on topics close to the work in this thesis, like routing algorithms, switching techniques and network deadlocks. The main NoC motivations from a few of the first NoC research papers are also presented. The chapter is finalized with a short survey on research proposals which are the most related to the work in this thesis

2.1 Chapter Overview

The material in this chapter is organized in two main categories. The category which is located mainly in the first sections provides basic knowledge of a broad area. The other category gives more specialized information, which is closer related to the thesis contributions. The following is an ordered gross outline of the chapter contents:

System on Chip
History, Design complexity, Implementation aspects

General network concepts
Topology, Routing, Switching, Network routers

Selected network topics
Wormhole switching, Deadlock free routing

Network on Chip
Motivation, Proposals, Design parameters, Evaluation

Related work
Irregular mesh NoC architectures and routing algorithms

The following sub-section gives a short historical resume over the progress of digital electronic systems. Though being a brief and simplified story, it hopefully provides some background to the current status of the area.

2.1.1 The Road to Networks on Chip

Long ago, electronic circuits were formed solely by discrete components interconnected by visible wires. These circuits performed, at that time, amazing tasks, like lighting up a lamp or enabling communication with telegraphy. Even more remarkable tasks were performed with the arrival of the vacuum tubes (Spangenberg 1948). Striking calculations were then possible with the first generation computers, e.g. “Eniac”, but they consumed enormous amounts of energy.

When the transistor came, it could replace the power hungry vacuum tubes. The transistor, built from semi-conducting silicon, enabled completely changed circumstances. Electrons were basically facing a new less obstructive world and the same operations could be performed with much smaller amounts of energy. The tiny transistor was quickly put to use and it became an important component in both digital and analogue electronics.

The transistor marked the start of an evolution of increasingly powerful electronic components. As illustrated in Figure 2-1, the next in line to hit the markets, was the *integrated circuit (IC)*. The IC technology enabled chip integration of several transistors and other components like resistors and capacitors. Each component in an IC is built by

modifying chemical characteristics of a small piece of a few mm^2 sized silicon plate (chip). A metallization process then creates the wiring that interconnects the components.

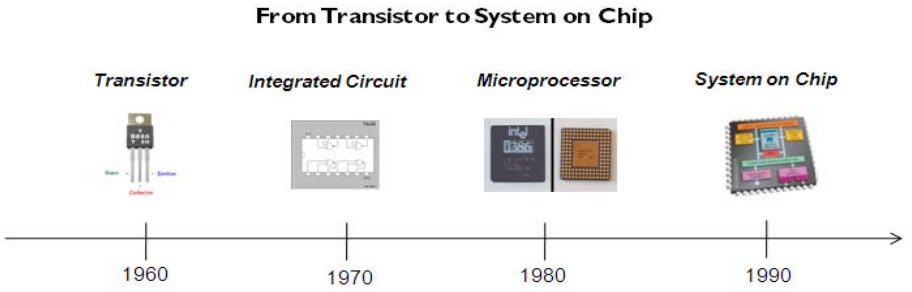


Figure 2-1: Timeline over some important electronic system design components

An effect of the integrated circuit was that significant functionality of electronic circuits was encapsulated and became new discrete "standard" components. Sadly, some would say, large parts of the art of circuit design became invisible to the human eye. Further readings on the transistor and the integrated circuits can be found in (Riordan 2004) and (Kilby 2000).

The first high volume digital circuits were various types of standard digital gates. MOSFET transistor technology (Mead & Conway 1979) offered further improvements, which are still seen in, for example, the rapid capacity increase of single chip microprocessors (Arns 1998) (Tredennick 1996). A second effect was that, then in the early 80's, customers could design an *application specific* IC (ASIC) and let a factory produce the chip. Being application specific, they allowed for optimizing parameters like performance or power consumption

A middle road technology between general purpose processors and ASICs, are the programmable logic devices, for example FPGAs. Note that the definition of ASIC can vary, as some may categorize FPGAs as ASICs (Banker et al. 1993).

As a result of market requirements and increased chip capacity, technologies and concepts have merged and classification boundaries are now less clear. ASICs include micro-processors, micro-processor chips carry custom hardware units and programmable logic devices are boarded by both processors and optimized hardware. To summarize; what used to be a system of discrete chips became a *system on* a single *chip*. The SoC concept was born.

Over the years the transistor size has shrunk dramatically and today the number of transistors on a single chip is counted in hundreds of millions. Designing systems that fully utilize such huge capacities is a challenging task. As chip capacity increased even further, efficiency of the available SoC design methodologies decreased. Therefore, in the year of 2000 some researchers proposed a new paradigm for SoC design, called Networks on Chip.

2.2 SoC: Managing Complexity in a Small World

The SoC concept grew from a constant desire to reduce size and increase performance of electronic systems. An essential prerequisite for SoC was the manufacturing technology that enabled integration of discrete circuit board components into a single chip (Birnbaum & Sachs 1999).

Reduced design complexity is one of the main issues for SoC (Kucukcakar 1998). Even though the transistor capacity of chips increase at a high rate (rate often referred to as *Moore's Law* (Moore 1965) (Moore 1975)), the design productivity does not increase accordingly. As noted in (Henkel 2003), the existing design methodologies do not allow designers to fully utilize all the available transistors.

One SoC design issue which has grown in importance is power consumption (Gries 2004). For battery operated devices, it directly affects the usability. Also heat management is a difficult task in advanced chips (Collins 2003). Economical realities have a strong impact, especially in issues related to implementation of the system in a silicon chip. The magnitudes of the issues in turn depend on the targeted implementation technology (Henkel 2003).

The following sub-sections present some basic ideas and concerns related to design and manufacturing of SoC.

2.2.1 Cores and Core Based Design

SoC is often linked to a design-style based on interconnecting cores (Kucukcakar 1998). Core-based design is a modular approach, where design complexity is reduced by means of well specified blocks (cores) and interfaces.

A core is in itself a more or less advanced electronic system with a specified functionality. Typical cores are general purpose processors, digital signal processors, memories and special purpose hardware units. Cores may be sold as separate designs by specialized design companies that do not manufacture chips themselves. Several free (open source) cores are also available from individuals and organizations. Cores that are not sold as physical components are often referred to as IP-cores (Intellectual Property).

A core can be specified and traded in varying degrees of detail or abstraction levels. Common terms in this respect are soft cores, firm cores and hard cores.

- Soft core – functional specification (hardware description language e.g. VHDL)
- Firm core – structural specification (components, net-list)
- Hard core – complete physical design specification

A soft core can be seen as a functional specification (Dey et al. 2000). These are usually described in a hardware description language like VHDL. Soft cores contain no information about their physical implementation. A firm core includes additional

information about the internal component structure and interconnect. A hard core contains a complete design specification down to transistor layout.

Seamless interfacing of heterogeneous cores is critical for an advanced SoC. Several standardized interfaces were developed to support integration of components at the core level. The main core-interconnect technologies for current industry SoCs are bus systems or point to point connections (Pasricha et al. 2008) (Goossens et al. 2004) (Loghi et al. 2004) (Lahiri et al. 2001). Several IP-cores are equipped with interfaces for industry-standard bus architectures, such as AMBA, CoreConnect or Wishbone. A comparison of bus architectures is given in (Kyeong Keol Ryu et al. 2001)

Use of platforms is one way to further amortize the high costs of developing and producing SoCs. A platform usually consists of hardware cores, interconnect and software that can be heavily reused over several applications (Keutzer et al. 2000).

2.2.2 SoC Example: Advanced Set-Top Box Application

An informative case study of an Advanced Set-top Box (ASTB) SoC from Philips is described in (Goossens et al. 2004). The communications within this system have highly varying requirements, with respect to QoS, data-rates, latency and jitter. The functionality of different parts also exhibits varying computational requirements, since for example, audio processing is much less demanding than video processing. Design requirements, like possibilities of reuse and product differentiation, are best supported by a programmable system.

A block model of the system architecture is shown in Figure 2-2 (Goossens et al. 2004). The combination of functional and design requirements results in a system platform with a large mix of elements. There are three processors: one MIPS and two TriMedia VLIW DSPs (TM32). There are also 60 function specific cores, e.g. mpeg2 decoders, audio video I/O and peripherals like UART and USB. The memory requirements are too high for using only on-chip storage, therefore data and instructions for TriMedia processors are merged to one off-chip memory.

Three different types of interconnect are used for system communication. A bus structure (M-DCS and T-DCS) is used for traffic with low data-rate, but with requirements of low latency. The other two interconnects connect to the external memory. One of these is dedicated for processor cache misses. The other is used by the cores and is implemented as a pipelined multiplexed connection (PMA).

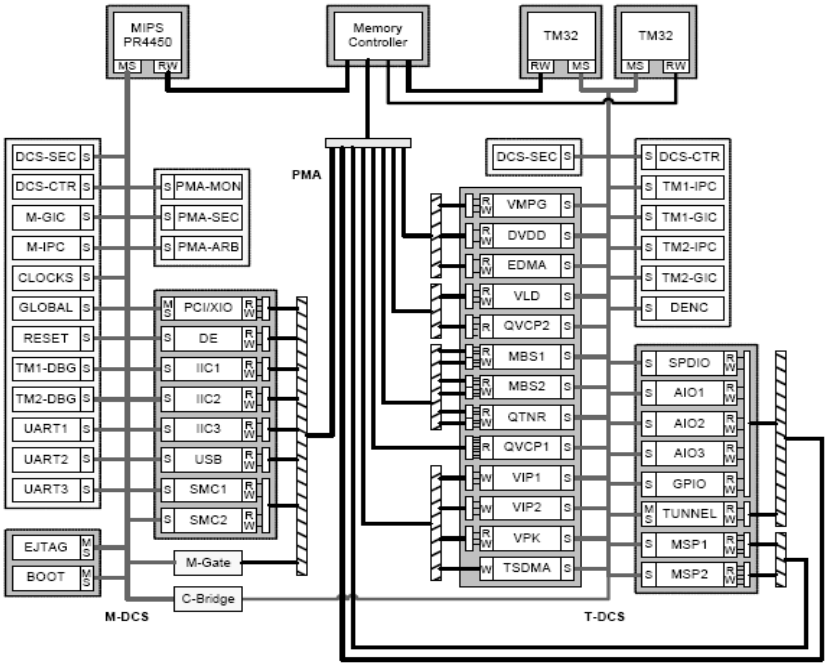


Figure 2-2: Organization of cores and interconnect in Viper 2 multimedia SoC platform (Goossens et al. 2004)

2.3 Chip Manufacturing Technology

The large capacity of silicon chips is a result of advanced manufacturing technology. Though the size of a chip has not changed significantly over the years, sizes of the basic electrical components have.

This evolution is often, or at least used to be, described in terms of scale integration. The increasing number of transistors on a chip was indicated from SSI (Small Scale Integration) to LSI (Large SI) and the long lasting era of VLSI (Very LSI). This terminology seems to have decreased in popularity, even though ULSI (Ultra LSI) sometimes can be seen when referring to the latest technology.

Currently, it is common that the technology level is indicated by the size of the transistors. In this respect the term submicron (sub-micrometre) refers to sizes smaller than 1 μm. Today 65 nm is the standard technology for sophisticated chips. Some of the most advanced processor manufacturers have also started manufacturing with technologies at 45 nm. The small component geometries require extreme precision tools and even the smallest dust particle may destroy the delicate circuits.

Chip manufacturing is characterized by large initial investments but low unit production costs. The reason is that it is very expensive to design advanced circuits and to set-up the necessary equipment for manufacturing the chips. Once this is done, the additional cost of each chip is low. But, high volumes or very price-insensitive customers are required to cover the initial costs.

Programmable logic devices, like FPGAs, are an alternative to implement electronic circuits on chip while avoiding high initial costs. An FPGA is flexible in the sense that it can be (re-)programmed and implemented by simply connecting the device to a computer. Nevertheless, hard silicon chips can be more optimized, enabling lower unit cost, higher performance and lower power consumption as compared to an FPGA.

2.3.1 The Flexibility vs. Performance Trade-off

As in all commercial activities, design and production of SoC is guided by one strong motivator: maximum return of invested money. As usual, customers constantly require lower cost and higher performance. This has led to an interesting trade-off between flexibility and performance in the SoC industry. Usually, both properties are desired but unfortunately highly conflicting.

To explain this conflict, we can start by analyzing the implied meaning of system flexibility and performance.

Flexible systems: A flexible system is easy to improve, adapt and reuse. To reduce high development costs and shorten time to market, companies favor flexible systems. This often requires that top performance is sacrificed for achieving decent performance on average.

High performance systems: The main argument for a system *user* is not how the system is designed, but rather how it performs. To be high performing (except in terms of flexibility) usually requires a high degree of specialization and optimization

There is a huge variety of options for flexibility performance trade-offs in an electronic system. A main trade-off is between software or hardware implementation of functionality. This trade-off is theoretically fundamental because it relates to the actual purpose of the choices. A programmable processor is just hardware organized to allow flexible use and can (at least in theory) never reach the performance obtained with dedicated hardware.

In reality, the gap is narrowing from both ends. From the software end, customers (electronic system designers) require flexible systems; therefore general processors sell in huge numbers, enabling large investments to design high performance processor hardware. On the hardware end comes the programmable FPGAs, whose general nature enables large sales and high development costs, resulting in high performance and capacity. As noted by Rabaey (Rabaey 2005), the number of successful ASIC projects is declining in favor of flexible software, as a result of the large production investment costs ($1M, 0.13um CMOS).

2.3.2 Physical Challenges in Integrated Circuits

As sizes have shrunk, the impact of physical realities on circuit behavior has become higher. Deep submicron usually implies structures below 130 nm and indicates that special physical problems must be considered by the designers. The adverse effects are related to both wiring as well as components.

The impact of physical laws on IC interconnect design is described in (Davis et al. 2001). The paper outlines basic fundamental limits for achievable performance, energy consumption and material properties. Technological limits are also considered where, for example, 3-D chips are proposed to solve the foreseen unrealistic number of metal layers required for interconnect in 2-D chips.

Effects of small material changes, process variations, grow as the design elements get smaller. One example in this area is given by (Ashouei et al. 2006), who propose techniques to handle leakage currents caused by process variations.

Whereas downscaling has positive effects on transistor delay, it has negative effect on global interconnect performance. This problem can be mitigated by the use of extra thick wires, which on the other hand impacts negatively on wire routability. As shown by (Man Lung Mui et al. 2004), long wires may even be the limiting factor for SoC performance. Alternative interconnects, like optical and RF technologies are outlined in (Havemann & Hutchby 2001), which also note the promise of 3-D chips.

Several physical SoC interconnect complications are described in (Nurmi et al. 2005). Faster clock speeds and changed wire geometries increase noise levels in the chips. Low power requirements in turn make wires more susceptible to noise and increase the risk of signal degradation. The current techniques, in which these problems are handled in an iterative mix of pre-layout estimation and more accurate post-layout analysis, are becoming less efficient. It may even be necessary to accept erroneous signals and, instead, apply higher level fault-tolerance techniques to achieve the necessary yield and performance. Yield and manufacturing chips are further discussed in (Yu Cao et al. 2003).

Even more complications than the ones discussed are expected as miniaturization requirements push for integration of both analog and digital circuits in a single SoC (Levin & Ludwig 2002) (Rabaey et al. 2006).

2.4 Terminology and Concepts of Computer Networks

NoC research inherits many concepts and design ideas from the area of computer networks. This is natural since NoCs by name are also networks, though on a smaller scale. This section gives an overview of some network terminology and concepts, which can be found for example in (Duato et al. 1997) or (Culler et al. 1998).

2.4.1 Topology

Topology is an abstract representation of network structure. It is usually defined as a graph, where the edges are the network links and the vertices are the network router (switch) nodes. Topologies are commonly classified as being either regular or irregular. Irregular topologies can be of any shape, whereas regular topologies are characterized by a uniform and homogenous structure.

The regular topologies star, torus, ring and mesh are shown Figure 2-3. The size of meshes is commonly given in the form *R X C*, where *R* represents the number of node rows and *C* the number of node columns. If the number of nodes in both dimensions are equal i.e. *R=C*, two and higher dimensional mesh structures are often described as *n*-dimensional *k*-ary, where *k* represents the number of nodes in each of the *n* dimensions. Examples of some other regular topologies are cube, tree, butterfly, and hypercube.

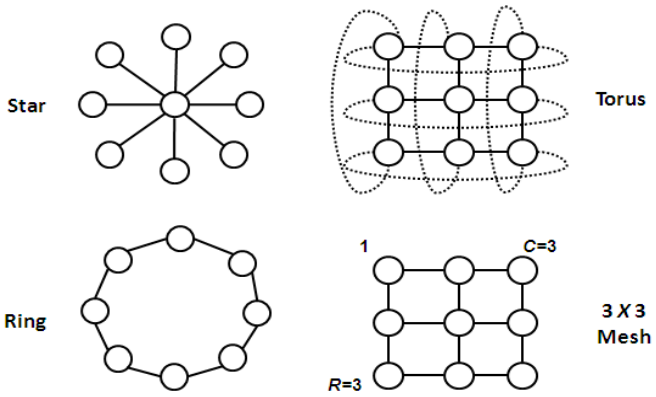


Figure 2-3: Examples of some regular network topologies

Advantages and disadvantages of regular topologies have been extensively studied. Several parameters affect performance in different topologies; both network architectural parameters, like bisection bandwidth and inter-node channel width, as well as different traffic scenarios, like communication type or locality of communication.

2.4.2 Routing

Routing is the mechanism that determines message routes, i.e. which links and nodes each message will visit from a source node to a destination node. A *routing algorithm* is a description of the method that determines the route. Common classifications of routing algorithms are:

- Source vs. Distributed routing
- Deterministic vs. Adaptive (Static vs. Dynamic) routing
- Minimal vs. Non-minimal routing
- Topology dependent vs. Topology agnostic routing

Source vs. Distributed routing

This classification is based on where the routing decisions are made. In source routing, the source node decides the entire path for a packet and appends it as a field in the packet. After leaving the source, routers switch the packet according to the path information. As routers are passed, route information that is no longer necessary may be stripped off to save bandwidth.

Source routing allows for very simple implementation of switching nodes in the network. However, the scheme does not scale well since header size depends on the distance between source and destination. Allowing more than a single path is also inconvenient using source routing.

In distributed routing, routes are formed by decisions at each router. Based on packet destination address, each router decides whether it should be delivered to the local resource or forwarded to one of the neighboring routers. Distributed routing requires that more information is processed in network routers. On the other hand, header size is smaller and less dependent on network size. It also allows for a more efficient way of adapting the route, depending on network and traffic conditions, after a packet has left the source node.

Deterministic vs. Adaptive routing

Another popular classification divide routing algorithms into deterministic (oblivious, static) or adaptive (dynamic) types. Deterministic routing algorithms provide only a single fixed path between a source node and a destination node. This scheme allows for simple implementation of network routers.

Adaptive routing allows several paths between a source and a destination. The final selection of path is determined at run-time, often depending on network traffic status.

Minimal vs. Non minimal routing

Route lengths determine if a routing algorithm is minimal or non-minimal. A minimal algorithm only permits paths that are the shortest possible, also known as profitable routes. A non-minimal algorithm can temporarily allow paths that in this sense are non-profitable. Even though non-minimal routes result in a longer distance, the time for a packet transmission can be reduced if the longer route allows for avoiding congested areas. Non-minimal routes may also be required for fault-tolerance.

Topology dependent vs. Topology agnostic routing

Several routing algorithms are developed for specific topologies. Some are only usable on regular topologies like meshes, whereas others are explicitly developed for irregular topologies. There is also a specific area of fault-tolerant routing algorithms. These are designed to work if the topology is changed by faults, where for example a regular topology is turned into an irregular topology.

2.4.3 Switching

The switching technique determines how network resources are allocated to a message on a route between a source and destination node. The basic techniques are circuit switching and packet switching. Circuit switching allocates all necessary resources on a source-destination route before sending a message.

Packet switching iteratively allocates one link until reaching the destination. Common terms related to *packet switching* are:

- Store and Forward, Wormhole and Cut-through switching
- Buffers and Virtual Channels

Store and Forward, Wormhole and Cut-through switching

These techniques are related to whether network flow control is based on packets or flits. Flit (flow control unit) here means a part of a packet. Store and forward exhibits flow control on packet level, where a packet must be completely received in a node before transmission to the next node is started.

In wormhole switching, flow is controlled on flit level. Packets are transported between network nodes on a flit by flit basis, where the first flit (header) determines the direction. If the desired output in a router is free, the connection is locked and the header is forwarded. The rest of the flits follow the locked route and when the last flit (tail) has passed it releases the lock.

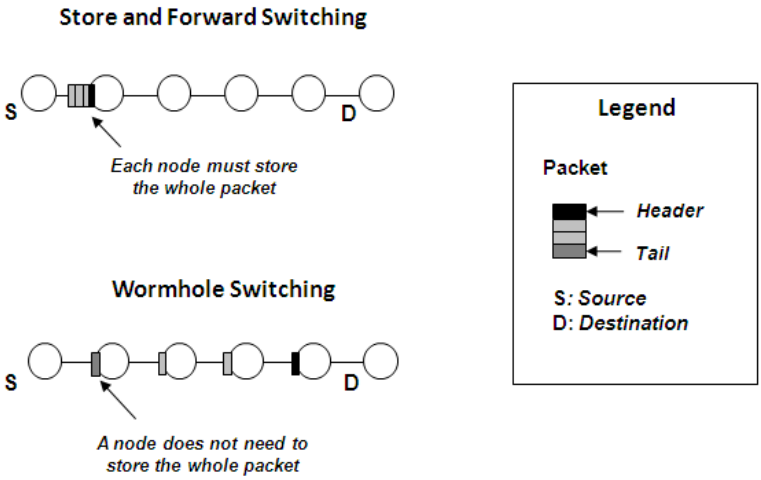


Figure 2-4: Examples of store and forward and wormhole switching

An example of store and forward and wormhole switching is given in Figure 2-4, where a packet is transmitted from source to destination using either store and forward switching or wormhole switching. Using store and forward, the first node must receive a whole packet before transmission to the next node can be performed. This is not necessary when using the wormhole technique, where a flit can move on even though the whole packet

has not arrived to a node. As can be seen, the flits of a packet proceed towards the destination in a pipeline fashion.

Therefore, wormhole switching is advantageous for two reasons. First, it is only necessary to keep buffers large enough to carry one flit in a node. Second, packet throughput and latency is improved because the packet is transported in a pipelined manor. A drawback with wormhole switching is that the risk of contention increases, since one packet may occupy several routers and links.

Cut-through is a switching technique in-between store and forward and wormhole. This technique allows packets to be forwarded on a flit by flit basis. Still, each node *must be able* to store a whole packet, similar to store and forward. Consequently, this technique has similar buffer requirements to store and forward, but latency and throughput characteristics closer to wormhole switching.

Buffers and Virtual Channels

Buffers are usually implemented in network routers to reduce effects of temporary variations in traffic intensity. Input buffers store arriving flits waiting for available output channels. Output buffers store packets already assigned output channels but waiting for availability of next router inputs. FIFO at each router port is a common buffer strategy, although buffers shared by all ports may provide higher buffer utilization efficiency.

The virtual channel concept is a method to partition a physical channel into several logically separated channels. This is accomplished by assigning a separate buffer to each virtual channel. Since the physical link is shared, the bandwidth of each virtual channel will be reduced. Nevertheless, virtual channels can be efficient for performance improvement in wormhole switched networks.

This is because they can release paths that are otherwise blocked for packets, due to other packets in front of them. By sharing the channel, several packets can advance simultaneously, though each at a lower data rate. Figure 2-5 gives an example of such a situation.

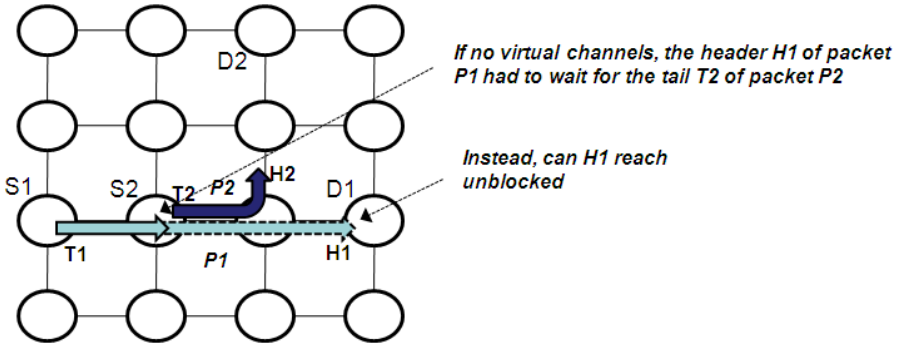


Figure 2-5: Virtual channels for resolving contention on channels

The figure illustrates two packets in a network. Packet P1 with header H1 and tail T1 and packet P2 with header H2 and tail T2. P1 travels from source S1 to destination D1 and P2

from source S2 to destination D2. The existence of two virtual channels, allows for P1 to proceed unblocked to D1. If virtual channels were not used, H1 would have to wait for T2 to release the blocked path.

2.4.4 Quality of Service

An important property of a network is that it provides fair and uniform performance to traffic of equal priority in the network. For networks used in real time systems, it may be necessary that routing schemes can provide latency, throughput or some other quality of service (QoS) guarantees to selected groups of communications. These guarantees may be implemented by providing different priorities to different types of traffic or by implementing special mechanisms to reserve network resources.

2.4.5 Deadlock, Livelock and Starvation

Three properties which are necessary in all usable communication networks are freedom from deadlock, livelock and starvation:

Deadlock - Packets are involved in a circular wait that cannot be resolved

Livelock - Packets wander in the network for ever without reaching the destination

Starvation - Packets never get service in a router

Several strategies can be applied to handle deadlocks. Deadlock avoidance techniques, for example, ensure that deadlock never can occur. Deadlock recovery schemes, on the other hand, allows formation of deadlocks but resolves them after occurrence.

Livelock may occur in networks with non-minimal routing algorithms, i.e. where packets may follow paths that do not always lead them closer to the destination. A common solution to livelock is to prioritize traffic based on hop-counters. For each node a packet traverses, a hop counter is incremented. If several packets are requesting a channel, the one with the largest hop-counter value is granted access. This way, packets that have long circled the network will receive higher priority and eventually reach the destination.

An example of starvation is when packets with higher priorities constantly out-rank lower priority packets in a router. As a result, the lower prioritized packets are stopped from advancing in the network. Starvation is an important aspect when designing the router arbitration mechanism.

2.5 Deadlocks and Wormhole Switching

Though being more efficient than store and forward, the wormhole switching technique is more prone to create deadlocks in a network. This is because a packet is allowed to hold several network resources, while requesting use of others. Figure 2-6 exemplifies a deadlock situation in a network.

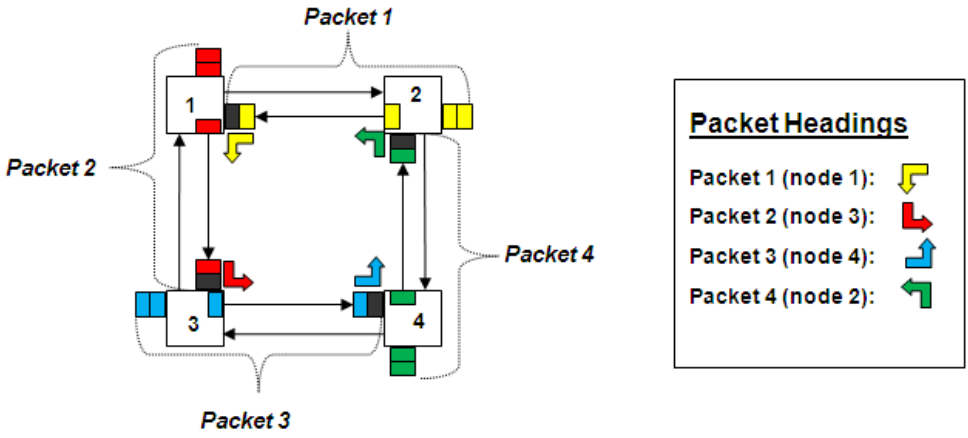


Figure 2-6: Four packets in a deadlock situation

Packet 1 wants to turn south at node 1, but is blocked by Packet 2 that stretches through node 1. Packet 2 requests turning east at node 3 but is blocked by Packet 3. Packet 3 requires a north turn but is stopped by Packet 4. Packet 4 needs to turn west but is blocked by Packet 1.

Undoubtedly, there is a cyclic dependency among the packets such that none of the packets can proceed. The network has entered a state of deadlock, which cannot be resolved without using special mechanisms. This may be costly if deadlocks occur frequently. Therefore, it is an important property of routing algorithms for wormhole switched networks to be deadlock free. The Turn Model and the concept of channel dependency graphs (CDG) are two methods for designing deadlock-free routing algorithms.

2.5.1 The Turn Model

The Turn Model is a methodology for designing deadlock-free routing algorithms for *n*-dimensional meshes. The analysis is based on which turns packets are allowed to make in a network. From the example in Figure 2-6, it can be seen that four turns (arrows) are allowed. By using the Turn Model, we could quickly have seen that this configuration was prone to create deadlocks. This is because the Turn Model tells that allowed packet turns must not be able to create an abstract cycle of turns. In (Glass & Ni 1992), Glass and Ni prove the following theorem:

> Theorem (Glass & Ni 1992):"The minimum number of turns that must be prohibited to prevent deadlock in an n-dimensional mesh is n(n-1) or a quarter of the possible turns"

Figure 2-7 shows the possible turns and allowed turns in two-dimensional mesh for X-Y and West-First routing algorithms. The possible turns form cycles and allowing them all would enable packet deadlocks. In X-Y routing, some turns are disallowed (un-shaded in

Figure 2-7) and a packet must first proceed in the horizontal (X) direction until it reaches the column of the destination. Then it continues vertically (Y) to the destination. As can be seen from the allowed turns of X-Y, it is not possible to produce a cycle using these turns. According to the Turn Model this routing algorithm is deadlock free.

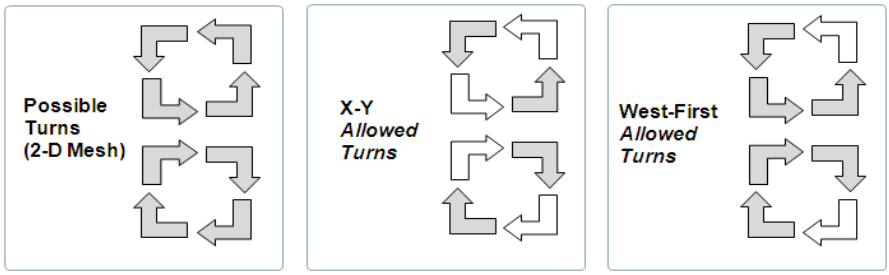


Figure 2-7: Possible turns in 2-D mesh and allowed turns for X-Y and West-First routing algorithms

The partially adaptive West-First routing algorithm (Glass & Ni 1992) allows a few more turns than X-Y. The turns of West-First in Figure 2-7, reveal that multiple routes are possible for all packets except those with destination towards west (left), which have to proceed according to just West-First. Otherwise it will not be possible to reach the destination, because turns from a vertical route to the west are not allowed.

2.5.2 Channel Dependency Graphs

Another technique for deadlock analysis is the use of channel dependency graphs (CDG). A CDG is constructed from the network topology, where the network channels (links) are the vertices of the CDG. There is an arc (dependency) between two channels if a second channel can be used immediately after the first.

Let us use the topology of the example in Figure 2-6, and annotate the channels according to Figure 2-8(a), such that a channel l_{ij} connects a node i with a node j. Note that only one of the two directions between each node is used in the example. The corresponding CDG is shown in Figure 2-8(b). The cycle in the CDG indicates that deadlock can occur.

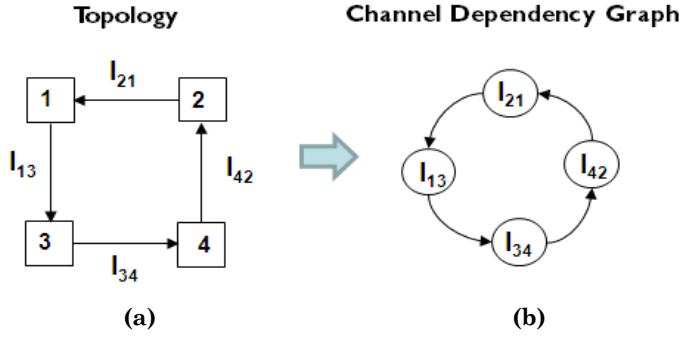


Figure 2-8: Topology and CDG: (a) channel labeling and (b) corresponding channel dependency graph (CDG)

If the CDG in this example had been acyclic, routing had been deadlock free. In (Dally & Seitz 1987), Dally and Seitz proved the following theorem:

> Theorem (Dally & Seitz 1987): "A routing function R for an interconnection network I is deadlock free iff there are no cycles in the channel dependency graph D."

Duato (Duato 1993) extended the theorem to adaptive routing algorithms and also noted that deadlock freedom by acyclic CDGs is in fact a sufficient but not necessary condition.

A CDG for X-Y routing applied to the example network would look like Figure 2-9. There is no arc between I_{42} and I_{21} because it is not possible to use I_{21} immediately after I_{42} using X-Y routing. The same is true for I_{13} and I_{34}. Therefore, no cycle exists in the graph, and, consequently, this routing algorithm is deadlock free.

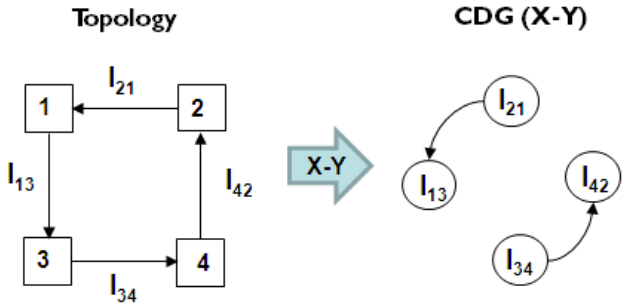


Figure 2-9: Topology and CDG for X-Y routing

2.5.3 Other Techniques to Handle Deadlocks

The concept of Channel Wait for Graphs (CWG) (Jayasimha et al. 1996) is similar to CDG but less restrictive. The reason is that a CWG capture information which is not used in a CDG. Therefore, it is possible that a routing algorithm which produces a cyclic CDG is deadlock free if it shows an acyclic CWG. A thorough study on deadlock-free routing is found in (Fleury & Fraigniaud 1998).

In the basic deadlock avoidance techniques, deadlocks are prevented by enforcing routing restrictions, either on turns or arbitrary channel traversals in a CDG. However, restricting routes decreases adaptivity. Virtual channels can reduce this negative effect by increasing the number of available channels and separating traffic. In (Duato 1993), Duato shows that highly adaptive routing algorithms can be designed using the virtual channel technique. This is achieved by providing a routing sub-function, which is deadlock free and virtual channels dedicated for this purpose.

Using deadlock-free lanes (Anjan & Pinkston 1995) is an alternative similar to virtual channels, but it is based on deadlock recovery instead of deadlock prevention. It requires few resources to support deadlock safe routing, since it is only necessary to keep a single extra flit buffer at each node. However, they are restricted to be used only in the case of deadlock occurrence.

2.6 Network Routers

Topology, routing and switching affect the design of network routers. Different techniques require different router functionality, which in turn affects performance. The basic task for a router is to switch an incoming message to an output channel. This task can be partitioned into the sub-units that are shown in the general router structure in Figure 2-10.

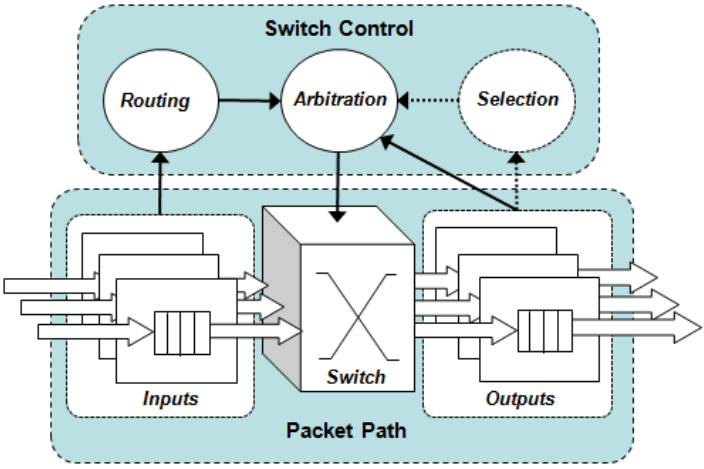


Figure 2-10: General structure of a network router

In the packet path, the inputs receive packets sent from neighboring router outputs. Before traversing the switch, the correct route for each packet must be determined by the routing function. The routing function decodes arriving packets for route information, e.g. destination address, and determines the correct output. Arbitration is performed if there are multiple requests for the same output.

If routing is adaptive, it is possible that the routing function returns multiple allowed outputs. Then, a selection function must be invoked to select the preferred output, depending on, for example, buffer levels. Once a selected output request is granted, a packet will traverse the switch to the output. The output transmits the packet to a neighbor router input under the control of a given transaction protocol. Note that concrete architectures may vary significantly due to system requirements.

2.6.1 Routing Function

The routing function returns the allowed output channels for messages arriving at router inputs. Note that this function is *not necessary* for source routing. The information that is necessary for routing decisions determines the domain of the routing function. For example, the routing function, $R: N \times N \rightarrow \wp(C)$ (Duato 1993) returns the admissible output channels $c \in C$ from the current node $n_c \in N$ and the destination node $n_d \in N$ of a packet. Other routing functions may be defined over input channels instead of the

current node (Dally & Seitz 1987). Additional information like source address can also be used for routing decisions.

When there is only one path between each source and destination node, i.e. $Size\big(\wp(C)\big) = 1, \forall R$, the algorithm is considered to be static or deterministic.

2.6.2 Arbitration and Selection

It is possible that there are several concurrent requests for an output in a router. Then, an arbitration function determines which of the requests should be serviced. Several different schemes can be used for arbitration, for example round-robin or priority policies (Culler et al. 1998).

In deterministic routing, only one output can be selected by one input. But adaptive routing functions may return multiple outputs i.e. $\exists R: Size(\wp(C)) > 1$. In this case, the selection function determines the preferred output. There are several techniques that can be used, for example, making a (pseudo) random decision or taking a decision according to a favored dimension. Selection decisions can also be based on output buffer states or information of congestion in certain directions.

It should be noted that the arbitration and selection functionalities are interdependent, and the order in which they are performed affects the performance.

2.7 Router Architecture and Trade-offs

The cost of a routing scheme is reflected in the implementation cost of the router. Generally, there is a trade-off between cost and performance, implying that routing schemes providing higher performance are costlier as compared to routing schemes with lower performance.

The main design frame is given by the complexity of topology, routing algorithm and switching technique. Within this, trade-offs can be made to customize the architecture. This section discusses some of the available choices.

2.7.1 Buffering Strategy

Not only routing and switching affect the performance of a network. Another important design choice is what type of queuing strategy the router is equipped with (Hluchyj & Karol 1988) (Jiang Xie & Chin-Tau Lea 1999). The most common is to use routers with input queuing, where packets are buffered before routing and selection takes place. Usage of input-buffers is logical when the switch fabric operates at the same speed as the input units, something that is common in crossbar based switches (Jiang Xie & Chin-Tau Lea 1999).

However, simple input queuing suffers from head of the line blocking, which results in that the performance of a router saturates at about 60% of its nominal capacity. If the switch fabric is fast enough to empty all input channels in one cycle, output queuing

should be used. With this strategy, router performance is able to reach its full potential. There also exist techniques to improve the performance of input-buffer based routers.

2.7.2 Algorithmic vs. Table Based Routing

The complexity of topology and routing algorithm has a strong impact on routing function implementation possibilities. Routes in regular topologies are more easily expressed by small functions as compared to routes in irregular topologies. Therefore, algorithms for regular topologies have a higher probability to be implemented by hardware logic functions. This is also referred to as algorithmic routing. Combined with simple routing algorithms, like X-Y, this enables implementation of small and fast routers in the network. The drawback of algorithmic routing is that the routing function cannot be changed after manufacturing (unless implemented using programmable logic).

Irregular topologies are harder to capture by small logic functions. Therefore, routing algorithms for irregular topologies are strong candidates for table-based implementation. Using a table also gives the opportunity to implement more complex routing functions on regular topologies. A disadvantage is that a table can take large space if many destination addresses should be stored. For a mesh network of size $N \times N$, N^2 rows will be necessary for the table. If there are k bits for each row, a naive implementation will require kN^2 bits of storage for the table. Compression techniques can in some cases though, reduce the size required by the tables (Palesi et al. 2006).

2.7.3 Selection and Arbitration Complexity

Compared to a router for a deterministic routing algorithm, the path decision functionality in a router for an adaptive routing algorithm adds latency to a packet header's path. This is because the selection function is serially dependent on the output of the routing function. Since the operations can be pipelined, this does not necessarily affect the throughput. Look-ahead routing (Vaidya et al. 1999) is a technique to alleviate the serial dependency between routing and selection. The idea is to perform the routing function one node in advance such that only selection has to be performed at the current node, for a specific packet. This method effectively reduces the packet header latency through an adaptive router.

A selection function in its simplest form does not consider dynamic information. More elaborate selection architectures are needed for dynamically based policies. Still, congestion aware techniques, e.g. in (Ascia et al. 2006), needs extra hardware support throughout the network. Efficient arbitration involves a trade-off between matching quality and implementation complexity. The iSlip arbitration (McKeown 1999) is one approach to balance the requirements of fairness and resource efficiency.

2.7.4 Routing Parallelism

The crossbar is a common solution for connecting the input and output ports in a router. This allows for simultaneous routing of packets that are headed for non-conflicting

outputs. Routing performance is improved if all route requests are served in parallel. This can be achieved by implementing the routing function at each input link. Such arrangements, however, may have a significant impact on router area if the routing function is complex or requires large tables. It is possible, though, that the actual packet routes do not require that all inputs implement the full table.

2.8 Deadlock-Free Wormhole Routing Algorithms

This section provides an overview of highly referred deadlock avoidance based routing algorithms for wormhole switched networks. In literature they are often called: deadlock-free wormhole routing algorithms.

Most of the material is general to routing in any wormhole switched network, although the material is biased towards low cost solutions. Surveys of wormhole routing techniques are provided by Ni and McKinley (Ni & McKinley 1993) and Mohapatra (Mohapatra 1998). Several aspects on adaptive routing are discussed in (Gaughan & Yalamanchili 1993). Routing algorithms for wormhole switched networks have been classified in many different ways in literature. This section discusses them with respect to their topological scope.

2.8.1 Topology Specific Routing Algorithms

These algorithms are only applicable for a certain topology. Usually this means that regularity is exploited to enable simple routing functions.

One of the simplest deterministic routing schemes is the e-cube or dimension ordered routing, where a packet proceeds according to the dimensions of the network in descending order. For 2-D mesh this is basically X-Y routing. Dally and Aoki (Dally & Aoki 1993) present a technique to combine e-cube with virtual channels.

Several adaptive routing algorithms for 2-D meshes are proposed by (Glass & Ni 1992). These algorithms are called partially adaptive since they do not allow the use of all paths between a source and a destination. For example, the West-First algorithm requires that a message heading west should always proceed in the west direction first. This restriction is a result of the requirement of deadlock freedom.

Odd-Even (Ge-Ming Chiu 2000) is another partially adaptive algorithm for 2-D mesh. It aims to provide more evenly distributed adaptivity compared to the ones proposed in (Glass & Ni 1992).

The concept of virtual channels has been used to design adaptive deadlock-free routing algorithms for n-cubes (Duato 1993). An adaptive routing algorithm for 2-D mesh with minimum number of virtual channels is proposed in (Schwiebert & Jayasimha 1993). An example of a non-minimal routing algorithm for meshes using virtual channels is found in (Dally & Aoki 1993). Load balanced routing on torus topology is presented in (Singh et al. 2004)

2.8.2 Topology Agnostic Routing Algorithms

Topology agnostic routing algorithms are usable on any topology, both regular as well as irregular. In general, irregular topologies imply a table-based implementation of the routing algorithm.

One well referred algorithm is Up*/Down* (Schroeder et al. 1991). Up*/Down* routing restrictions are based on tree-search, where one of the network nodes is labeled as root. The effectiveness of Up*/Down* is mainly dependent on which node is selected as root, and what technique is used to assign the direction of the channels.

Several improvements to the original Up*/Down* have been proposed, for example, in (Sancho et al. 2004) and (Koibuchi et al. 2001). Virtual channels were combined with Up*/Down* in (Silla et al. 1998).

Meija et al. (Mejia et al. 2006) proposed a novel methodology to design deadlock-free routing algorithms, both for regular as well as irregular topologies. It is based on dividing the network into small deadlock-free routing segments which are combined to form a globally deadlock-free routing algorithm. This algorithm does not require any additional virtual channels.

2.8.3 Fault Tolerant Routing

Fault-tolerant algorithms are developed to provide paths to all source-destination pairs even in the presence of faults in the network, while ensuring freedom from deadlocks. The X-Y routing algorithm, for example, is incapable of delivering packets for many pairs if a single link failure exists in a network. Adaptive routing algorithms may cope with a few faults, but are in general not fault tolerant.

Designing a routing algorithm which is efficient, deadlock free and fault tolerant is a hard problem. Therefore, several fault tolerant schemes use virtual channels to provide higher route flexibility. Increased fault-tolerance of e-cube (dimension order) by using virtual channels is proposed in (Boppana & Chalasani 1995) and (Jipeng Zhou & Lau 2000). An adaptive fault-tolerant algorithm for meshes and hypercubes is presented in (Chien-Chun Su & Shin 1996).

The use of virtual channels increases implementation cost as it adds both to resources and to the design complexity. Some researchers have proposed algorithms which do not require use of virtual channels. For example, special routing rules have been implemented to increase fault tolerance of standard algorithms (Jie Wu 2003). Other techniques in this category are provided in (Chen & Chiu 1998) and (Glass & Ni 1996).

2.9 NoC: Addressing the SoC Interconnect Problems

Even though the use of standard buses and interfaces improved reuse and modularity, the fundamental scalability problem of SoC interconnect remains. Several NoC architectures

have been proposed to address the scalability problem. Some of the motivation and challenges for NoC are presented in the following subsections.

2.9.1 Motivation for NoC

The initial papers on NoC emerged in the early years of the 21st century (Guerrier & Greiner 2000) (Dally & Towles 2001). Over the years, numerous NoC related papers have been published. The least common denominator for the motivation of NoC is the inherent limits of ad-hoc SoC interconnect techniques. Some aspects of these limitations are:

Bus performance limits: The shared bus technology cannot support application communication requirements.

Chip design complexity: The combination of unpredictable electrical parameters and complexity of on-chip wiring prohibits the use of high performance circuits.

Design productivity: Efficient reuse of cores is hindered by tight coupling of computation and communication structures.

The most frequent motivation for NoC is the scalability problems of bus-based interconnect. Guerrier and Greiner (Guerrier & Greiner 2000) outline that future traffic rates of video, memory-cache and interrupt handling will be un-manageable by bus-based systems. Goossens et al. (Goossens et al. 2004) describe bus limitations in their multimedia SoC and note that the scalability problem is not as severe in packet switched networks.

Chip implementation aspects add additional difficulties for system designers. Dally and Towles, (Dally & Towles 2001) see electrical problems with long global wires that are auto-routed late in the design phase. Therefore, wire driving circuits have to be defensively designed, sacrificing performance for noise tolerance. A NoC with pre-routed wires enables the use of higher performing circuits and provides reduced design complexity. High fixed manufacturing costs is another problem of SoC. Hemani et al. (Hemani et al. 2000) see NoC as a natural extension to the general purpose FPGAs, which allow for reduced unit costs by high volumes of production.

Designing a complex SoC is costly and time consuming. A NoC based on GALS concept simplifies design by enabling efficient IP design reuse (Hollstein et al. 2006). In this respect, NoC provides the possibility of separating the communication and computation domains. This will reduce SoC development efforts with respect to design, test and verification (Kumar et al. 2002).

2.9.2 NoC Design Challenges

Though the networking aspects of NoC are advantageous for several reasons, some important drawbacks exist. Current SoC interconnect techniques have evolved closely with the development of design methodologies, applications and production technology. This section discusses some of the challenges that face NoC based systems.

Resource utilization

SoCs are tightly constrained in that communication components and computation components are directly competing for chip resources. The computation/communication trade-off emerges both in the area as well as the power domains. A NoC in itself is likely to require more chip resources than bus-systems (Guerrier & Greiner 2000) (Dally & Towles 2001). With respect to power consumption, Banerjee et al. (Banerjee et al. 2007) show that NoC routers require significant amounts of energy.

Application requirements

Several SoC applications are real-time systems where constraints on communication are stringent (Goossens et al. 2004). Bus-based interconnects provide fixed, low latency communication once bus access is granted. In this respect is a distributed router network likely to incur higher latency. Without special mechanisms, latency will be non-deterministic and depend on traffic conditions (Guerrier & Greiner 2000).

In order to provide Quality of Service in NoC, Bolotin et al. (Bolotin et al. 2004) label packets into different priority classes with separate buffers. Varying packet priorities are also used in (Daewook Kim et al. 2004). Guaranteed communication services using looped containers is proposed in (Millberg et al. 2004). Goossens et al. (Goossens et al. 2004) highlight the importance of both Guaranteed Throughput (GT) and Best Effort (BE) service classes in NoC. Techniques for enhancing efficiency of GT and BE services are proposed in (Andreasson & Kumar 2005).

Design methodology

The usefulness of NoC is not only dependent on the hardware architecture. An important aspect of NoC based systems is to reduce design complexity and allow the application of core based design. The motivation is strong for reuse of both cores as well as interconnects. Platform system reuse offers the opportunity for diversification of products by changing a few cores or even only software.

Designing complex NoC systems will require good tool support at several design stages. From higher level mapping of applications to NoC resources, to physical chip layout. Examples of mapping techniques are given in (Ascia et al. 2004) (Hu & Marculescu 2005). Design tools for the synthesis of customized NoC structures are presented in (Srinivasan et al. 2006).

2.10 NoC Proposals: Merging Network Concepts and SoC Technology

Many factors affect the overall functionality of a NoC interconnection system. Unfortunately, several of these are conflicting in that optimizing one affects others negatively. One of the key issues is to pair the unique single chip network constraints with an efficient design methodology. There are several aspects on chip resource utilization. One, for example, relates to the infrastructure, i.e. wiring and routers.

Another aspect on resources is the overhead requirements of the communication protocols.

Most NoC proposals consider packet switched communication and well-structured interconnect. However, the principles of the optimal NoC are still under debate. The following sub-sections discuss some of the different viewpoints.

2.10.1 Layout and Topology

NoC resources and communication requirements are not homogenous. In theory, these characteristics favor customized interconnect structures. In reality, this may be in conflict with design and implementation requirements. This conflict has fueled arguments for the optimal NoC interconnect organization; irregular or regular structures.

Irregular interconnect proposals

A customized NoC with an irregular topology is proposed in (Srinivasan et al. 2006). The main motivation is the possibility of achieving optimal resource utilization and performance. Proteo NoC (Saastamoinen et al. 2002) is based on reusable interconnect components that can be used to design a customized NoC infrastructure. Neeb and Wehn (Neeb & Wehn 2006) advocate the use of irregular topologies in the design of NoC systems. The topology is based on a chain, which is grown with edges to suit the communication graph of an application. Bertozzi and Benini (Benini & Bertozzi 2005) address chip wiring delays by pipelined parameterized interconnect. It is shown that the customized interconnect results in lower power consumption and higher performance as compared to a regular mesh interconnect.

Regular interconnect proposals

Dally and Towles (Dally & Towles 2001) argue that regular interconnects, defined at early design stages, provide more control over electrical parameters and enable the use of high performance circuits. Two-dimensional mesh is seen as the most suitable NoC topology. Interconnect reuse motivates an even higher effort in optimization and further performance improvements.

Advantages of 2-D mesh topology, with respect to energy and performance, are acknowledged in (Hu & Marculescu 2005). In (Sung-Rok Yoon et al. 2006), 2-D mesh is used in a case-study on a WLAN receiver. Kumar et al. (Kumar et al. 2002) note that 2-D mesh may suit the anticipated NoC communication patterns with high degree of locality. Simple addressing and multiple routes are also in favor of the two-dimensional mesh.

Several other regular topologies have been proposed for NoC. A few examples are star (Daewook Kim et al. 2004), butterfly (Pande et al. 2003), honeycomb (Hemani et al. 2000) and fat-tree (Guerrier & Greiner 2000).

2.10.2 Routing and Switching

Advanced routing and switching techniques may provide high communication performance. The improved performance usually comes with a higher implementation cost in terms of chip area and power consumption. Different emphasis on cost and performance has motivated NoC routing and switching proposals with varying degree of implementation complexity.

Routing

Hu and Marculescu (Hu & Marculescu 2005) see deterministic routing as most suitable considering overhead and packet reordering problems of adaptive routing. Still, a routing scheme that combines adaptive and deterministic routing is presented in (Hu & Marculescu 2004). Murali et al. in (Murali et al. 2006) propose an energy efficient technique for packet reordering in multi-path routing.

Efficient selection policies may enhance performance of adaptive routing. In their proposal for congestion aware selection, Ascia et al. (Ascia et al. 2006) argue that NoCs are well suited to provide control signals for sensing distant congestion.

Source routing is proposed in (Benini & Bertozzi 2005) (Dally & Towles 2001) due the possibility of fast and simple routers. Deflective or "hot potato" routing was proposed in (Millberg et al. 2004)

Switching, buffers and virtual channels

Wormhole switching is proposed by several researchers, for example (Bertozzi et al. 2005) (Guerrier & Greiner 2000) (Sung-Rok Yoon et al. 2006) (Hu & Marculescu 2005). Nevertheless, the simplicity of store and forward is preferred in a CDMA SoC (Daewook Kim et al. 2004).

Buffers are relatively expensive resources in NoC routers. Therefore, Hu and Marculescu (Hu & Marculescu 2005) see implementation with small fast registers instead of memory as most feasible for NoC routers. Specialized fast FIFO buffers are proposed in (Pande et al. 2003). Neeb and Wehn (Neeb & Wehn 2006) advocate the use of virtual channels to improve adaptivity. Virtual channels are also proposed in (Dally & Towles 2001).

2.11 Evaluation of Network Performance

The dynamic behavior of a network is complex with several interdependent variables. This makes it hard to analytically estimate performance of certain architectural features. Analytical results on network performance do exist. Agarwal (Agarwal 1991), for example, derives performance equations for different configurations of n-dimensional k-ary meshes, considering both with and without contention. An analytical performance model for fast design space exploration is given by (Ogras & Marculescu 2007). Methods that address real-time aspects, such as delay bounds, are proposed in (Qian et al. 2009) and (Manolache et al. 2006).

Simulators are widely used to evaluate and compare routing algorithms. Many NoC research projects develop their own simulation models that suit their evaluation purpose. Sun (Sun et al. 2002) used the network simulator ns-2 for evaluations of a NoC platform. Specific NoC simulators have been developed and used by a number of research projects, for example in (Thid et al. 2003) (Holsmark & Kumar 2005) (Palesi et al. 2009).

Simulation results of different schemes are scrutinized by analyzing certain performance parameters. Some of the most commonly used parameters are:

Packet latency: Time for a packet transmission from source to destination

Throughput: Received packets per time unit

Jitter: Variation in latency or throughput

Note that results are often averaged over all packets received in a simulation session.

Injected traffic patterns

When performing simulations, performance values are often collected with respect to the traffic which is injected to the network. The offered load is the amount of traffic that is released by the sources into the network. This traffic can have various characteristics, depending on the assumed real traffic. An example is the bursty behavior shown in Figure 2-11, where sources are actively generating several (bursts) packets without gap in-between, followed by longer periods of inactivity (gaps). Poisson and self-similar distributions are common models for injected traffic behavior (Kihong Park et al. 1996).

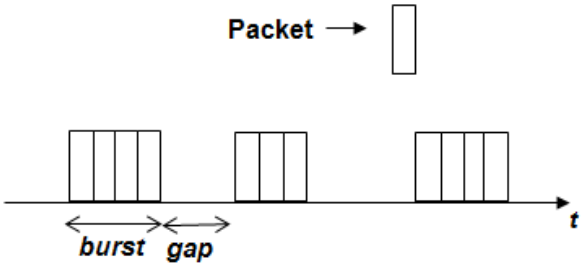


Figure 2-11: Bursty traffic behavior

Distribution of traffic destinations

There are several possibilities to distribute packets among the destinations in a network. Common synthetic traffic patterns are uniform random, random with locality, transpose and hot-spot (Ge-Ming Chiu 2000) (Palesi et al. 2009). Several of these patterns are often used when evaluating a routing algorithm, to investigate performance in different types of traffic scenarios. For example, the static X-Y routing algorithm shows very good performance with uniform random distribution of traffic destinations, as compared to many adaptive routing algorithms like Odd-Even (Ge-Ming Chiu 2000). However, the performance is quite the opposite when hot-spot traffic is considered.

2.12 Related Work

This section presents research which is most relevant to the work in this thesis. The material is concentrated to irregular mesh structures and deadlock-free routing with low resource requirements.

The objective for using irregular mesh (or partially regular) structures is to gain the design advantages of regularity while keeping the resource efficiency of irregular structures. Efficient deadlock-free routing is relatively easy in regular topologies, but harder in irregular. The block fault models used in fault-tolerant routing algorithms for regular topologies are similar to partially regular mesh NoC structures. One such algorithm has been improved and used for routing in this thesis.

Adaptive routing algorithms has several advantages over deterministic. However, the requirements of deadlock freedom significantly reduce the attainable adaptivity for virtual channel free routing algorithms. A routing algorithm that considers application traffic to increase adaptivity, without use of virtual channels, is also an output of this thesis work.

2.12.1 Partially Regular Mesh NoC Architectures

Bolotin et al. (Bolotin et al. 2004) propose a non-homogeneous mesh topology NoC, allowing rectangular cores larger than the mesh tile. Deadlock-free routing is achieved by extending X-Y routing with hard coded paths that are computed off-line.

A middle-path between a customized and a regular NoC structure is proposed by Ogras and Marculescu (Ogras & Marculescu 2006). They propose insertion of shortcut links in a regular 2-D mesh to increase network efficiency and still keep benefits of regularity.

Hu and Marculescu (Hu & Marculescu 2005) consider a 2-D mesh for application and communication mapping on NoC. To reduce waste of resources, the mapping scheme can be extended to handle oversized IPs.

Neeb and Wehn (Neeb & Wehn 2006) propose a regular grid based layout for equally sized core slots. Interestingly, the proposed *topology* is irregular and tailored to suit the application. The routing algorithm is table-based and uses virtual channels for increased adaptivity.

2.12.2 Deadlock Free Routing under Low Resource Requirements

The implementation of routing restrictions for deadlock avoidance impacts on the adaptivity of a routing algorithm. Duato (Duato 1993) proposed a technique which uses virtual channels to enable highly adaptive routing.

Up*/Down* (Schroeder et al. 1991) is a topology independent routing algorithm that does not require virtual channels. Cost may be relatively high though, as a result of table based implementation. The routes returned by Up*/Down* are partially adaptive and decided by an initial tree-search of the topology.

The segment based routing (SR) algorithm by Meija et al. (Mejia et al. 2006) is an alternative that aims for improved route control, as compared to algorithms like Up*/Down*. Although the basic version of SR requires a table based implementation, it is shown that the table sizes can be reduced applying compression techniques (Flich et al. 2007).

Routing algorithms designed for regular topologies are generally not usable for irregular topologies or regular topologies with faults. Still, standard algorithms may cope with certain configurations of faults due to the availability of multiple routes. This ability has been shown, for example, by Turn Model algorithms (Glass & Ni 1996).

Higher degrees of fault tolerance require special mechanisms. One example of X-Y routing, enhanced with fault tolerance, is given in (Jie Wu 2003). A routing algorithm explicitly developed for fault tolerance is proposed by (Chen & Chiu 1998).

2.13 Final Comments

This chapter has provided basic information about System on Chip (SoC), networks and Networks on Chip (NoC). It can be observed that most NoC routing proposals are based on work within the area of high performance computer networks. This is logical, since the main purpose of designing systems within the NoC paradigm, is to achieve better efficiency and scalability by utilizing concepts of shared networks.

Two main differences are quickly noted between off-chip and on-chip networks. One is the scarcity of resources due to the single chip limitations. In effect, network resources are directly competing for area and power usage with processing and storage resources. In this aspect, NoC interconnect must be less demanding as compared with off-chip interconnect.

A complicating issue is the various sizes of computing resources that form a complete on-chip system. Therefore, it is relatively more problematic to implement symmetrical topologies in NoCs, since size variations result in wastage of silicon area. This issue is not that central for off-chip networks, since the placement of network resources is less constrained by the environment. On the other hand, non-regular NoCs may have significant drawbacks regarding message routing, system scalability and design effort.

3

Regions in Mesh Networks on Chip

There is a continuous debate regarding the most suitable topology and layout for NoC interconnects. Advocates of irregular structures argue that regular structures waste chip resources and loose performance by not optimizing according to the requirements of a specific system. Quantitative evaluations often support these claims. Opponents dismiss these quantitative arguments and, instead, claim that qualitative advantages, like reduction of design complexity, will provide performance rewards in the end. It is hard to judge which approach is most favorable. One thing seems clear though. Irregular NoC structures are closer to the traditional SoC interconnect techniques. Regular structures are more related to symmetric multiprocessor on-chip interconnects and FPGA architectures.

With respect to regular networks, 2-D mesh is favored by several researchers because of its good NoC design properties. The structure naturally matches 2-D chip layout, while network resources can be kept simple and yet enable high performance. A major drawback with symmetry is the inability to efficiently handle cores of different sizes. The concept of region was proposed as a solution to this problem (Kumar et al. 2002). This chapter elaborates further on the region concept and identifies key implications and possibilities. One important aspect on NoC with regions is the availability of efficient deadlock-free routing algorithms.

3.1 The Region Concept

A symmetrical 2-D mesh NoC is characterized by a layout where the links are of equal length and the core slots are of equal size. This implies that the size of the largest core determines the slot size. If the difference in size is large, a lot of area will be wasted. The region concept (Kumar et al. 2002) was proposed to handle this problem.

Figure 3-1 illustrates a NoC with standard sized tiles and one larger region tile. A region can be described in terms of three parts, namely: core of the region, the wrapper and the access points (see Figure 3-1).

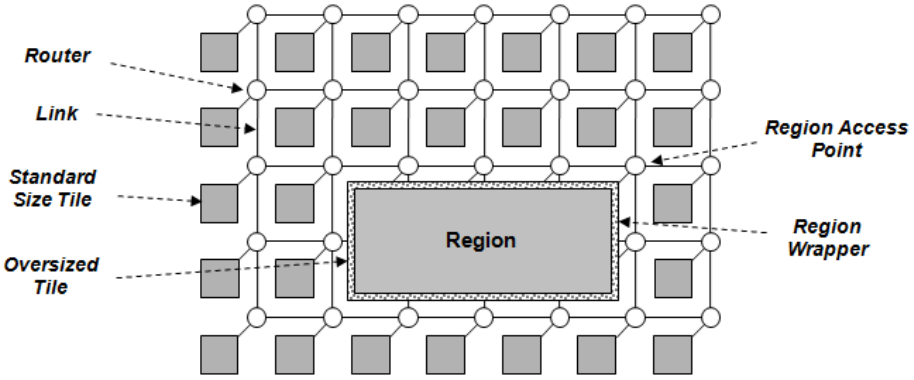


Figure 3-1: NoC with Region

The core of the region is the actual oversized resource. By covering several resource slots, routers cannot be placed within the area of the region. The wrapper is the boundary that provides an interface between the region and the surrounding network. The access points are the network routers that connect to the region via the wrapper and enable communication with the rest of the network resources.

The region concept adds a new degree of freedom with respect to the design of regular 2-D NoC. This increases complexity for system designers but also brings new possibilities. One example of new possibilities is related to access points. As seen in Figure 3-1, the NoC layout enables a region to be connected to several network routers. A natural consequence is that a region can be equipped with multiple access points. However, the use of multiple access points has an adverse impact on the complexity of the routing algorithm. This and other issues are discussed later in this chapter.

This chapter is based mainly on work published in (Holsmark & Kumar 2005) and (Holsmark et al. 2006)

3.1.1 Non-Intrusive and Intrusive Region

A region removes some routers and links that are present in a regular 2-D mesh. As a result, some routing paths that exist in the 2-D mesh are unavailable. This problem can be handled in two different ways. One way, as implied in (Kumar et al. 2002), is to make the

region wrapper emulate routers which would have existed if there was no region. This requires that an additional network has to be implemented in the wrapper around the region core. Full connectivity between region access points implies significant complexity and non-negligible resource requirements of such a network.

Another way, which is assumed in this thesis, is to configure the existing routers and links outside the region to find a communication path around the region. This basically implies the development of new routing algorithms or the adaptation of existing routing algorithms. The intuitive solution is to only modify the paths blocked by the region. However, route changes may affect other communications in the network. One important property that can be altered is deadlock freedom, which is further discussed in Section 3.4.

3.1.2 Logical Regions

Several applications of a region imply that the region structure is physically different in design from its surrounding NoC. This is, however, not necessary; it is possible that the region is defined as a logical structure. A logical region is physically identical with the surrounding NoC but is configured as a separate network. This network is logically isolated with respect to the surrounding NoC communication.

The use of logical regions assumes that there are configurable routers in the NoC that can be used for defining and maintaining a region. These routers on the region boundary isolate the computation and communication within the region from external traffic.

3.2 Accessing Regions

How the region is accessed and how it can access other resources is an important issue while designing with regions. Since it occupies a larger area than a standard resource, it may be useful to consider several addresses and several access points to it. The purpose for which the region is used can affect how the region is designed. A large multi-port shared memory is likely to require several access points distributed around the entire border, whereas a parallel processing system as a region may be accessed only by a few resources outside the region.

A consequence of connecting to several routers is that a region may have several network addresses. Therefore, besides standard interface tasks like protocol conversion and buffering, a region wrapper may have to handle multiple addresses. A large region may also profit from using several internal access points. The interface between the core and the external router is in such cases extended from a one-to-one relationship to: one or multiple internal access-points, being connected to one or multiple external access-point.

3.3 Applications of the Region Concept

In addition to handling cores of larger size, the region concept could be useful for several other objectives. The boundary that the region defines enables effective separation of communications inside a region from communications outside a region.

Special communication requirements can then be fulfilled by enforcing specific rules for traffic within the region. Another region application is the reuse of earlier designed systems. Market demands or other design objectives can hinder designers from converting an existing system to a NoC format. The region concept allows such systems to be more easily reused and incorporated in new NoC systems.

3.3.1 Power and Communication Management

Power consumption is one of the main issues in SoC design. One technique to manage power is to turn-off or reduce activity in certain areas. In this respect, a region allows for local power control of internal routers and resources. If reduction or shut-down of power is required within a region, there would still exist routes around the region border, for the pass-through communication.

Regions can be used for encapsulating a group of resources which have very high or special communication requirements which cannot (are infeasible to) be supported by the general NoC communication infrastructure. Within such a region, there could be specialized interconnections as well as communication protocols for achieving the required objectives.

Quality of service is an example besides the traditional power/performance objectives. Some traffic that requires increased predictability properties could be isolated and may not be interfered by other traffic. Data security is another feature that can be enabled by separation of communication.

3.3.2 Design Reuse

We argue that reuse of multi-core subsystems will become an important application of the region concept in the near future. Usually, large investments are made in the development of electronic systems. It is unlikely that several of these systems will physically fit in the general slot for a core in the mesh NoC.

If a system is to be reused as a sub-system in a NoC, conversion to the NoC format may incur large additional costs, which are only related to the new communication technique. Not only may the redesign effort be too high, the redesigned subsystem may not even be able to achieve the required functionality in the NoC context.

By allowing and encapsulating NoC resources of varying size, the concept of region offers the possibility of raising the level of reuse from a core to a level where specially designed multi-core subsystems can be reused.

As an example of reuse, the region concept can be applied for integration of advanced subsystems, developed for efficient processing of multi-media applications. Several such systems are currently available as separate SoCs.

Figure 3-2 illustrates the possibility of reusing a multi-core multimedia SoC, presented in (Ishiwata et al. 2003), as a NoC region.

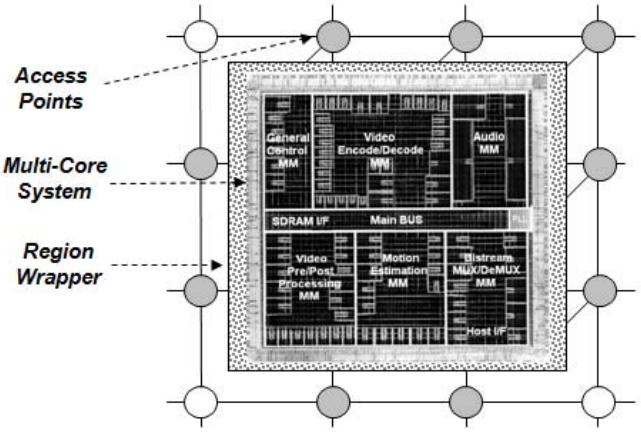


Figure 3-2: Multi-core subsystem (Ishiwata et al. 2003) as a NoC region

3.4 Region Effects on Network Traffic

Incorporating regions in mesh networks results in a change of the communication infrastructure and the existing mesh routing algorithms cannot be directly reused. Several properties are affected by the introduction of regions in mesh NoC. Important among these are:

- Deadlock avoidance
- Route length
- Traffic congestion

Deadlock avoidance

Deadlock avoidance is an important aspect which is affected by allowing regions in NoC. Resource efficient 2-D mesh routing algorithms avoid the occurrence of deadlocks by imposing routing restrictions. If these are followed, the freedom from deadlock is guaranteed. A region blockage may result in that some destinations become un-reachable unless routing restrictions are broken.

One case is illustrated in Figure 3-3, where it is necessary to break X-Y routing restrictions to reach all destination nodes. For example, the prohibited *Y-X* turns,

indicated by dotted arrows, have to be used if source node S1 should reach destination node D1 and if source S2 should reach destination D2.

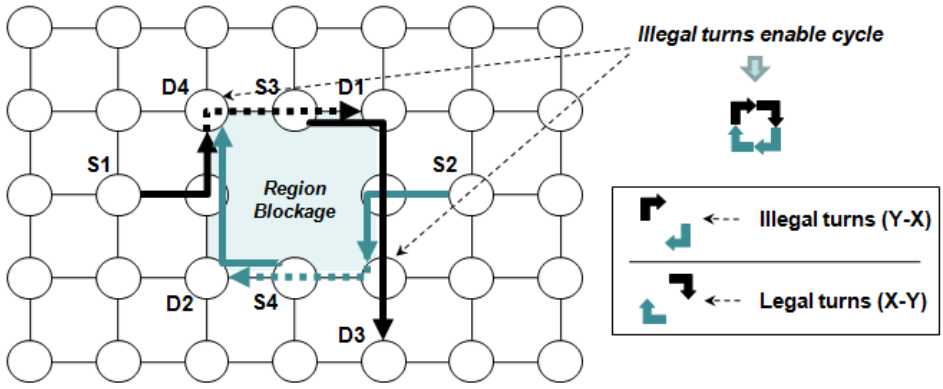


Figure 3-3: Non deadlock-free routing

As shown to the right in the figure, the disallowed *Y-X* turns form a cycle of turns with the allowed X-Y turns. If such cycles exist, routing is not guaranteed to be deadlock free. Therefore, to avoid deadlocks it is necessary to use specialized routing algorithms that safely can route in a NoC with regions.

Route lengths

Compared with a 2-D mesh, route lengths are adversely affected by a region (except for corner placed regions). For some routes this is inevitable, like the route from S2 to D2 in Figure 3-4. In a 2-D mesh this is a straight route but a region blockage disables this minimal path. In other cases, exemplified by the route from S1 to D1, short (minimal) routes may exist but routing rules may prohibit them, permanently or temporary, to be used.

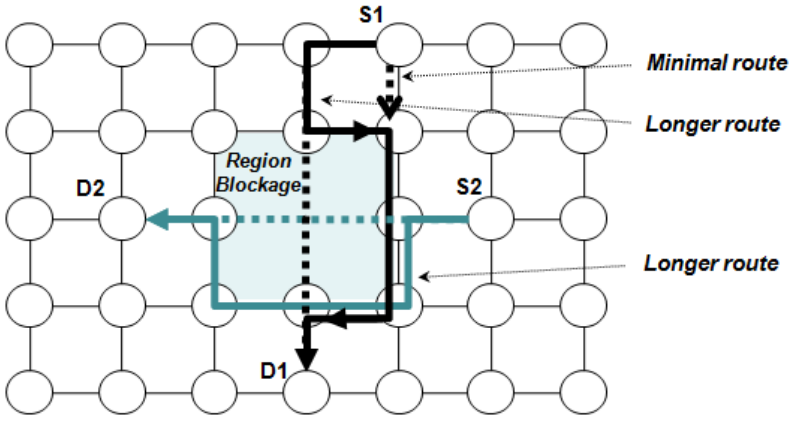


Figure 3-4: Longer routes induced by region blockage

Traffic congestion

In addition to increased route lengths, regions affect traffic distribution in the network. This can raise local congestion levels, especially around borders of regions. An example is given in Figure 3-5, where the traffic flow from the source node S2 to the destination node D2 is obstructed by the region and has to circumvent it. As a result, border links of the region may be more heavily used as compared to other links. Also shown in Figure 3-5, there exist several minimal routes from source S3 to destination D3. Route congestion can then be reduced by shifting the S3, D3 route away from the region border without increasing the distance (and latency) of S3, D3 communication.

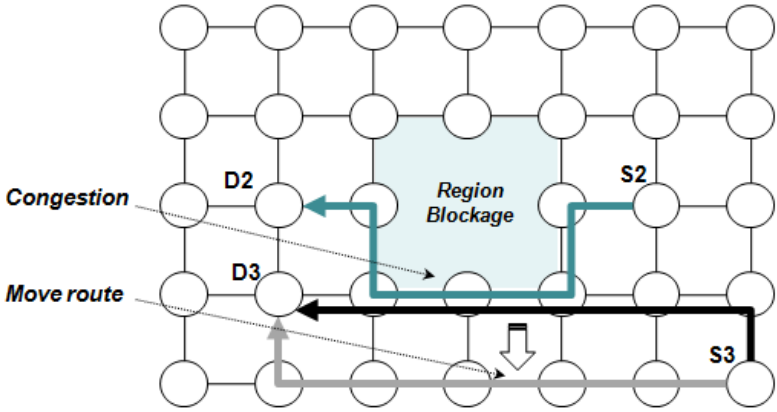


Figure 3-5: Route management to avoid congestion

3.5 Routing in NoC with Regions

A critical aspect of NoC with regions is efficient routing. Considering deadlock prone wormhole switching, system functionality requires that deadlock avoidance is not overlooked. The structural limits, with fundamental traffic impact, are hardly changed by any routing algorithm. Still, there are several options, both for the minimization of adverse traffic effects, as well as for obtaining deadlock-free routing.

3.5.1 Deadlock Freedom

Regarding routing algorithms, there exist a few choices. Generally, achieving minimal routing for all source destination pairs is only possible by the use of virtual channels. This is an alternative, but does not come for free. Another approach is to use routing algorithms for irregular topologies, like Up*/Down* (Schroeder et al. 1991). These are deadlock free on any topology, although, the price to pay is relatively lower efficiency in regular topologies.

Several fault-tolerant routing algorithms for 2-D mesh networks use a fault model (faulty blocks) that resembles a region (Je Wu & Zhen Jiang 2002). These are typically based on algorithmic routing but enhanced by mechanisms to enable circumvention of faulty blocks while still remaining deadlock free. Virtual channels (VC) are heavily used in this area but some non-VC algorithms are also available. Even so, compared to non-fault tolerant versions, they require overhead to manage routing around blocked areas (faults).

3.5.2 Reducing Congestion

Adaptive routing is one solution that can reduce the problem of local congestion around region borders. Normally, the term adaptive refers to a dynamic possibility to detect congestion and take action to divert from it. In this sense it is reactive. When regions are used in a NoC it is possible that static region information is incorporated in the routing algorithm. This way, occurrence of congestion can be minimized or avoided.

For dynamically created logical regions, routers in the network must be aware of their presence in order to effectively route traffic around them. This awareness about position, size, and shape must be efficiently distributed and stored in network routers. Both storing and using the awareness will add to the area and delay of the routers. There is a possibility of partly distributing awareness among routers, leading to solutions with cost/performance trade-off.

3.6 Design Exploration with Regions

The concept of region extends the possibilities for the NoC paradigm based SoCs in interesting and useful ways. New design space exploration activities can be performed in NoC systems with regions. One example is the application mapping, where the use of oversized powerful cores provides new opportunities for exploration.

Another issue is the extended possibilities for core access. Contrary to standard resources, a region can use several network routers as external access points. This section outlines some of the options available for region access. It also discusses trade-offs related to placement of regions and global traffic patterns.

3.6.1 External Access Points

When using a region, the issue of external access-points and addresses to the region must be defined. The number of access points mainly determines the communication bandwidth between a region and the rest of the network, whereas the position of access points more affects the communication latency. Three major options for external access point placement can be identified; single access-point, multiple access-points and extended multiple access-points.

Single access point implies the use of a corner router that originally had a resource connected. This method gives a unique address to the region and has the lowest impact on routing.

Multiple access points extend the single access point scheme to use all the routers on the border, which in a 2-D mesh had connections to resources within the region, as access points. This opens up the possibility to assign the region core multiple distinct addresses to increase bandwidth and access efficiency.

Extended multiple access points allows the use of all possible routers on the boundary as multiple access points to the region. In this case some routers are connected to both a standard resource and a region. This scheme allows the largest number of access possibilities but it complicates addressing, as some routers will need to define multiple destination addresses.

Spare addresses will exist as a result of removed routers but their use requires specific considerations. If access points are to be separately addressed, *only one* may use a specific spare address. Otherwise, as shown in Figure 3-6, a packet for example at node (3, 3) with destination (2, 2) cannot be routed to a unique access point. If access points should be distinguishable, addressing protocol needs to incorporate more information, exemplified by the extended address (3, 2, North) in Figure 3-6, where North is a specific output link.

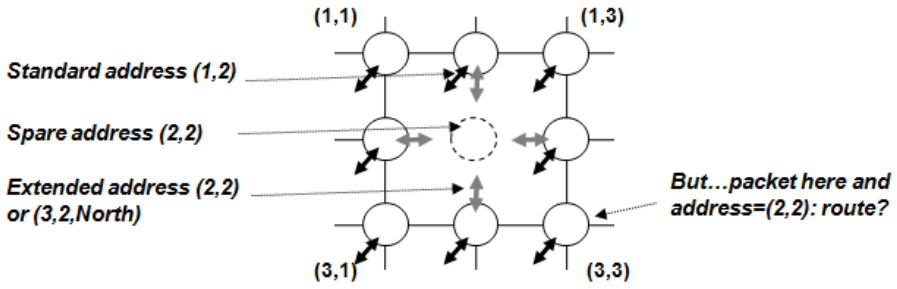


Figure 3-6: Possible access points to a region

3.6.2 Shape and Placement of Regions

The shape of a region has impact on area efficiency and usable routing algorithms. The region concept presented in (S. Kumar et al. 2002) suggested a convex shape of a region. This is easier to handle in terms of routing but may not be optimal from the point of view of placement. Two important aspects to consider in the placement of regions are:

- Access to region - The region should be placed close enough to meet the requirements of the communication to/from it.
- Adverse effect on traffic - The placement of the region should have minimum adverse effect on other traffic in the network.

As an example of a placement trade-off, consider the route-lengths in a network. This parameter is directly affected by the placement of a region. If a region is placed in a corner, it has no negative impact on route-lengths as compared to a regular mesh. On the other hand, there will be a long distance between the region and the cores on the opposite corners and edges. A region placed in a more central position will have less variance in distance to the cores, but will also induce longer routes due to its blockage.

3.7 Discussion

The region concept presented in this chapter results from the SoC characteristic with non-uniform size of available cores. The search for similar ideas in the area of computer networks has not been very fruitful. This is probably due to the virtually un-constrained (compared with on-chip constraints) placement possibilities that are available for configuration of these networks.

Consequently, routing algorithms specifically designed for irregular mesh structures like regions could not be found in literature. Algorithms for irregular networks exist but are generally intended for networks with a higher degree of irregularity. However, similar network structures can be found in research related to fault-tolerant routing (Chen & Chiu 1998) (Jie Wu 2003). Several of these routing algorithms assume faults to be contained in rectangular blocks similar to regions.

Even though both routing algorithms for irregular networks as well as fault-tolerant routing algorithms are usable for NoC with regions, they are generally less efficient due to their more complex implementation.

Being a compromise between regular and irregular structures, NoC with regions loses some of the advantages of regular networks but gain in area efficiency. The disadvantages are more likely to occur in general communication scenarios. Compared with fully application customized networks, NoC with regions may never reach the theoretically optimal efficiency, but will benefit from the design advantages of a structured regular interconnect scheme.

4

Two Routing Algorithms for NoC with Regions

This chapter describes two deadlock-free routing algorithms that can be used for routing in a mesh topology NoC with regions (partially regular NoC). In the following chapters, these algorithms are used and compared in evaluations of routing performance on partially regular NoCs. One of the algorithms is adopted from the area of fault-tolerant routing in general computer networks. The original version of this algorithm contained a few errors which were corrected in the context of this work. The other routing algorithm is directly targeting the NoC area and evolved from considering the application specific nature of embedded systems. The main idea is to relax the assumption that every node communicates with every other node, which has been used in the development of deadlock-free algorithms for off-chip networks. This allows for a higher degree of optimization of an application specific routing algorithm.

The characteristics of the algorithms are fundamentally different in several aspects and often reside on opposite ends with respect to routing algorithm classification. Interestingly, also the methodology and theoretical framework used for the algorithms differ. Still, one property is common: an important objective for both of the algorithms is to guarantee deadlock-free routing with a small amount of resource requirements.

4.1 Overview and Characteristics of the Routing Algorithms

This chapter describes two different algorithms for deadlock free routing in wormhole switched networks. Both routing schemes are applicable for routing in two-dimensional mesh topologies with or without regions. One of the proposals is a corrected version (Holsmark & Kumar 2007) of a fault-tolerant routing algorithm developed by Chen and Chiu (Chen & Chiu 1998). The other proposed scheme is a methodology to design application specific routing algorithms (APSRA) (Palesi, Holsmark & Kumar 2006). The APSRA methodology optimizes routing adaptivity by utilizing knowledge of communication within applications.

The fault-tolerant and APSRA routing schemes diverge in several aspects. Still, some characteristics are shared by both algorithms. The following subsections discuss a few basic aspects and properties of the proposed routing techniques. The contents of this chapter are based on material in (Holsmark & Kumar 2007) and (Palesi, Holsmark & Kumar 2006).

4.1.1 Scope and Basic Properties

The main point in both the fault-tolerant and APSRA routing proposals is to provide deadlock free routing without requiring virtual channels. Deadlock freedom is in both cases based on deadlock avoidance using routing restrictions on link traversals.

The original fault-tolerant algorithm (Chen & Chiu 1998) publication provides a *complete algorithm* for routing messages in a network. Therefore, it is relatively easy to use the algorithm as basis for implementation in network routers. The Turn Model is used for proving deadlock freedom, and it is shown that certain turns can never combine to form a cycle of turns. The purpose of the algorithm is to be able to route in a 2-D mesh even if faults appear during run-time. It is a *general* routing algorithm in the sense that it works for any traffic scenario and fault configuration (region placement) in a NoC. As a result it has good scalability properties and it supports dynamic changes of topology as long as it is derived from a 2-D mesh.

The application specific routing scheme (APSRA) is presented as a methodology to *design routing algorithms*. The methodology provides a theoretical framework and algorithms to support algorithm design for a prospective application. Cycle-free channel dependency graphs (CDG) are used to prove that routing algorithms produced using the APSRA methodology will be deadlock free. The purpose of the methodology is to optimize routing algorithms by using application specific knowledge. Information about task communication is incorporated when designing the routing algorithm such that fewer routing restrictions are necessary in order to guarantee deadlock freedom. As a consequence, *application specific* routing algorithms can have higher adaptivity without sacrificing the important property of deadlock freedom. However, any change in communication pattern or topology requires a re-analysis and possibly re-design of the complete routing algorithm.

4.1.2 Classification of Routing Algorithms

The two routing schemes share a few basic properties. Wormhole switching and no requirements for virtual channels are typical low cost alternatives. Distributed routing is not generally a low budget scheme, even though it usually requires smaller header size than source routing. But, it is more costly with respect to implementation of the routing function in network routers. Deadlock avoidance is normally less resource requiring than, for example, deadlock recovery schemes. Avoidance has a performance drawback though, as it is overly pessimistic and protects from deadlocks that may never occur.

The diverging characteristics of the two algorithms are contrasted in Figure 4-1. The fault-tolerant algorithm is developed for 2-D mesh, but, since it can tolerate faults it works also for 2-D mesh with irregularities. It is deterministic in the sense that, at any given time, there is only a single route for each communication pair. However, if the topology changes, these routes may change. If there are no faults (regular 2-D mesh), routing is minimal. But if the topology is faulty, non-minimal routes may be used. Considering these issues, the characteristics of the fault-tolerant algorithm in Figure 4-1 refers to the situation of a given fixed partially regular mesh topology.

	Fault-tolerant	Application specific
Topology	2-D mesh	Topology agnostic
Route diversity	Deterministic	Adaptive
Route lengths	Non-minimal	Minimal
Implementation	Algorithmic	Table-based

Figure 4-1: Diverging characteristics of fault-tolerant and application specific algorithms

The adaptiveness of the application specific routing algorithm provides a way to adapt routes to avoid traffic congestion. Resource requirements and router delays generally favor algorithmic over table-based implementation. However, issues like routing complexity or table compression may reduce this advantage.

4.2 The Original Fault-Tolerant Routing Algorithm

Chen and Chiu (Chen & Chiu 1998) presented a fault-tolerant algorithm that can be used for routing in the presence of faulty blocks. But, this original algorithm (Chen & Chiu 1998) has two severe problems:

1. In certain situations, the allowed turns combine and form a cycle that can lead to a deadlock. Therefore, the algorithm is *not deadlock free.*

2. There are certain nodes in the network which cannot be reached from a set of source nodes using the algorithm, although there exists possible paths. Therefore the algorithm is also *incomplete*.

This section first presents the basic ideas of the routing algorithm, followed by a description of the identified problems. Examples are given to show that the algorithm can result in deadlocks, and in some cases also fail to route messages between some source-destination pairs, even if paths actually exists. Our new corrected version of the algorithm is described in the next section.

4.2.1 Network Structure and Fault Model

Chen and Chiu (Chen & Chiu 1998) borrow the idea of rings and chains from (Boppana & Chalasani 1995) to isolate the faulty nodes from the rest of the network. Below is a summary of the terminology and important assumptions about nodes, faulty blocks, fault-rings and fault-chains (Chen & Chiu 1998).

Faulty nodes and fault-blocks

Nodes are classified as either non-faulty or faulty. Due to the position of faulty nodes, non-faulty nodes may be *deactivated*. If required for creation of rectangular faulty blocks, a non-faulty node which has at least two neighbors that are faulty or deactivated becomes deactivated. A deactivated node that has at least one active node as neighbor can communicate via their active neighbors but are not used as intermediate nodes for routing of messages. *Active* nodes are non-faulty nodes that do not belong to a faulty block.

Fault-rings and fault-chains

A faulty block is surrounded by either a fault-ring or a fault-chain (see Figure 4-3). A faulty block not touching any boundary of the mesh is surrounded by a *fault-ring*. A faulty block touching a boundary is surrounded by *fault-chain*. A fault-chain touching the north or east boundaries of the mesh uses the rules of a *fault-ring*. A fault-chain touching only the south boundary of the mesh uses the rules of the *s-chain*. Any other fault-chain uses the rules of the *non s-chain*.

Interaction of fault-rings and fault-chains

The definitions of faulty and deactivated nodes imply that there have to be at least two active nodes between two different fault-blocks (if there is only one, this will be deactivated and one faulty block will be formed). Therefore, if any fault-ring/chain has an entire side facing another fault-ring/chain, they cannot share any node. Still, two fault-rings/chains may have one overlapping channel between them. Consequently, there can be at most two nodes that are part of two different fault-rings/chains. Each of these two nodes is then a corner node in each of the fault-rings/chains.

4.2.2 Basic Routing Principles

For messages which do not encounter any fault-ring or fault-chain, routes are non-adaptive with maximum one turn from source to destination. For messages encountering faulty blocks it is necessary to allow some turns that are forbidden during normal routing. Only a few combinations of forbidden turns are allowed in a clever manner such that these turns can never combine with each other (or with allowed normal turns) to form a cycle.

Routing under fault-free conditions

When routing on paths not affected by faults, messages are forwarded in the network according to their type, as illustrated in Figure 4-2. Messages are classified as column-first (CF), row-first (RF) or row-only (RO), depending on the relative locations of the source and the destination of the message.

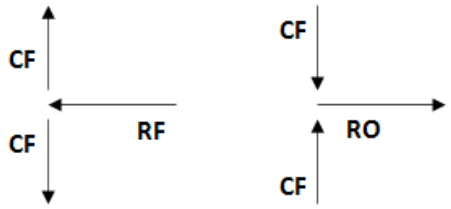


Figure 4-2: Message types and corresponding allowed routes in algorithm (Chen & Chiu 1998)

A message is of type row-first (RF) if its destination is towards west. RF messages must always proceed west first and can only turn north or south at the column of the destination. If the destination of a message is not towards west but towards north or north-east or south or south-east it is a column-first (CF) message. A CF message always travels north or south from its current position before taking a turn to the east. Messages with destination in the same row to the east are labeled as row-only (RO) messages.

Depending on turns, a message may change type. If an RF message makes a west-north or west-south turn it becomes a CF message. A message of type RF can thus change to CF upon reaching the column of destination. A CF message can change to RO when it reaches the destination row and the destination is to the east. An RO message always travels to east, does not take any turns and never changes type.

Routing in fault-affected areas

If a message hits the border of a faulty block, special rules apply depending on the type of the message and whether the border resides on a fault-ring or a fault-chain. There are different rules for routing around faulty areas depending on whether faults are surrounded by an s-chain (chain that touches the south border only), a non s-chain (chain that touches only the west or west and south border) or a fault-ring (all other positions of rings and chains).

Figure 4-3 illustrates the routes for some messages when travelling in the presence of faulty blocks. Note that the bidirectional channels between nodes are omitted for improving readability. The faulty nodes are colored black and cannot be used for routing. A message route is identified by source *Sn* and destination *Dn* node.

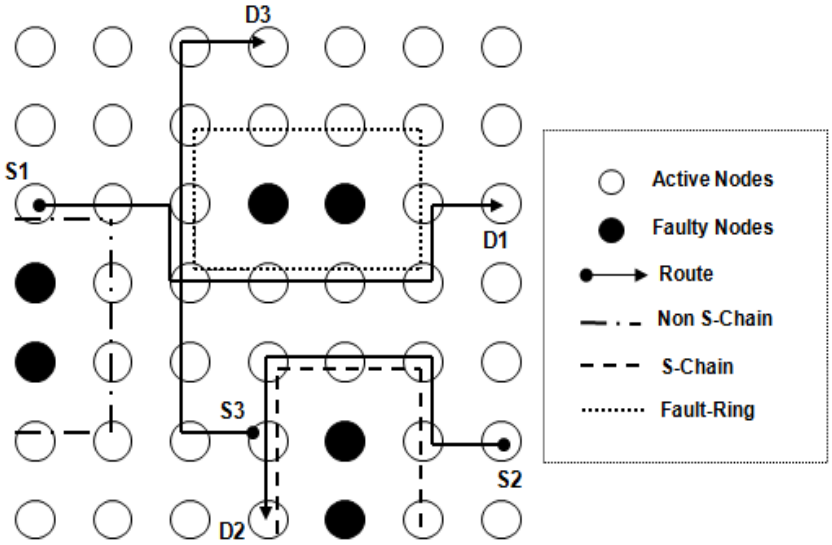


Figure 4-3: Message routes when encountering fault-rings and fault-chains

4.2.3 Possibility of Deadlock

Under certain conditions the algorithm (Chen & Chiu 1998) will enable a packet deadlock. Consider Figure 4-4, where four different messages are travelling in a fault-affected network. The four faulty nodes at the south boundary are surrounded by an s-chain. The three faulty nodes not touching any border are surrounded by a fault-ring.

Message routes are defined according to the original *Message-Route* algorithm (Chen & Chiu 1998). The terms *Normal-Route*, *Ring-Route* and *Chain-Route* are sub-behaviors in the *Message-Route* algorithm for handling messages which are not blocked, blocked by a ring or blocked by a chain respectively.

The message from source node S1 to destination node D1, (S1, D1) is a CF message since its destination is towards south-east. Its path is not blocked by any fault and it can therefore proceed using *Normal-Route*, taking a south-east turn when it reaches the destination row. However, before reaching D1 it is blocked by message (S2, D2).

Message (S2, D2) is an RO message since it has the destination in the same row to its east. Its route is determined by *Chain-Route*, since an s-chain surrounds the faulty nodes that obstruct its normal route. According to *Chain-Route* an RO message should be routed clockwise around the chain. Therefore (S2, D2) turns north at the chain but is then blocked from further advancement by message (S3, D3).

Message (S3, D3) is a CF message with destination in the same column towards north. But, as it resides on an s-chain, *Chain-Route* determines the initial route. *Chain-Route* defines that a CF message heading SN (south to north) at the west boundary of an s-chain should use SN channel if available and if the destination is not to the west of the current node. The destination is not to the west of the current node, thus the available SN channel is selected.

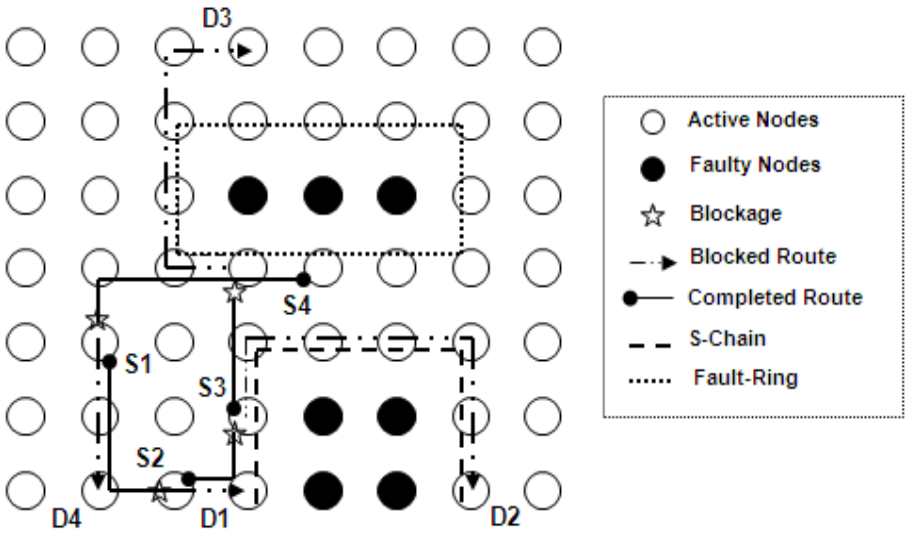


Figure 4-4: Four messages involved in a deadlock situation

Message (S3, D3) then reaches the fault-ring, where *Ring-Route* determines the path. *Ring-Route* returns clockwise route direction since the destination node is not lower than the reference node (top right corner) of the ring. Thus, (S3, D3) should turn west but is blocked by message (S4, D4).

Message (S4, D4) is an RF message with destination at south-west. It is first guided by *Ring-Route* at the fault-ring. According to *Ring-Route* an RF message should use EW (east to west) channel if it is available. One hop outside the ring it makes a west to south turn according to the *Normal-Route* procedure. But it cannot proceed because it is blocked by message (S1, D1).

Therefore, since the four messages are involved in a circular wait that cannot be resolved by *Message-Route* algorithm, a state of deadlock has occurred.

4.2.4 Incompleteness of the Routing Algorithm

The original *Message-Route* algorithm is also unable to route some messages from a set of source nodes to a set of destination nodes. Two cases where messages are stopped from advancing in the network are illustrated in Figure 4-5 and Figure 4-6. As a result, these messages will never reach their destinations.

Case 1

The first case relates to an RF or a CF message heading from north to south (NS) with destination on the west border of an s-chain, below the north-west corner node. At the north-west corner of the s-chain, any message with destination in the same column is CF. Procedure *Chain-Route* determines clockwise route at north-west corner of the s-chain and the only alternative is to turn east. In the next node these messages are stopped since the desired route is to turn back again. Because 180 degree turns are not allowed using *Message-Route*, the messages are blocked. An example of the problem is illustrated in Figure 4-5, where two faulty nodes are surrounded by an s-chain. White nodes are destinations that cannot be reached from any of the nodes marked with grey color.

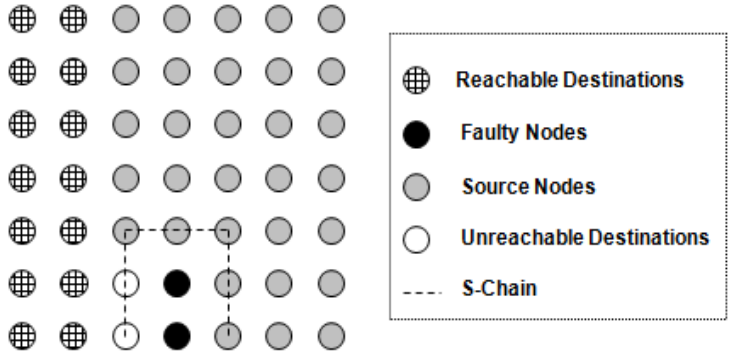


Figure 4-5: Case 1: source nodes with unreachable destinations

Case 2

The second case relates to an RO or CF message proceeding clockwise on any fault-chain with destination towards east of the east border of the chain. In this case, *Chain-Route* decides a clockwise direction for the message.

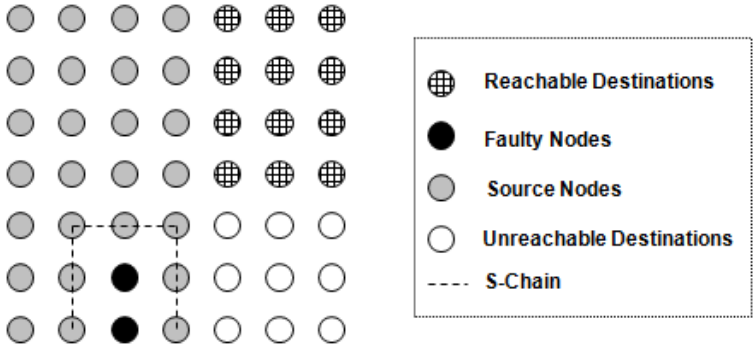


Figure 4-6: Case 2: source nodes with unreachable destinations

The message will propagate around the ring towards the east border, and pass the row of the destination node even when it becomes an RO message. This is because *Chain-Route*

gives no other alternative for an RO message other than to always proceed clockwise. Consequently, the message is blocked in a similar way as in Case 1 and cannot reach the destination. An illustration of Case 2 with affected source nodes and non-reachable destination nodes around an s-chain is given in Figure 4-6.

4.2.5 Analysis of Identified Errors

Deadlock: Error in the proof of Lemma 1 (Chen & Chiu 1998)

As shown in the previous section, deadlock is possible when using the *Message-Route* algorithm. To understand the occurrence of errors, the theoretical proof of deadlock freedom for *Message-Route* was analyzed. The basic technique in the proof is to show that certain combinations of turns which can lead to a deadlock will never occur or will never get aligned to form a circular path.

The proof uses two lemmas before stating the theorem of deadlock freedom. Lemma 1 states that an EN (east to north) turn cannot be aligned with a NW (north to west) turn as long as the EN turn does not occur at the south-east corner of a fault-chain. This lemma is proved by enumerating different conditions where an EN turn can occur. Three different cases are identified where this turn can occur and the algorithm ensures that the dangerous alignment does not take place.

But one case where an EN turn can occur is missed. This is when an RO message makes an EN turn at the west border of an s-chain. The turn follows from *Chain-Route* where an RO message should always be routed clockwise when travelling on a fault-chain. It is possible for such EN turn to align with an NW turn. Therefore, Lemma 1 does not hold. Since Lemma 1 is used in the proof of Theorem 1, neither the theorem holds. Hence, the *Message-Route* algorithm is not deadlock free, at least not in the presence of s-chains in the network.

Blocked messages: Error in the algorithm

The *Message-Route* algorithm is incomplete and may result in that some messages are blocked indefinitely and cannot reach their destinations. A few cases seem to have been missed when defining the algorithm. One missed case occurs for a CF message heading from north to south on an s-chain. There is only one alternative and that is to proceed in clockwise direction.

This alternative works for messages on a non s-chain proceeding from north to south. It can be suspected that the error is a result of a mix-up between the chain types. The other case involves both chain types and the problem is related to an RO message not being able to leave a chain. As this case is handled properly when a ring is involved, the most probable cause is that the correct behavior simply was left out by mistake in the description of *Chain-Route*.

4.3 The Corrected Fault-Tolerant Routing Algorithm

This section presents our new corrected version of the original algorithm (Chen & Chiu 1998). This version, published in (Holsmark & Kumar 2007), can be used for routing in the presence of regions for reaching all destinations in a deadlock-free manner. For our purpose a faulty block described in the original algorithm is equivalent to a region.

Modifications are proposed in *Chain-Route* to solve the problems related to routing around chains. The solution to avoid deadlock while routing around an s-chain is based on the method to route CF messages which are going from north to south at the west border of a fault-ring. All messages, except those blocked by the s-chain, are prohibited from travelling from south to north on the west border of an s-chain. Other messages are forced to first take a west hop before proceeding northwards. This will break the link between the EN turn at the s-chain and the NW turn at a fault-ring or chain positioned above the s-chain. The necessary corrections to avoid blockage of messages and make the routing algorithm complete are also incorporated.

4.3.1 The Corrected Message-Route Algorithm

Although corrections apply only to procedure Chain-Route, the full algorithm is given for the sake of completeness. As the original algorithm is somewhat unclear in the treatment of overlapping fault-rings/chains, a procedure, *Overlapped-Ring-Chain-Route*, is added to resolve this situation. This procedure lists rules of handling this situation as described in (Chen & Chiu 1998). The corrections are typed in bold font. We use the terminology of the original algorithm.

```
Procedure Message-Route-Modified
/* Message mg is sent from source S to destination D. C is the current node of the header flit. */
if (C is the destination D){
    Consume mg;
else
    if (current node C is S, and is unsafe)
        Send mg to an active neighbor;
    else/* active node */
        begin
        Determine the message type (RF, CF, or RO) of mg;
        if (C is not on a fault ring or fault chain)
            Normal-Route(mg);
        else
            if (C is on a single fault ring)
                Ring-Route(mg);
            else
                if (C is on a single fault chain)
                    Chain-Route-Modified (mg);
                else
                    Overlapped-Ring-Chain-Route (mg);
end;
```

```
Procedure Overlapped-Ring-Chain-Route (mg)
if (mg is RO message)
    Select Ring-Route(mg) or Chain-Route(mg) as used in previous node;
else
    switch (mg's direction)
        case (EW message)
            Compare the reference point of the overlapped chains/rings
            Select Ring-Route (mg) (Chain-Route (mg)) if reference point of ring (chain) was more
            westwards;
        exit;

        case (WE message)
            Compare the reference point of the overlapped chains/rings
            Select Ring-Route(mg) (Chain-Route(mg)) if reference point of ring (chain) was more
            eastwards;
        exit;

        case (SN message)
            Compare the reference point of the overlapped chains/rings
            Select Ring-Route(mg) (Chain-Route(mg)) if reference point of ring (chain) was more
            northwards;
        exit;

        case (NS message)
            Compare the reference point of the overlapped chains/rings
            Select Ring-Route(mg) (Chain-Route(mg)) if reference point of ring (chain) was more
            southwards;
        exit;
end;
```

```
Procedure Normal-Route(mg)
switch (mg's type)
    case (RF message)
        Use EW channel to forward mg;
    exit;

    case (CF message)
        if (mg is NS message)
            Use NS channel to forward mg;
        else
            Use SN channel to forward mg;
    exit;

    case (RO message)
        Use WE channel to forward mg;
    exit;
end;
```

```
Procedure Ring-Route(mg)
switch (mg's type)
    case (RF message)
        if (EW channel is available)
            Use EW channel to forward mg;
        else
            Route mg clockwise;
    exit;
    case (CF message)
        if (mg is SN message)
            if (C is on the north boundary of the fault ring)
                Normal-Route(mg);
            else
                if (C is on the west boundary of the fault ring and D is in the same column as C)
                    Normal-Route(mg);
                else
                    if (D is lower than the reference node of the ring)
                        Route mg counter-clockwise;
                    else
                        Route mg clockwise;
        else /* NS message*/
            if (C is on the east or south boundary of the fault ring)
                Normal-Route(mg);
            else
                if (C (including S) is on the west boundary of the ring and EW channel is available)
                    Route mg along EW channel;
                else
                    Route mg counter-clockwise;
    exit;
    case (RO message)
        if (C is in the same row as D and WE channel is available)
            Use WE channel to route mg;
        else
            Route mg counter-clockwise;
    exit;
end;
```

```
Procedure Chain-Route-Modified (mg)
    Switch (mg's type)
        case (RF message)
            if (the fault chain is an s-chain)
                if (EW channel is available)
                    Route mg along EW channel;
                else
                    Route mg counter-clockwise;
            else /* not s-chain */
                if (C is in the same row as D)
                    Route mg along EW channel;
                else
                    if (D is higher than C)
                        Route mg counter-clockwise;
                    else
                        Route mg along clockwise direction;
        exit;
        case (CF message)
            if (mg is an NS message)
                if (the chain is an s-chain)
/*Correction 1: Reach destinations on west border of s-chain*/
                    if (C (including S) and D is on the west border of the chain)
                        Route mg along NS channel;
                    else
                        Route mg clockwise;
                else /* not s-chain */
                    if (NS channel is available and D is not to the west of C)
                        Route mg along NS channel;
                    else
                        Route mg clockwise;
            else /* SN message */
/*Correction 2: Avoid deadlock when RO makes EN turn at s-chain*/
                If (the chain is an s-chain)
                    if (C is on the north or east boundary of the chain)
                        Normal-Route(mg);
                    else
                        if (C (including S) is on the west boundary of the ring and EW channel is
                        available)
                            Route mg along EW channel;
                else /* not s-chain */
                    if (SN channel is available and D is not to the west of C)
                        Route mg along SN channel;
                    else
                        Route mg counter-clockwise;
        exit;
        case (RO message)
/*Correction 3: Reach destinations located east of chain*/
            if (C is in the same row as D and WE channel is available)
                Use WE channel to route mg;
            else
                Route mg clockwise;
        exit;
end;
```

4.3.2 Discussion on Changes to the Algorithm

There are three corrections in the procedure *Chain-Route*. The first correction enables CF messages travelling from north to south to reach destinations south of the north-west corner of the s-chain. If the current node or the source node is on the west border and the destination also is on this border, the message should be routed via the north to south channel. This will not violate the proof since no message makes an SW turn on or below the east border of an s-chain.

The second correction is made to avoid the occurrence of a deadlock when making an EN turn at the west border of an s-chain. It makes a CF message going from south to north on the west border of an s-chain divert one hop west before proceeding north. This will stop these messages to use the link immediately below the north-west corner of the s-chain in order to prevent this link to be involved in a deadlock.

The third correction makes it possible for messages to reach destinations located to the east of a chain. As previously described, these were unreachable from source nodes at certain locations. These messages can now leave a chain using a WE channel when the destination row is reached.

The new procedure, *Overlapped-Ring-Chain-Route* is added to clarify the routing decision when a message is on a node which is part of two rings or a ring and a chain or two chains.

4.3.3 Proof of Deadlock Freedom

The cause of failure of the original algorithm was found in the proof of Lemma 1 (Chen & Chiu 1998), which resulted in a non-valid proof of the main theorem. Both lemmas and the theorem are restated here for the modified algorithm. Note that Lemma 1 is more general with respect to an EN turn as compared to the corresponding lemma in (Chen & Chiu 1998). The new proofs are written in the same style as in the original algorithm. Because of the modifications to the algorithm the errors in the original algorithm (Chen & Chiu 1998) are removed.

Lemma 1 Using the *Message-Route-Modified* algorithm, an EN turn cannot be aligned with an NW turn as long as the column segment between the turning nodes of the EN and NW turns does not include the east boundary of a fault-chain.

Proof: According to *Message-Route-Modified* algorithm, an EN turn can only be made in four different cases:

1. An RO message encounters a fault-ring, and makes an EN turn at the south-east corner node of the ring.

2. An SN message encounters a fault-ring, and the destination of the message is lower than the reference node of the ring.

3. A message makes an EN turn at the south-east corner node of a fault-chain.

4. An EN turn is made by an RO message on encountering an s-chain.

The proof for the first three cases for the original *Message-Route* algorithm given in (Chen & Chiu 1998) still holds for the modifications. Therefore, only the fourth case is proved here.

There are only two cases in which an NW turn can be made using *Message-Route-Modified* algorithm. In the first case a message is routed around a fault-chain in counter-clock wise direction and makes an NW turn at the north-east corner of a fault-chain. The second case is when a CF message is blocked by a ring.

According to Lemma 1, it is only necessary to prove that an EN turn made by a message blocked by an s-chain does not align with a CF message blocked by a ring. It will be proved by contradiction that this situation is not possible.

Assume now that an EN turn is aligned with an NW turn. Let T_1 represent the EN turn and T_2 represent the NW turn. For the turns to be aligned there must now exist a set of messages, m_1, m_2, ..., m_k such that m_1, which is an RO message, takes T_1, m_k takes T_2, m_i waits for m_{i+1} for all $1 \leq i \leq k$-1, and the column segment that starts from T_1 and continues to T_2 is a sub-path of the waiting path of the messages. According to *Chain-Route-Modified* procedure the turning node of T_1 cannot be the north-west corner of the s-chain in this case. The following shows that T_1 and T_2 cannot be aligned.

An NW turn can be made by m_k if m_k is an SN message and encounters a fault-ring. There are two different sub-cases where T_2 can be positioned.

Case 1: The turning node of T_2 is no higher than the north-west corner of the s-chain. In this case the column channel of T_2 is located on the west boundary of the s-chain. Our algorithm requires that an SN message, which starts from a source node, located below the north-west corner node, on the west boundary of the s-chain, is routed one step to the west as long as such channel is available. Recall that two rings/chains can overlap at most one channel. Thus, such a channel is available at and below the source node of the column channel of T_2. Therefore, m_k cannot possibly travel through the SN channel of T_2. Hence, T_2 cannot be made by m_k.

Case 2: The turning node of T_2 is located higher than the north-west corner of the s-chain. Consequently, there must exist some message, m_i, travelling north through the NS channel immediately below and above the north-west corner of the s-chain. Note that m_1 must make NE turn at this node, thus m_1 cannot be m_i. Actually, any RO must proceed east here, hence m_i cannot be RO. Neither can m_i be an RF message as these must be routed east to west in the case considered. Next, assume that m_i is a CF message. In this case m_i must be an SN message but our algorithm requires that an SN message, which starts from a source node, below the north-west corner node, on the west boundary of the s-chain, is routed one step to the west as long as such channel is available. In the case considered such a channel is available below the north-west corner node of the s-chain. Therefore m_i cannot possibly travel through the SN channel immediately below the south-east corner of the s-chain. Hence, the existence of such an m_i is not possible.

□

Since the algorithm for the cases covered by the original Lemma 2 in (Chen & Chiu 1998) have not been changed in *Message-Route-Modified* and its proof does have any errors we can leave this lemma unchanged. But we restate this lemma for the modified algorithm.

Lemma 2 Using the *Message-Route-Modified* algorithm, an ES turn cannot be aligned with an SW turn as long as the column segment between the turning nodes of the ES and the SW turn does not include the east boundary of any fault-chain.

As Theorem 1 and the original proof in (Chen & Chiu 1998), see Appendix II, are based on the validity of these two lemmas we also use them to prove deadlock freedom for the *Message-Route-Modified* algorithm. The proof for the modified algorithm will be identical to the proof of corresponding theorem in (Chen & Chiu 1998).

Theorem 1 *Message-Route-Modified* algorithm is deadlock free in a 2D mesh.

The theorem can easily be proved using Lemma 1 and Lemma 2 and following the same steps as were followed for the proof of the original *Message-Route* algorithm.

4.4 Application Specific Routing

Most routing algorithm proposals for NoC systems are directly adopted from routing algorithms intended for off-chip high performance computer networks. One important feature of these algorithms is the ability to support a general communication scenario among the connected computers. For example, using the X-Y routing algorithm in a regular 2-D mesh makes it possible to guarantee that every node can communicate with any other node in the network, while guaranteeing freedom from deadlocks. However, the X-Y algorithm does not allow adaptive message routes and is overly restrictive from a deadlock point of view.

It is likely that several NoC systems will exist in the embedded systems domain. An important aspect of embedded systems, in contrast to general computer networks, is their application specific characteristics. These characteristics can be used for the design of a routing algorithm. In (Palesi, Holsmark & Kumar 2006) we proposed a methodology called Application Specific Routing Algorithms (APSRA), which utilizes the knowledge of actual network traffic.

4.4.1 Background and Basic Idea

The worst-case traffic assumption limits the possibilities of adaptive routes. For example, allowing full adaptivity, at least in the absence of virtual channels, is not possible even in a regular network like the 2-D mesh. Several partially adaptive routing algorithms, (e.g. West-First, Odd-Even) have been proposed to increase adaptivity as compared to static algorithms like X-Y. These algorithms try to achieve as much (or evenly distributed)

adaptivity as possible and still be deadlock free. Note that if the topology changes due to designer intervention, as in the case of regions, or faulty routers it is no longer possible to use any algorithm that assumes a regular topology.

When designing a routing algorithm it is possible to use the concept of a CDG (channel dependency graph) to prove the property of deadlock freedom. Deadlock-free message routing is guaranteed if there are no cycles in the CDG obtained from the routing algorithm. But, also the CDG method assumes a worst case communication pattern. In order to obtain a cycle-free CDG it is often necessary to remove some adaptivity regarding the packet routes. If it is the case that some nodes actually never communicate, the CDG is too restricted with respect to adaptivity, since its creation was based on the worst case scenario.

In an embedded system, on the other hand, it is possible that a designer has knowledge about which node pairs in a network communicates with each other. This information can be used to create an *application specific channel dependency graph* (ASCDG). The ASCDG is based on the same information as the CDG, but also considers information on the communicating nodes. The result of the added information is that the ASCDG is likely to have fewer cycles than the CDG. In effect, the use of an ASCDG makes it possible to allow more adaptivity as compared to a CDG, while still securing freedom from deadlocks.

4.4.2 Definitions and Proof of Deadlock Freedom

In order to correctly perform message routing in a network, APSRA generated routing tables should guarantee that communication in the network can be performed without possibilities of deadlock. This subsection describes the theorethical foundation upon which this property is built.

Definition 1 A *network* is modeled by a graph $T=(P,L)$ where each vertex $p \in P$ is a network (router-) node and each directed arc $l_{ij} \in L$, represents a unidirectional link connecting node p_i with node p_j. Let $out(p)$ / $in(p)$ represent the set of output links / input links of a node p respectively, and let $src(l)$ / $dst(l)$ represent the source / destination nodes of a link l respectively.

Definition 2 A *routing function* for a node $p \in P$ is a function $R(p): L_{in}(p) \times P \rightarrow \wp(L_{out}(p))$. $R(p)(l,q)$ gives the set of output channels of node p that can be used to send a message received from the input channel l and whose destination is $q \in P$.

Definition 3 Given a communication graph $CG(T,C)$, a topology graph $TG(P,L)$, a mapping function $M: T \rightarrow P$ and a routing function R, there is an *application specific direct dependency* from $l_i \in L$ *to* $l_j \in L$ iff

$$dst(l_i) = src(l_j), \qquad (1)$$

and

$$\exists c \in C: l_j \in R(dst(l_i))(l_i, M(dst(c))) \qquad (2)$$

Condition (1) states that there exists a possibility for a message to use l_j immediately after l_i. Condition (2) states that there exists a communication that can actually use l_j immediately after l_i.

Definition 4 An *application specific channel dependency graph* for a given communication graph *CG*, a topology graph *TG*, and a routing function *R*, is a directed graph *ASCDG(L,D)*. The vertices of *ASCDG* are the channels of *TG*. The arcs of *ASCDG* are the pair of channels (l_i,l_j) such that there is an application specific direct dependency from l_i to l_j.

The routing tables produced by APSRA implement the routing function. In the current APSRA version, this is initially assumed to be a minimal fully adaptive routing algorithm. This means that all shortest path routes that exist between sources and destinations, as a result of mapped tasks, are available for message transportation. By applying the routing function (Definition 2) it is possible to find the application specific direct dependencies (Definition 3). The ASCDG (Definition 4) is built by connecting links using application specific direct dependencies.

The final routing function is restricted if cycles need to be removed from the ASCDG. Since the routing tables implement restrictions on communication according to the ASCDG, we need to prove that an acyclic ASCDG results in deadlock-free communication. In (Palesi, Holsmark & Kumar 2006) we proved the following theorem.

Theorem 1 A routing function *R* for a topology graph *TG* and for a communication graph *CG* is deadlock free if there is no cycle in its application specific channel dependency graph *ASCDG*.

Since an acyclic ASCDG has the same significance as an acyclic CDG, the proof is almost identical to the proof for deadlock freedom in (Duato 1993). It is based on the possibility of numbering channels in decreasing order, where a minimal of such channel order is a sink.

Proof: As the *ASCDG* is acyclic, it is possible to establish an order between the channels of *L*. Suppose that there is a deadlocked configuration for *R*. Let l_i be a channel of *L* with a nonempty queue such that there are no channels less than l_i with a nonempty queue. The following two cases are possible:

Case 1: l_i is a minimal. In this case, as proved in (Duato 1993), l_i is a sink, i.e. a channel such that all the flits that enter on it reach the destination in a single hop. Then, the flit at the queue head can reach its destination in a single hop and there is no deadlock.

Case 2: l_i is not a minimal. For each $l_j \in L$ such that $l_i > l_j$, there are no flits in the queue for channel l_j. Thus, the flit at the head of the queue for l_i is not blocked, regardless of whether it is a header or a data flit, and therefore, there is no deadlock.

□

4.5 APSRA Methodology for Design of Routing Algorithms

The objective of the APSRA methodology is to design deadlock-free routing algorithms that consider only the actual communication among nodes in a network. By doing this, less routing restrictions need to be applied as compared to the routing algorithms designed by assuming worst case traffic.

4.5.1 Overview of the APSRA Design Flow

Figure 4-7 shows an overview of the design flow for designing routing algorithms using the APSRA methodology. The inputs of the design process are information about communication among tasks and the network topology. The tasks are then mapped (Map) to resources in the network. The mapping result, i.e. information about which network resources the tasks are assigned to, is fed into the APSRA algorithm.

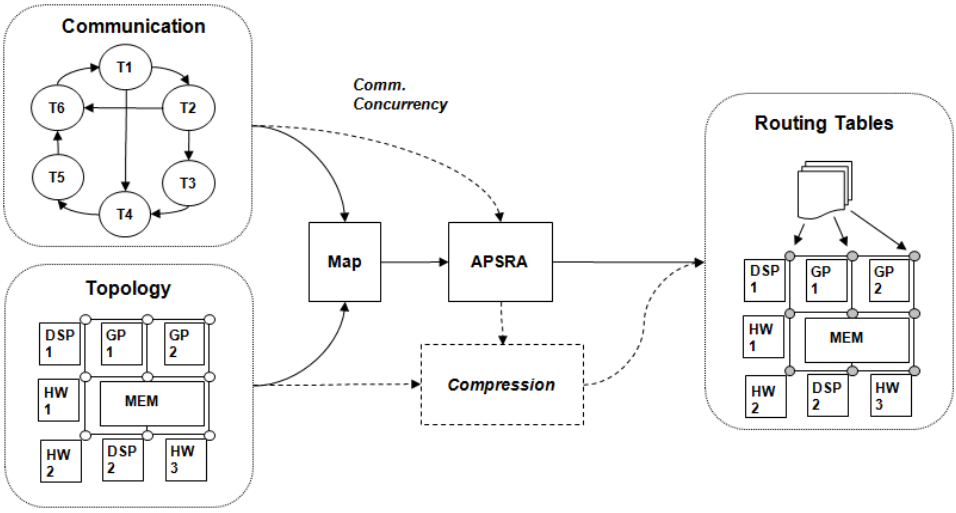


Figure 4-7: Overview of APSRA design methodology

The APSRA algorithm produces routing tables that can be programmed into the network routers. Optionally, information about communication concurrency can be used in APSRA to further optimize the produced routing algorithm (Palesi et al. 2007). Generally, tables generated by APSRA require one entry per destination node in each router (or each input if multiple tables are used). However, it is possible to reduce the size of routing tables by using an optional compression technique described in (Palesi et al. 2006).

The internal tasks that the APSRA function performs are depicted in Figure 4-8. The mapping result is first used to create the ASCDG. The initial ASCDG is currently built by assuming a minimal fully adaptive routing function. If the ASCDG does not contain any cycles, routing tables can immediately be produced. In this case all actual communications are allowed minimal fully adaptive routing.

In the case that cycles are found in the ASCDG, these have to be removed. This means that one or more routes have to be restricted in terms of adaptivity. The current method to select which routes should be removed is guided by an objective to maximize average adaptivity of all communications in the network. After a cycle-free ASCDG is created, the routing tables can be produced. The produced tables disallow some minimal routes which could have been available if the routing function was fully adaptive.

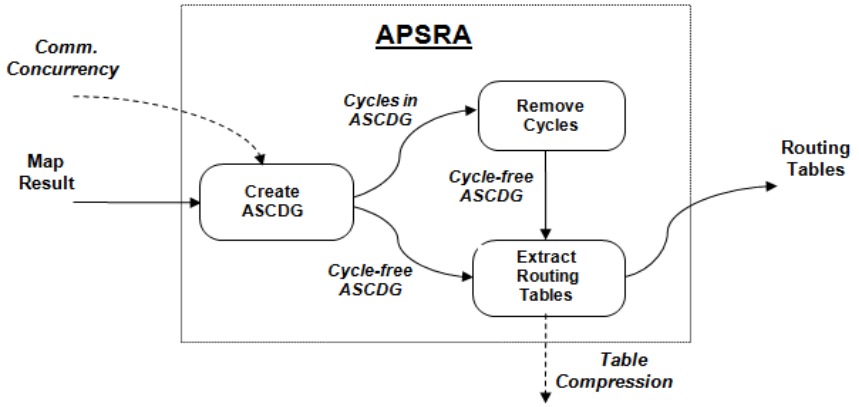


Figure 4-8: Overview of APSRA tool functionality

4.5.2 The APSRA Algorithm

The main algorithm for implementing APSRA is shown in Figure 4-9. The input of the APSRA algorithm is a communication graph CG, a topology graph TG and a mapping function M. The output is a set of routing tables RT. First, the routing algorithm R is assigned a minimal fully adaptive routing algorithm based on the current CG, TG and M. Then the initial ASCDG is created in procedure *BuildASCDG*. *GetCycles* extracts cycles in the ASCDG. If there are cycles, these have to be removed. This is the task of procedure *RemoveCycles*. If cycles are successfully removed, routing tables are extracted. If not successful, the empty set is returned.

The execution time of the APSRA algorithm depends mainly on network size and the number of communications. The *BuildASCDG* procedure needs to annotate all minimal paths for each source destination pair. For an *NXN* mesh and worst-case traffic, the complexity of this task is $O(2^n)$. Therefore, the execution time of *BuildASCDG* may be infeasible for large networks if the distances of several communications are large. In real systems though, such extreme conditions are unlikely. Usually, mapping of tasks results in placement of the most frequently communicating peers close to each other (Ogras &

Marculescu 2006). If the problem still exists, a practical solution could be to consider only a subset of the minimal paths for long distance communications.

```
APSRA(in : CG, TG, M; out : RT){
R← MinimalFullyAdaptiveRouting(CG, TG, M);
BuildASCDG(CG, TG, M, R, ASCDG);
GetCycles(ASCDG; C);
RemoveCycles(C, ASCDG, CG, TG, M, R, success);
if( success )
ExtractRoutingTables(R, RT);
else
RT ← ∅;
}
```

Figure 4-9: APSRA implementation algorithm

The procedure *RemoveCycles* tries to remove cycles formed in the ASCDG. The task of removing cycles in the ASCDG involves decisions on which cycle should be removed first and which of the dependencies that forms a cycle should be broken. The strategy for selecting which cycles to remove is determined by the designer. Random selection can, for example, be used if no specific objective exists. In the basic APSRA, *RemoveCycles* cuts cycles such that the loss of adaptivity is minimized. An important constraint during cycle removal is that there exists at least one path for each source destination pair.

As cycles may be inter-dependent, the order in which cycles are cut may affect the quality of the solution. In the basic implementation, order is not considered. Instead, backtracking and depth-first search of removing sequences are used. As shown in the adaptivity analysis in (Palesi, Holsmark & Kumar 2006), this method produces fast solutions under the communication scenarios and mesh sizes considered. But the time complexity of this procedure grows exponentially with the size of the network.

4.5.3 APSRA through an Example

An example of the functionality and result of APSRA is illustrated in Figure 4-10. The APSRA design flow is enclosed by dotted lines. The figure shows each of the basic design components related to APSRA:

Inputs: Topology and Communication graphs

Outputs: CDG, Initial ASCDG and Final ASCDG

The basic channel dependency graph (CDG) is obtained from the topology, under the assumption of worst-case traffic. As can be seen, the CDG contains two cycles which have to be removed to secure freedom from deadlocks.

In the APSRA flow, information about the actual communication among the tasks is captured by a communication graph. This knowledge is combined with the topology to create the initial application specific channel dependency graph (ASCDG).

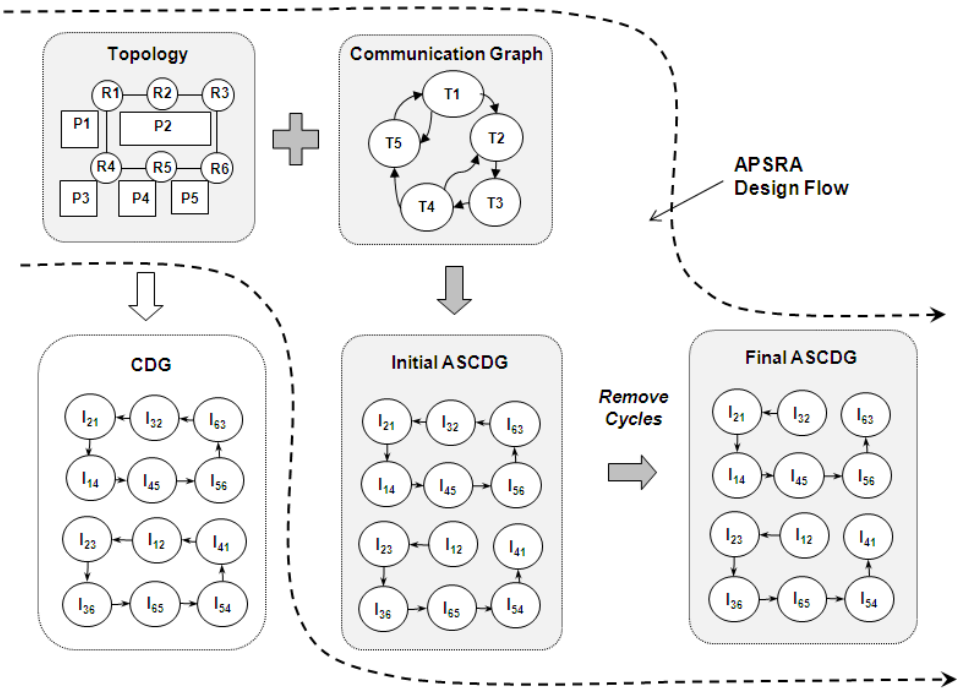


Figure 4-10: Example: topology, communication and ASCDG

Assume that the five tasks are mapped to a resource with the same number, i.e. *M(T(i)) = P(i)*. Let resource P2 be connected to the upper rightmost router R3. The links (vertices of the CDG) are numbered such that a link l_{ij} is an output link of node *Ri* and an input link of node *Rj*.

One of the two cycles in the CDG is not present in the initial ASCDG as a result of absence of actual communication. The arc between l_{41} and l_{12} does not exist since there is no communication from resource P3 (or R4) to resource P2 (or R3).

Compared to the basic CDG this means that only one cycle has to be removed to guarantee freedom from deadlocks. As a result, the actual communication can be performed with higher degree of adaptivity as compared to cutting two cycles.

Let us now look at the remaining cycle. The arc from l_{63} to l_{32} is a result of the communication T5 to T1. This communication can also be performed by the links l_{65} to l_{54} to l_{41}. Therefore, we can cut l_{63} to l_{32} and still be able to route the desired communications. The price to pay in this case is the loss of adaptivity for communication T5 to T1, which is now constrained to use only one path.

4.5.4 Cycle Removal in ASCDG: Objective Function

Although the ASCDG in most cases contains fewer cycles than the corresponding CDG, the remaining cycles have to be removed in order to obtain a deadlock-free routing solution. There are several options to guide the selection of cycles to remove, for example:

- Maximization of adaptivity
- Static route planning
- Load balancing

In the basic version of APSRA, the objective is set for maximizing adaptivity of messages travelling in a network. If a non-adaptive routing algorithm is preferred, APSRA can be used to select fixed paths for messages, with the objective to achieve a balanced traffic distribution. If the required communication volumes are known, it could be used to further refine the path selection in design of adaptive routing algorithms (see for example (Palesi et al. 2008).

Definition of adaptivity

Adaptivity is a term that indicates the number of paths available for communication. A static routing algorithm provides only a single path for each source-destination pair and it is therefore non-adaptive. Adaptivity $\alpha(c)$ for a communication c is defined as

$$\alpha(c) = \frac{\Phi(c)}{TMP(c)}$$

where $\Phi(c)$ represents the number of permissible paths for c and $TMP(c)$ represents the total number of minimal paths. The average adaptivity α over all communications is calculated by

$$\alpha = \frac{1}{C} \sum_{\forall c \in C} \frac{\Phi(c)}{TMP(c)}$$

Cutting cycles by minimizing loss of adaptivity

The technique we proposed in (Palesi, Holsmark & Kumar 2006) cuts cycles by trying to minimize the loss of adaptivity. The initial ASCDG is obtained by applying a fully adaptive minimal routing algorithm for all communication among tasks in a system. Every edge in this ASCDG is a channel dependency, caused by message routes traversing two consecutive channels.

When a dependency is removed there will be a reduced number of message paths for the affected communications. In effect, the removal of some dependency in a cycle impacts on both the individual adaptivity $\alpha(c)$ for an affected source destination pair, as well as on the average adaptivity α for all communications.

Let $\Phi_d(c)$ be the paths removed for communication c by cutting dependency d. Then the resulting adaptivity for c is reduced to $\alpha_d(c)$

$$\alpha_d(c) = \frac{\Phi(c)}{TMP(c)} - \frac{\Phi_d(c)}{TMP(c)}$$

Given all communications affected by d the reduced adaptivity is α_d

$$\alpha_d = \frac{1}{C}\sum_{\forall c \in C}\frac{\Phi(c)}{TMP(c)} - \frac{1}{C}\sum_{\forall c \in C}\frac{\Phi_d(c)}{TMP(c)}$$

In order to maximize adaptivity, the difference α-α_d should be minimized. Then the edge to be removed is the one that minimizes the quantity

$$\min_{d \in D_c}\frac{1}{C}\sum_{\forall c \in C}\frac{\Phi_d(c)}{TMP(c)}$$

In other words, the dependency that minimizes the lost adaptivity should be selected by considering that each dependency is weighed by the adaptivity of the communications which use it. An important restriction during the procedure of removing cycles is that there should still remain at least one path between each source-destination pair. Note that some communications may actually only have a single path.

The current version to cut dependencies considers one arbitrary cycle at a time and produces a locally optimal solution i.e. minimal loss of adaptivity, with respect to this single cycle. However, it is not necessary that the overall adaptivity is maximized by trying to maximize the adaptivity considering a single cycle. This is because the removal of a dependency can affect more than a single cycle.

Also the constraint that all destinations must be reachable is affected by the order that dependencies are cut. Consequently, to produce a globally optimal routable solution, one need to consider all cycle-dependencies and all the possible orders these can be cut. Therefore, the worst case time complexity of this task is exponential.

4.6 Extensions to APSRA

This section briefly describes ways to extend the basic APSRA methodology. Some improvements to the original approach have already been published. These are mainly regarding optimizations to improve communication adaptivity and resource requirements. Ideas on adapting APSRA to another concept for deadlock-free communication are also discussed.

4.6.1 Communication Concurrency

As mentioned in previous sections, the APSRA methodology can be further optimized by considering the concurrency of task communication in a system. The basic idea, elaborated in (Palesi et al. 2007), is that only communications which are concurrently

active can form application specific dependencies. The communication graph is divided into communication scenarios with communications that are concurrently active. Each of these scenarios contains fewer communications as compared to the initial communication graph.

Every communication scenario results in a separate ASCDG that, as a result of fewer communications, is less likely to contain cycles. Therefore, higher degrees of adaptivity can be achieved, since fewer cycles means less restrictions of adaptivity. It should be noted that this optimization comes with some important implementation issues. There may be considerably large overhead to synchronize communication scenarios among routers in the network. For example, one key requirement is that the transition time from one communication scenario to the next needs to be long enough, such that all initiated communications related to the first scenario can reach their destinations.

4.6.2 Table Compression

One disadvantage of the basic APSRA methodology is the requirement of a table-based implementation, where the table size is dependent on the number of nodes (destinations) in the network. This is because each communication through a router must have its destination in an entry of the router table. However, the size of router tables can be compressed by exploiting information about network topology. For example, in (Palesi et al. 2006) we showed that a table size of two entries for each output port can in some cases provide performance results within 3% of the performance using an uncompressed table.

4.6.3 A Potential Methodological Change to APSRA

APSRA can be adapted to other frameworks to design deadlock-free routing algorithms. A known limitation with CDGs is that they are overly restrictive from a deadlock point of view.

In some situations, a cycle in a CDG can never result in deadlock due to the presence of escape paths. This is identified in (Jayasimha et al. 1996), where a less restricted CWG (channel wait for graph) is proposed for design of deadlock-free routing algorithms. Instead of recognizing dependencies between *possible* channel usages, as the CDG, the CWG is based on dependencies where a message *actually* can be blocked indefinitely by other messages.

ASCDGs are indirectly based on channel dependencies, since application specific dependencies are identified as potential communications traversing consecutive channels. Therefore, like CDGs, ASCDGs also impose more restrictions than necessary to guarantee freedom from deadlocks. To demonstrate that a routing algorithm can be deadlock free even though there are cycles in the ASCDG, consider the topology and communication graph in Figure 4-11.

For simplicity, assume that each task is mapped to a resource with the same index. The resulting ASCDG contains two cycles. According to the channel dependency based APSRA methodology both of these cycles have to be removed. However, looking at the

ASCDG and the dependency from I_{45} to I_{52}, this is due to a communication that starts from node 4 and ends at node 3. But the communication from node 4 to node 3 can take the route to node 6 instead of node 2, if it is blocked by the communication from node 5 to node 1, at node 5.

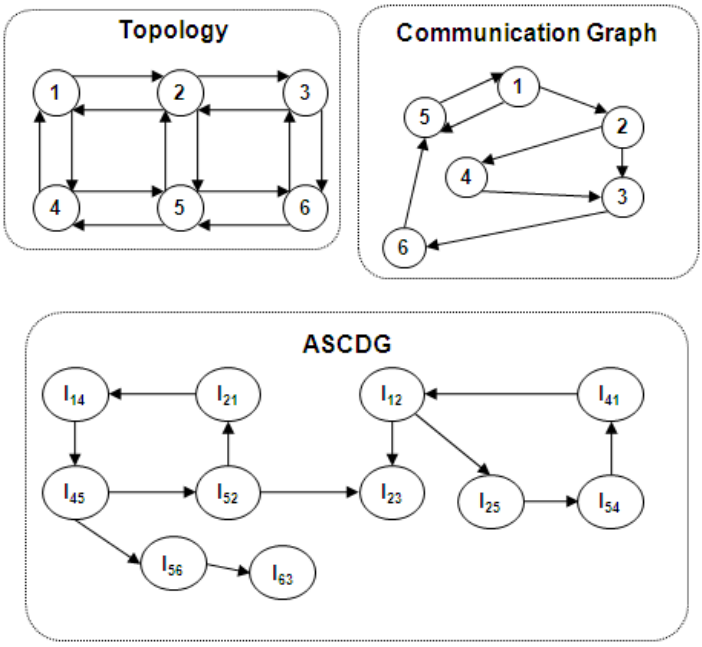


Figure 4-11: Example with two cycles ASCDG

Therefore, the ASCDG contains a cycle which cannot result in a deadlock. Since an escape path is provided, this would not result in a cycle if the application specific dependencies were based on waiting dependencies, like the CWG, rather than channel dependencies as the CDG.

Note that the assumption for the traffic to be deadlock free in the CWG context, is that an input buffer in a node must be empty before accepting a new message header. This condition is also necessary in Duato's methodology for deadlock freedom using virtual channels. Without this assumption, in the above example, a message from node 4 to node 3 can be indefinitely blocked behind a message from node 5 to node 1, in the input buffer of node 2, with no possibility to continue to node 3.

4.7 Discussion

This chapter has described two different approaches for routing in partially regular 2-D mesh topology NoC. Usage of these routing algorithms is not limited to on-chip networks, although the motivation to find feasible routing solutions was driven by the variability in size of the on-chip resources. The fault-tolerant routing algorithm was initially not

thought for on-chip networks. Therefore, the application of the improved version of this fault-tolerant algorithm naturally extends to larger scale networks.

APSRA, on the other hand, is directly developed with NoC in mind. NoC, as compared to larger computer networks, are more likely to be used in embedded systems that are optimized for specific applications. Nevertheless, the ideas may also be applicable to larger scale networks where some degrees of specialization exist, due to specific communication patterns.

One advantage of the fault-tolerant algorithm is its ability to tolerate faults in a network. The importance of this feature may grow as a result of decreasing yield in the chip manufacturing process. In this respect, APSRA generated algorithms do not immediately point towards higher fault-tolerance. But, considering static faults detected after the manufacturing process but prior to network configuration, APSRA can also help to utilize partly faulty chips by providing a larger number of paths as compared to a non APSRA solution. This direction has been explored by (Frazzetta et al. 2008).

5

Effect of Regions on Network Performance

Insertion of regions in a regular network in effect transforms it into a partially regular structure, since some routers and links need to be removed for allowing adequate area to oversized resources. The irregularities are likely to affect the communication performance provided by the network. One important effect of regions is that some messages are forced to use relatively longer routes to circumvent a region. In addition, routing algorithms for regular topologies cannot be used or arbitrarily modified for NoC with regions. Routing restrictions must be carefully arranged such that connectivity and deadlock freedom is secured in spite of region blockages. Therefore can routing algorithms which are applicable for NoC with regions, provide different routes than routing algorithms for regular mesh NoC. This difference can in turn influence the achievable network performance.

Communication performance is not only affected by the obstructive nature of a region. A region, as mentioned in Chapter 3, has the possibility to use several network routers as access-points to the surrounding network. This chapter evaluates through network simulations, the performance impact of incorporating regions of different sizes in various positions in a network. It also studies the effect of single versus multiple access points.

5.1 Design Space for NoC with Regions

Although the shape of a region is assumed to be rectangular, other parameters like size, position and access-points may influence traffic in different ways. An efficient routing algorithm is important for communication performance. Still, it is quite possible that routing performance can differ due to combined effects of region parameters and routing algorithm.

An apparent effect of including a larger sized region in a 2-D mesh is the immediate impact on network traffic. Two important adverse effects are the following:

- Increased route length due to region circumvention
- Congestion on region boundary

The evaluations in this chapter study a small part of the design space; the effect of blockage that a region imposes on network traffic. Several set-ups with different region parameters are simulated to get an overall view of the region blockage effect. The improved fault-tolerant routing algorithm presented in Chapter 4 is used in the evaluations. The main work in this chapter is based on material from (Holsmark & Kumar 2005).

5.2 Evaluation Method

5.2.1 Simulator Parameters

A simulation model of a 7X7 mesh topology NoC with regions was developed using Telelogic's (now IBM Rational) SDL (Specification and Description Language) tool. The minimal node number of both the row and column dimensions, (Row, Col) = (1, 1), are located at the northwest corner of the network. Regions in a network are defined by the coordinates (Row, Col; Row, Col) of the northwest corner node and southeast corner node (northwest; southeast).

The model implements wormhole switching with a packet size of 10 flits. Each router is equipped with a one flit input buffer, similar to (Chen & Chiu 1998). There is minimal delay from the input of one router to the input of a neighboring router of three clock-cycles, and a maximum link bandwidth of 0.5 flits/clock-cycle.

The flits in a packet are sent in a burst at a rate equal to the link bandwidth and the time-gap between the packets follow a Poisson distribution with λ=8. In all simulations, packets are generated during 200000 clock-cycles with warm-up duration of 50000 clock-cycles. The destinations for generated packets are uniform randomly distributed over the network.

5.2.2 Performance Parameters

Performance results for each simulation are presented with respect to offered load (injected traffic). Note that offered load for all results is given as a fraction of the capacity for a regular 2-D mesh network. The following parameters are used to study the performance on the NoC platform:

Average Latency (Flit): The average transmission delay of a flit from a source to a destination. This gives an overall view of how the performance in the network is affected by changes in network configuration and offered load.

Switched Flits/Router: The total number of flits that were switched by each router during a simulation. These values give an indication of the traffic distribution in the network.

Blocked Routing Cycles/Router: The total number of routing cycles during which flits were blocked by other traffic in a router. This can give information of where the network is most congested.

The average latency at low load can be expected to be close to the average *zero load* latency. Average zero load latency can be estimated by considering the average number of hops over all packets sent during a simulation.

As shown in Appendix I, the average number of hops $\overline{H}_{RC}$, in a mesh network of R rows and C columns with uniformly distributed traffic destinations is $\overline{H}_{RC} = (R + C)/3$. Adding one extra hop for the source router gives the average number of *router* hops $H = \overline{H}_{RC} + 1$.

Then a packet header on average traverses H routers with router delay t_r time units. As latencies in the simulations are collected for each flit, all flits will have the same latency as the header. The average zero load flit latency T_{0f} can then be estimated by:

$$T_{0f} = t_r * H = t_r * \left(\left(\frac{R + C}{3}\right) + 1\right)$$

Considering the simulation platform parameters: $H = \left(\frac{14}{3}\right) + 1 \approx 5{,}33,\ t_r = 3$

$$T_{0f} = 3 * 5{,}33 \approx 16$$

Thus, the average zero load flit latency is 16. Since there is an additional delay of 2 time units in the output buffer of the resource, the minimum average zero load flit latency is increased to 18.

This value is a statistical lower bound for latency of uniform traffic and should be close to simulation results at low network load.

5.3 Effect of Placement and Size

5.3.1 Effect of Region Size on Latency and Congestion

The first experiments compare the performance of a network with and without a region. Performance values are obtained for four cases using the modified Chen and Chiu algorithm given in Chapter 4:

1. Network without region: Find the average latency in a region free (fault-free) network. The fault tolerant mode of the routing algorithm is not activated in this case.

2. Network with blocking region: Find the average latency in a network where resources, routers and links covered by a region are inactive. This case activates the fault tolerant mode of the routing algorithm.

3. Network with non-blocking region: Find the average latency in a network where only resources covered by a region are inactive. However all routers and links are active. Therefore, similar to Case 1, this case does not activate the fault tolerant mode. But, as resources are inactive, the traffic generation properties are equal to Case 2. This case shows region routing performance with similar traffic generation conditions as in a regular network.

4. Increased size of blocking region: Find the average latency when increasing the size of the blocking region described in Case 2.

The 7X7 network set-ups are shown in Figure 5-1. Figure 5-1(a) illustrates Case 3 where 4 resources (middle of the network with dashed lines around the 4 tiles) covered by a region at position (3,3;5,5) are inactive but the routers are non-blocking.

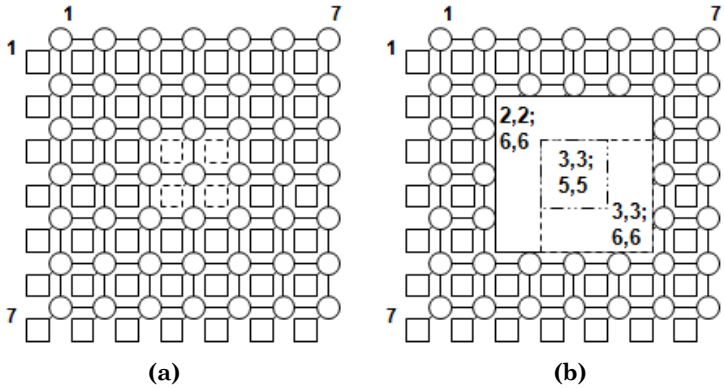


Figure 5-1: Network configuration for region effect experiment: (a) non-blocking region and (b) configurations of blocking region

Position (3,3;5,5) is also used for simulations of the same sized blocking region. Figure 5-1(b) illustrates the configurations where the blocking region is increased to sizes 3X3 and 4X4, with corner coordinates (3,3;6,6) and (2,2;6,6) respectively.

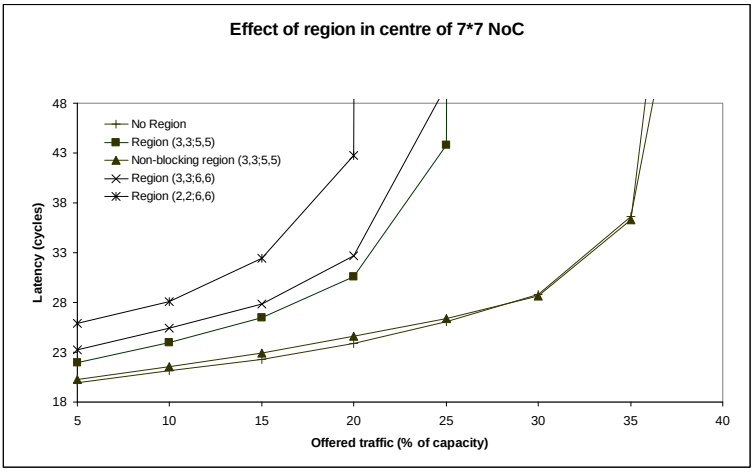


Figure 5-2: Average latency with and without region in NoC

Simulation results in Figure 5-2 show that the average latency without region and with the non-blocking region is quite similar, where the non-blocking region case gives slightly higher latency at moderate loads. This higher latency is a result of increased average distance, due to the disabled resources in the centre. At higher loads the lower numbers of active resources instead seem to result in less congestion and thus, relatively lower latency. In the blocking region simulations, the latency is noticeably higher since messages have to take longer routes to pass around the region. As can be seen, the latency increases with region size. Sensitivity to load also grows with size and it is not possible to handle more than 20% load for a NoC with one 4X4 region positioned according to Figure 5-1(b).

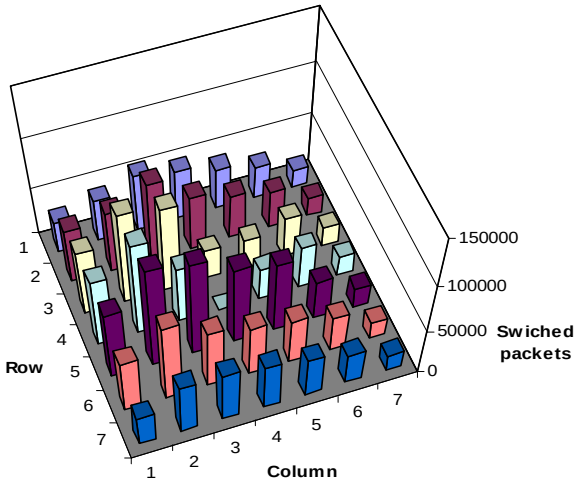


Figure 5-3: Switched packets by routers in the network at 20% load

To further investigate traffic behavior, a study on the switching activity of network routers was performed. The study was made by summing up the number of switched flits for each router during one simulation run. The results shown in Figure 5-3 are obtained from a simulation at 20% load with a region inserted at position in centre (3,3;5,5).

It can be seen that corners of the network are least used and that the major switching activity is performed by routers located around the crossing of the fifth row and second column of the network. There are two main properties in the routing algorithm that can explain this. One is that there is a bias in the routing algorithm which induces more traffic to the west of the network. The other is that the region activates the fault-tolerance mode of the algorithm, which in this case makes more packets to travel through the column which is one step to the west of its west edge. More traffic on the south border of the region is caused by the rule that all RO (row-only) packets have to pass on this row.

The effect of the region is even more evident in Figure 5-4. Instead of switched flits in routers, this figure captures the number of routing cycles/router when flits were blocked. As seen in Figure 5-4, most packets have been blocked in router (3, 2). The reason is that all packets, which are affected by the region on the north border and heading south, will have to use this for turning to south. Router (3, 3) is also quite blocked because packets heading south have to go west here coming from the north border. The other heavily blocked routers are the ones on the south border as a result of all RO and northbound CF (column first) messages traversing these routes.

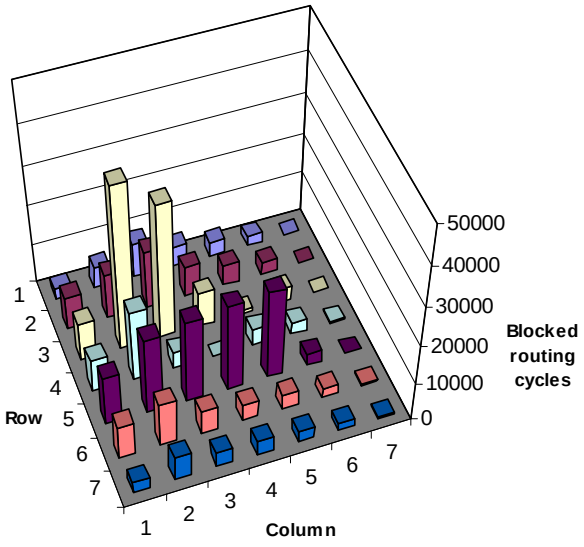


Figure 5-4: Blocked routing cycles in routers in the network at 20% load

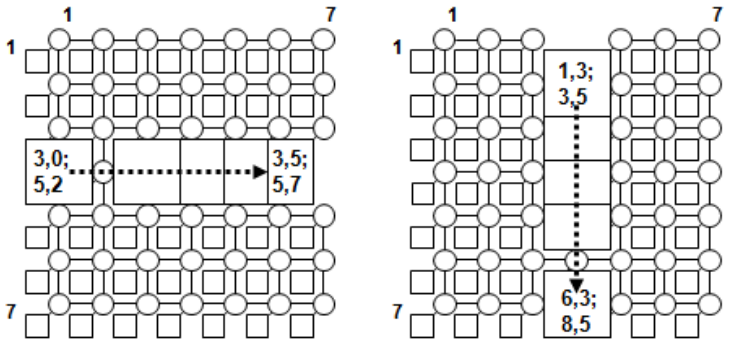
To summarize the results, it is clear that increased region size results in worse packet latencies. The analysis of blocked packets reveals that the routing algorithm produce uneven distribution of congestion, both with respect to routes on the region border as well as normal routes. This effect has negative impact on network performance.

5.3.2 Effect of Region Position on Latency

The evaluations in this section study the effect of region position with respect to latency. Figure 5-5(a) gives the set-up for a 2X2 region, initially placed with northwest corner at row three, column zero (crossing the west side of the NoC). The region is then moved from the left side of the NoC to the east side, as shown by the arrow in Figure 5-5(a). Note that the west-most region blocks fewer routers, as a result of the placement of resources.

Figure 5-6 shows the simulation results with load from 5 % up to 25 %. Note that the x-axis indicates the column position of the northwest corner of the region. A similar pattern can be recognized for all load values, in that the worst position from latency point of view is when the northwest corner is in the second column. The latency is also strongly affected by increased load, especially in this position. While latency is moderate in all other positions at 25 % load, at this position it becomes extremely high.

Again, this is due to the biased behavior of the routing algorithm which causes more congestion towards the west part of both the network and the region. The exception is in this case the west-most region that has the lowest latency. This is natural though, since it blocks fewer routes as compared to the other positions.



(a) **(b)**

Figure 5-5: Set-ups for (a) horizontal and (b) vertical shift of position for 2X2 region

As illustrated in Figure 5-5(b), a similar experiment was made by shifting a region of the same size in vertical position. The results in Figure 5-7 show that the highest latency is obtained with a region in the central position with decreasing values towards the edges.

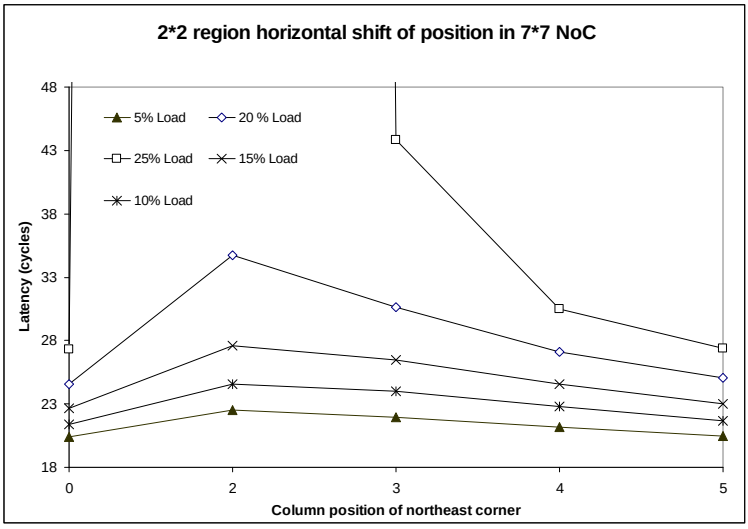


Figure 5-6: Latency for positions of horizontal shift of 2X2 region

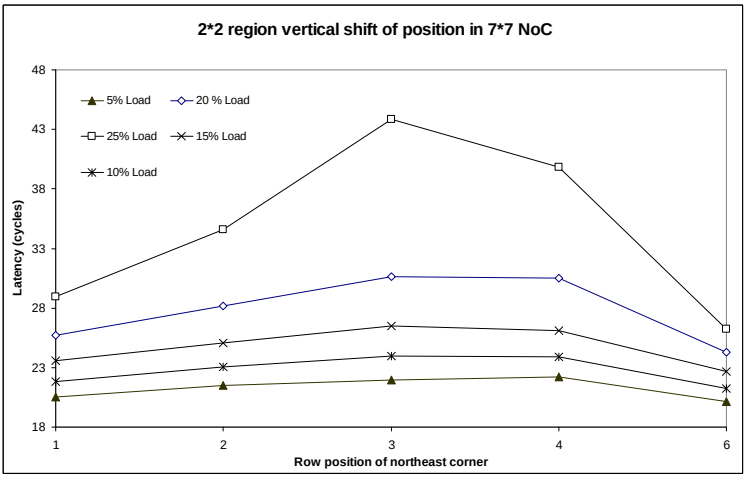


Figure 5-7: Latency for positions of vertical shift of 2X2 region

Considering all set-ups in Figure 5-5, the best position of a region seems to be towards northeast or southeast in the network. As the routes of the algorithm are biased towards west, both in the normal mode and in the fault tolerant mode, this conclusion seems logical.

5.3.3 Effect of Region Orientation on Latency

Figure 5-8 illustrates the set-ups for experiments on orientation of a region with a size of eight tiles. Latency results from these configurations, together with results from previous region configurations, are given in Figure 5-9.

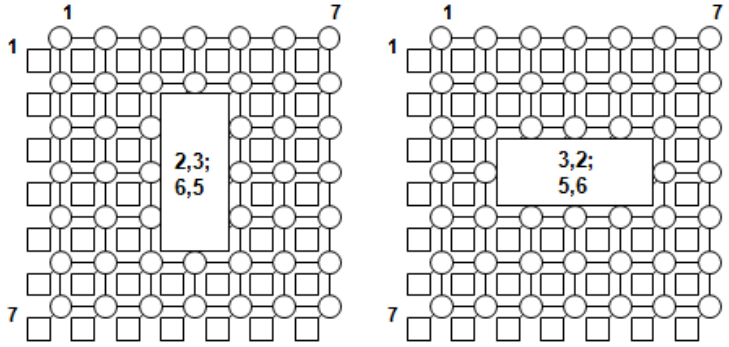


Figure 5-8: Non-square regions used in evaluation of orientation

It can be seen that the region with position (2,3;6,5) has higher latency than the region with position (3,3;5,5) but lower latency as compared to the largest region with position (2,2;6,6). Interestingly, the worst latency is found for the right-hand configuration in Figure 5-8 (3,2;5,6), which though it is smaller than (2,2;6,6) in Figure 5-1(b), actually results in higher latency at higher load than 5 %.

The reason for this behavior can be that the network is getting congested earlier because of the larger number of senders that generate packets in this case. When the network is less congested, latency is more reflecting the shorter distance every packet has to travel.

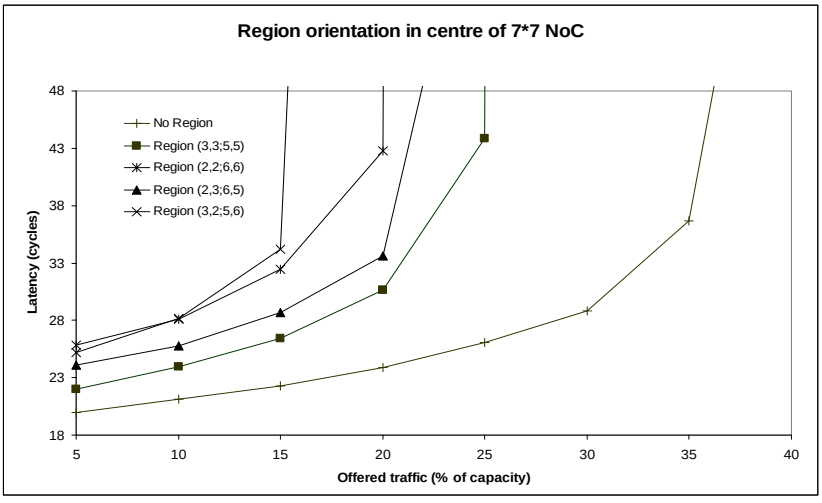


Figure 5-9: Latency comparison with non-square regions

The orientation experiment clearly shows the impact of the biased congestion from the routing algorithm. Results with the region more aligned towards the most congested side (3,2;5,6) of the network show significantly higher latency as compared to results with the other region orientation (2,3;6,5).

5.3.4 Effect of Multiple Regions on Latency

A final evaluation was made to study the scaling effect of placing multiple 2X2 regions in the network as illustrated in Figure 5-10. The positions of regions were mainly selected on the basis of the following two criterions:

- No effect between regions – regions should be distributed in the network
- Mixed effects of routing algorithm - regions at both "good" and "bad" positions

It can be seen in Figure 5-11 that the average latency with these three regions is just a little more than with one region at position (3,4;5,6). One circumstance can explain the modest effect on latency when increasing the number of regions. There are fewer senders in the case of multiple regions, which results in reduced amounts of network traffic and consequently lower probability of congestion.

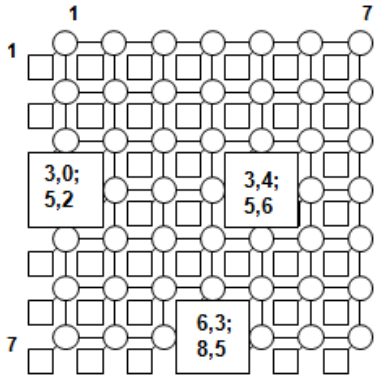


Figure 5-10: Position of multiple regions

An interesting observation in Figure 5-11 is that a single region at position (6,3;8,5) has almost no adverse effect on performance as compared to a region-free network.

In all, we find that *multiple regions*, in this limited evaluation, do not have a large effect on network performance. Instead, results indicate that latency is largely determined by the worst positioned region. This is assuming that regions are spaced with adequate distances with respect to each other.

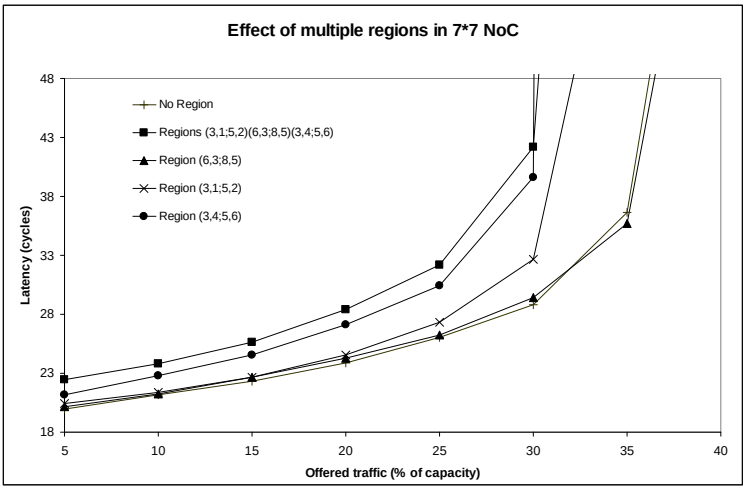
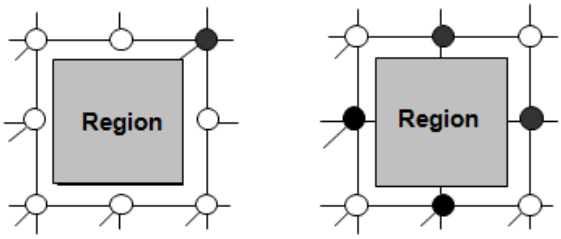


Figure 5-11: Latency comparison with multiple regions

5.4 Effect of Different Number of Access Points

Whereas normal sized resources are connected to only one router, regions can be connected to several *access points* due to their large size. In this section, the same routing algorithm and simulator as in the previous section, is used for a study on region access performance. Two different configurations of access points for a region of four slots, at position (3,3;5,5) are compared.

The set-ups are shown in Figure 5-12, where (a) illustrates the case of using the upper right corner router as a single access point, and (b) shows the configuration where 4 routers are used as access points (one router on each side).



(a) **(b)**

Figure 5-12: (a) Single access point for region and (b) multiple access points for region

Figure 5-13 gives the simulation results for these two cases plus, for reference, latency for a case without regions. Results indicate that using one access point on each side, results in about 5-15% lower latency, for load values stretching from 5-20%. This configuration also reaches the saturation point later. Because the region obstructs traffic passing through it, none of the region cases can match the case without region.

The reason for the higher performance using several access points is that the average distance is shorter between the sources and destinations in the network. This is because packets emitted from, or destined to the region do not have to circumvent it to reach the access point. Considering larger regions with relatively longer distances to a single access-point, it is likely that the advantage of using multiple access points will be even greater.

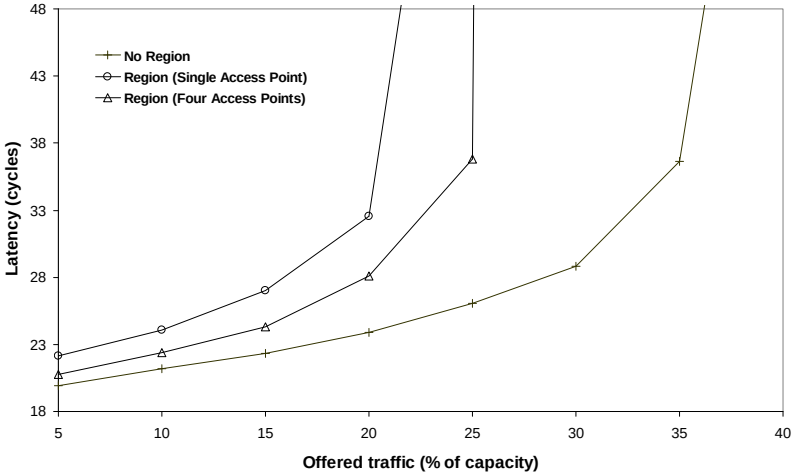


Figure 5-13: Number of access points effect on latency

The enhancement in performance using multiple access points comes at some cost. Routing protocols for regions with multiple access points will require extra bits to decide whether a packet is to be delivered to the access point of a region or a regular resource. This overhead depends on the number of access points. Multiple access points may also result in additional hardware in the region, for multiplexing and buffering data received through various physical access ports and for resolving conflicts. Extra hardware may also be required for de-multiplexing and distributing the messages leaving the region through different access points.

5.5 Discussion and Conclusions

This chapter evaluated the effect of regions on network performance, both with respect to different region configurations as well as in comparison with a regular network. The region concept allows for more efficient resource usage in mesh topology NoC architectures. But as expected, simulation results show that a region has a negative impact on communication performance.

Results indicate that the routing algorithm has a significant influence on communication efficiency. One effect is the bias of the routing algorithm in normal mode, which utilizes relatively more routing resources towards the west side of a mesh network. Another effect stems from the fault tolerant mode, which distributes more traffic to the west edge of a region. Still, by studying cases where the algorithm effect is similar, like vertical shift,

we see that position and size of regions affect average latency. Worse results are obtained for the larger or the more centralized a region is. The experiment of multiple regions shows that it is possible to position multiple large-sized cores, without affecting performance drastically.

Considering both region parameters and routing algorithm impact, the optimal position of a region is likely to be to the northeast or southeast corners of a network. Some results even show that the adverse effects of a region at certain positions are very small. Results also indicate that use of multiple access-points to regions can help general network communication performance. However, there is some extra hardware and protocol cost as a result of the increased routing complexity.

We will show further in the next chapter that the choice of routing algorithm significantly affects the performance in a NoC with regions. We expect that the conclusion about size and access-points will hold for also other algorithms. The conclusions on shape and placement may though vary, depending on the employed routing algorithm.

6

Comparative Evaluation of Two Routing Algorithms

As discussed in previous chapters, it is quite possible that different routing algorithms vary significantly in lower level functionality, even if they are developed for identical network structures. The difference in functionality is often related to specific objectives with each algorithm. In many cases, the objectives imply requirements that have a negative impact on network performance. For example, Chapter 5 showed that the intricate combination of fault tolerance objectives and deadlock freedom requirements, in the case of Chen and Chiu's routing algorithm, led to highly congested routes, especially in NoC with regions.

This chapter compares the improved version of Chen and Chiu's algorithm with a routing algorithm developed using the APSRA methodology. Though the algorithms are very different in many aspects, one important property is the same; they do not employ virtual channels to achieve deadlock freedom.

The two routing algorithms are evaluated for various instances of a mesh network with regions. The study includes analysis of adaptivity and how it is affected by different sizes and placement of regions. Routing performance is estimated by network simulations using both synthetic and application based traffic patterns.

6.1 Evaluation Method and Objectives

Over the years, routing algorithms have been evaluated and compared in numerous ways. A common approach when comparing algorithms is to utilize different network models. One type of models is the synthetic, which have no direct relation to a real system. Still, their general properties make them useful when analyzing and comparing the behaviour of different algorithms. Application models, on the other hand, are more close to real systems. But, since a specific system is modelled, the generality of the results are naturally quite low.

In this chapter, APSRA and Chen and Chiu's routing algorithm are compared using both synthetic and application based traffic patterns. The synthetic evaluations include both analysis of adaptivity and network simulations. The application based evaluation is performed by simulating a communication pattern from a multimedia application described in (Srinivasan & Chatha 2006).

This chapter is based on work published in (Holsmark et al. 2006) and (Holsmark et al. 2008).

6.2 Synthetic Network Model

The synthetic models assume that a region is an active resource that emits and receives messages. Four cases of a 7X7 NoC with a 2X2 region are considered in the evaluations:

- Center, 4AP: Centrally placed region with four access points
- Center, 1AP: Centrally placed region with one accesss point
- BL, 3AP : Bottom left corner placed region with three access points
- BL, 1AP: Bottom left corner placed region with one access point

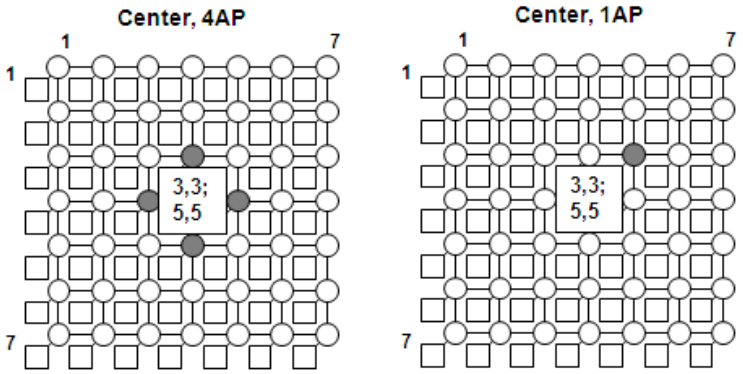


Figure 6-1: Center region with (a) four access points and (b) one access point

Figure **6-1** illustrates the two cases with a centrally placed region. Figure 6-1(a) shows the four access point scenario, where one node on each side is used as access point. Figure 6-1(b) illustrate the set-up with a single access point. Figure 6-2 gives the two network set-ups with the bottom left corner region. The region shown in Figure 6-2(a) uses three access points, which in this case is the maximum number of access points available. The configuration shown in Figure 6-2(b) utilizes one access point at the top right corner, which is the same relative position of the region as in Figure 6-1(b).

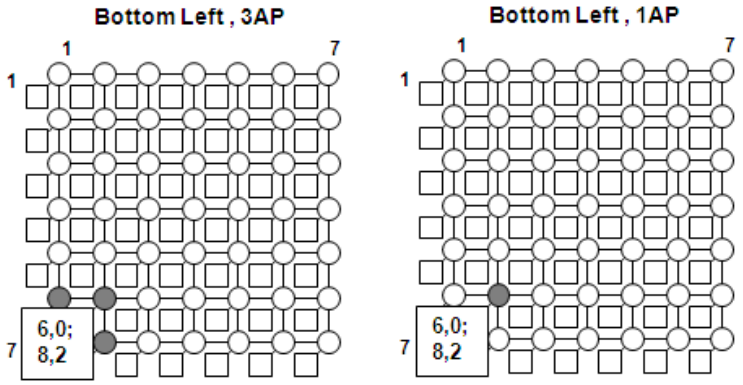


Figure 6-2: Bottom left region with (a) three access points and (b) one access point

6.3 Adaptivity Analysis

An adaptive routing algorithm may return several possible paths from a source node to a destination node. The extent to which a routing algorithm provides multiple paths for all network communications can be captured by analyzing *adaptivity* or *degree of adaptiveness*. The definintion of adaptiveness that is used in this section is given in Chapter 4. In short, it is defined as the *average of the degree of adaptiveness of all communicating pairs*. For a given source destination pair, the degree of adaptiveness is defined as the ratio between the number of *allowed* paths and the total number of *minimal* paths connecting the source node with the destination node. Note that by this definition, even a connected non-adaptive routing algorithm always has a degree of adaptiveness above zero.

Figure 6-3 shows the degree of adaptiveness of (adaptive) APSRA and (non-adaptive) Chen and Chiu's routing algorithm for the network configurations given in Figure 6-1 and Figure 6-2. The two left-most bars represent the degree of adaptiveness for a region placed at the center of the network with 4 access points (Figure 6-1(a)) and 1 access point (Figure 6-1(b)). The bars to the right represent the bottom left corner region configuration with 3 access points (Figure 6-2(a)) and 1 access point (Figure 6-2(b)).

It is clear that the degree of adaptiveness is not much affected by either the number of access points or position of the region. Nevertheless, it is evident that APSRA provides significantly more routes as compared to Chen and Chiu's routing algorithm.

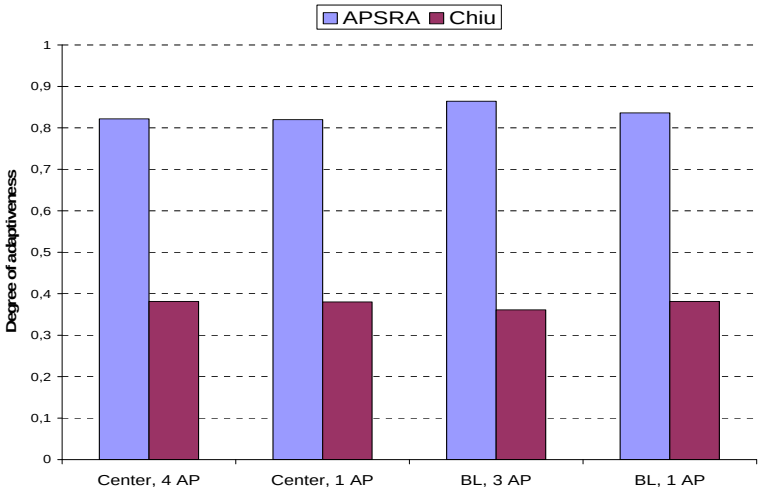


Figure 6-3: Adaptiveness vs. access points and placement of regions

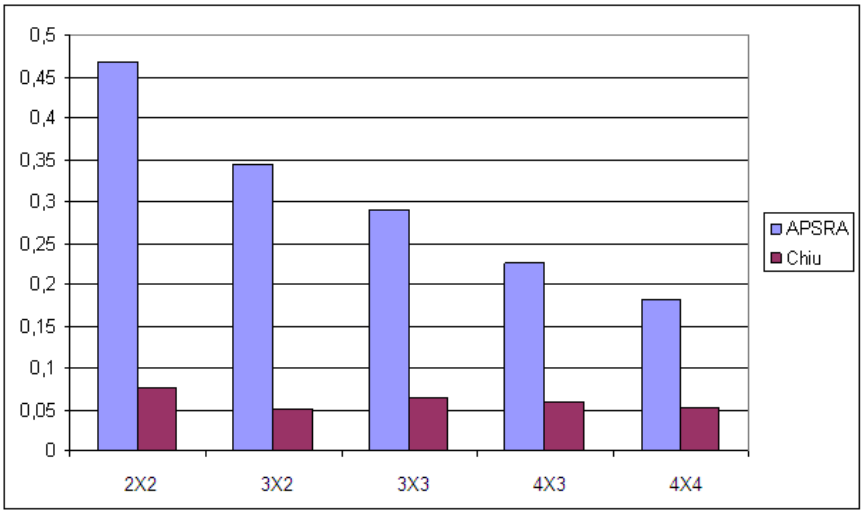


Figure 6-4: Relative adaptiveness vs. size of region for region in centre

To compare the adaptivity for different region sizes, we define a new adaptivity indicator called *relative adaptivity*. It represents the ratio between the total number of allowed paths *when a region is present* and the number of paths *without a region*. Figure 6-4 gives the relative adaptivity for different region sizes located at the center of the NoC, whereas

Figure 6-5 shows the results for regions located at the bottom left corner of the NoC. For all configurations, the region access point is located at the top right corner.

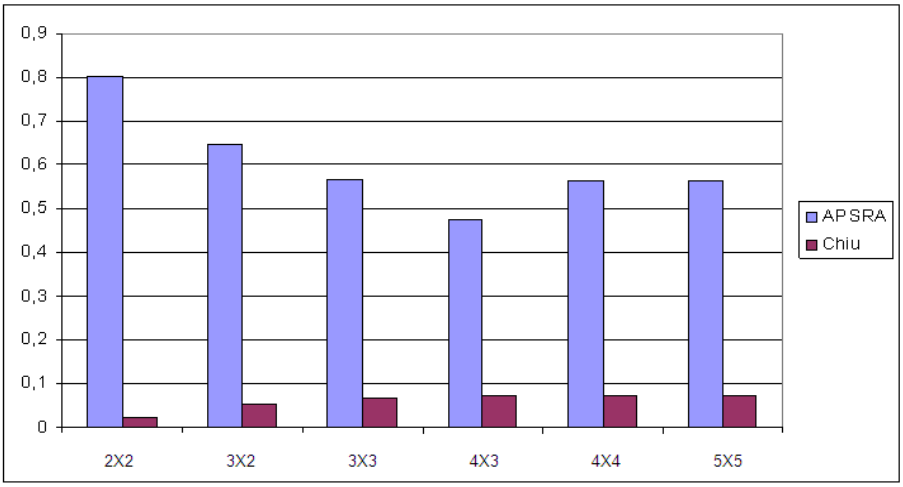


Figure 6-5: Relative adaptiveness vs. size of region for region in bottom left corner

As expected, the relative adaptivity decreases with the increase in size of the region in general. For regions located at the corner of the NoC there is a minimum in relative adaptivity when the region size is 4X4 (or half the dimension of the mesh NoC). If region size increases further, the relative adaptivity increases.

This effect is caused by the fact that a region, located at the bottom left corner of the NoC, obstructs only communications between nodes located at the northwest and southeast quadrants with respect to the region. The number of paths without region decreases on average (because the access point moves in direction of the center of the NoC), while the number of paths remains fairly the same when region size increases from 4X4 to 5X4 and further to 5X5.

6.4 Network Simulations with Synthetic Traffic

The two algorithms were evaluated using a similar simulator model as described in Chapter 5. The parameters *Average Latency* and *Blocked Routing Cycles/Router*, described in Section 5.2.2, are again used for characterizing network performance. Note that the measurements here are taken with respect to packets and not flits and that performance data are collected over 60 000 packets, after a warm-up session of 30 000 packets. Since the source destination pairs are constant during one simulation run, latency values are averaged over 5 random traffic scenarios to reduce the risk of exceptional cases.

We evaluate APSRA and Chen and Chiu's algorithm with a region either in the bottom left corner with 3 access points (Figure 6-2(a)) or in the centre of the network with 4

access points (Figure 6-1(a)). Simulations are performed with a synthetic communication pattern, where the destinations for generated packets are randomly selected with hot-spot probability of 60 % for the region access points. The increased region traffic reflects the assumption that a larger resource is more frequently accessed like, for example, a shared memory.

Network traffic is classified into three types as follows: (1) communication **traffic to region**, (2) **other traffic** where a resource other than the region is a destination, and (3) **all communications** which is the aggregate of the first two types of traffic.

6.4.1 Average Latency for *All Communications*

The first results, in Figure 6-6, show average latency for all communications in the network. The lowest latency is obtained by APSRA with central region (apsra_c_ap4). The second lowest latency values are achieved with Chen and Chiu's algorithm and central region (chiu_c_ap4).For the case when the region is placed at the bottom left corner, APSRA (apsra_bl_ap3) again shows lower average latency than Chen and Chiu's algorithm (chiu_bl_ap3). The difference is not as large, compared with the central region set-up, but grows with increased load.

Although APSRA displays lower latency for identical network configurations, results indicate that the position of the region has a higher impact than the used routing algorithm.

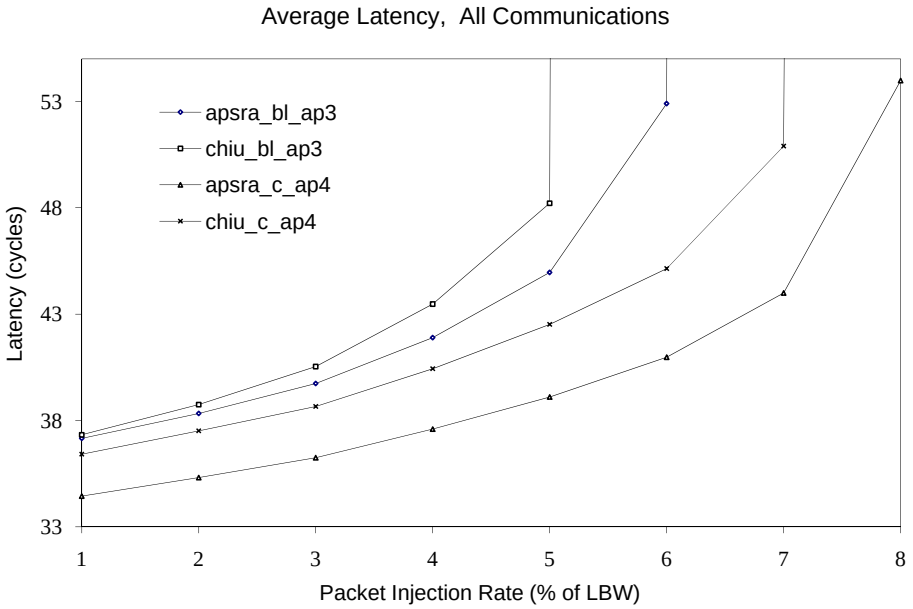


Figure 6-6: Average latency for all communications, with region in bottom left (bl) and centre (c), vs. injection rate in % of link bandwidth (LBW)

6.4.2 Average Latency for *Other Traffic*

In Figure 6-7, we give the average latency for traffic with destinations *other than* the region. The worst results with respect to latency, up to an injection rate of 5%, are produced by Chen and Chiu's algorithm and the region in centre (chiu_c_ap4). In this case, all the other configurations show similar latency results.

However, when injection rate is increased above 5%, Chen and Chiu's algorithm with the region in corner position (chiu_bl_ap3) rapidly saturates. Next to saturation is APSRA with the region in corner (apsra_bl_ap3). The best result for higher load is obtained by APSRA with region in centre (apsra_c_ap4), although it has slightly higher latency at lower injection rates. In any case, placing a region in the centre seems to have less impact in the sense of creating severe congestion.

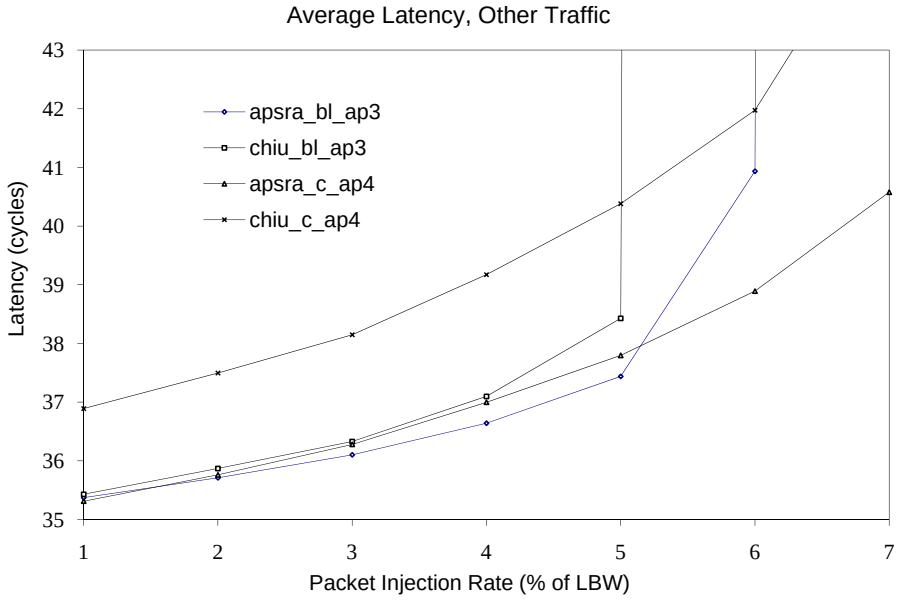


Figure 6-7: Average latency for other traffic with region placed in bottom left (bl) and centre (c), vs. packet injection rate in % of link bandwidth (LBW)

6.4.3 Average Latency for *Traffic to Region*

Figure 6-8 depicts the results on average latency for traffic destined only to region. Also in this case, APSRA with central region achieves the best performance in terms of low latency. It is notable though, that Chen and Chiu's routing algorithm with central region show better results than APSRA with the region at bottom left corner position.

Compared with the *other traffic* case, this is true for all injection rates. Worst performance is also in this measurement obtained by Chen and Chiu's algorithm with the region in the bottom left corner.

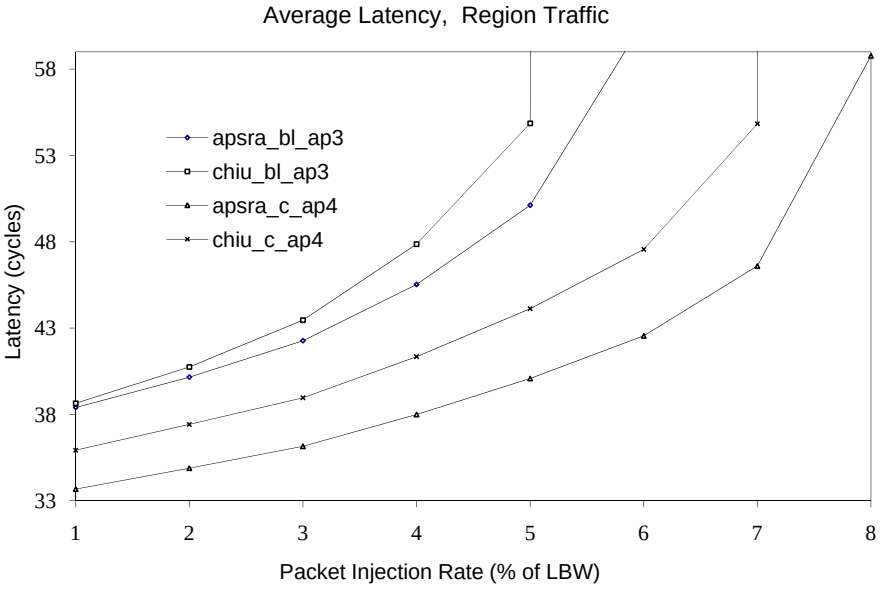


Figure 6-8: Average latency for communications destined to region in bottom left (bl) and centre (c), vs. injection rate in % of link bandwidth (LBW)

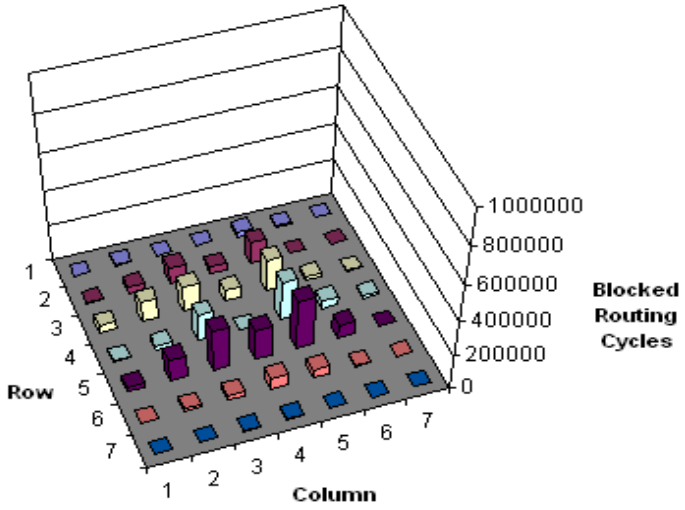
6.4.4 Analysis of Congestion

Figure 6-9 gives a hint to the reason behind the difference in average latency between APSRA and Chen and Chiu's algorithm. Each bar in the diagrams represents the sum of routing cycles when packets were blocked in a particular router. The results are taken from one of the simulations with 10 % packet injection rate, where the difference in latency was very large. Note that the *scale of blocked routing cycles is not the same* in the two diagrams.

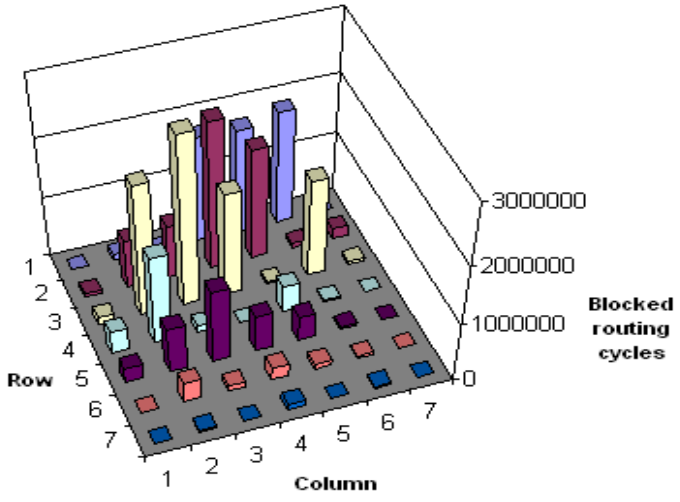
Figure 6-9(a and b) reveals that the APSRA algorithm does not cause as much blockage as Chen and Chiu's algorithm. It can be seen in Figure 6-9(a) that APSRA also causes a higher number of blocked packets around the border of the region. This increase is the result of packets that have to circumvent the region in order to reach its destination. Still, the distribution is even and much smaller than for Chen and Chiu's algorithm in Figure 6-9(b).

Note that Chen and Chiu's algorithm induces more blockage close to the north and west border of the region. Most likely, the reason is that these links are highly utilized by the algorithm in the procedures of routing around the region border. As a result, these paths quickly become congested, which in turn causes more chances of packets getting blocked. APSRA, on the other hand, is not biased towards specific routes and spreads traffic more evenly around the region border. Since APSRA in several cases has multiple paths to

select among, it is also possible to avoid congested routes which further decrease the accumulated blockage.



(a) APSRA algorithm

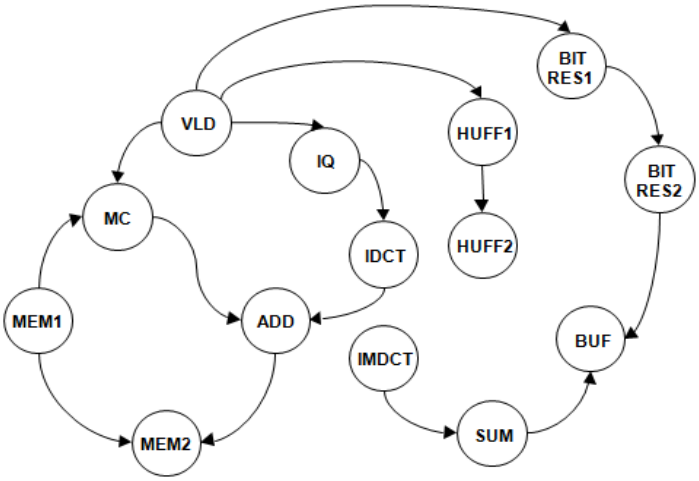


(b) Chen and Chiu's algorithm

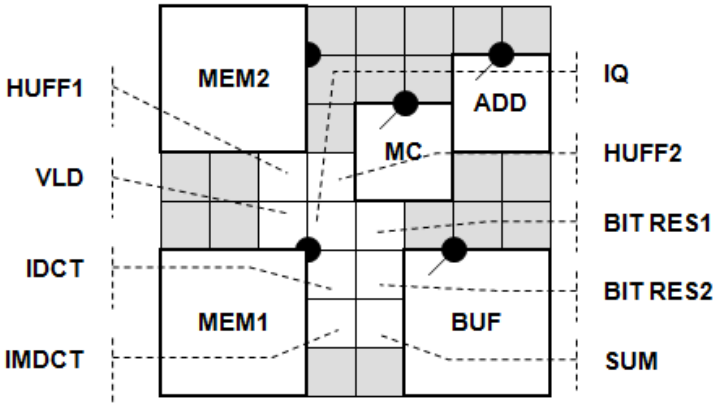
Figure 6-9: Blocked routing cycles/router with (a) APSRA algorithm and (b) Chen and Chiu's algorithm

6.5 Evaluation using Multimedia Application

As a real case study, we considered a multimedia application which implements a H.263 video decoder and a MP3 audio decoder (Srinivasan & Chatha 2006). Figure 6-10(a) shows the task communication graph of the application. Figure 6-10(b) gives the topological structure of the NoC and the mapping of tasks onto the tiles. The mapping was obtained by using a modified version of the approach presented in (Ascia et al. 2004).



(a)Task communication graph



(b)Topology and mapping

Figure 6-10: Multimedia application: (a) task communication graph and (b) topological structure and task mapping

A total of five regions are used in this set-up. Three big regions are used to host two memories and a buffer. Two small regions are dedicated to the motion compensation (MC) block and the ADD block. Each region is provided with one access-point. The location of the access-point is represented with a black dot. The remaining gray tiles of the NoC are assumed to communicate in a random fashion.

The degree of adaptiveness provided by the APSRA routing algorithm is 0.96. Notably, all communications belonging to the audio/video decoder itself can be forwarded using minimal fully adaptive routing. Only a few restrictions on routes are applied to the random traffic.

Figure 6-11 shows the average latency for APSRA and Chen and Chiu's algorithm at different packet injection rates. As can be seen, the APSRA algorithm has an overall performance advantage. The latency at moderate load is lower, and for higher packet injection rates, APSRA routes the communication below saturation up to a packet injection rate of 45 %. For Chen and Chiu's algorithm, this state instead occurs at around 30%.

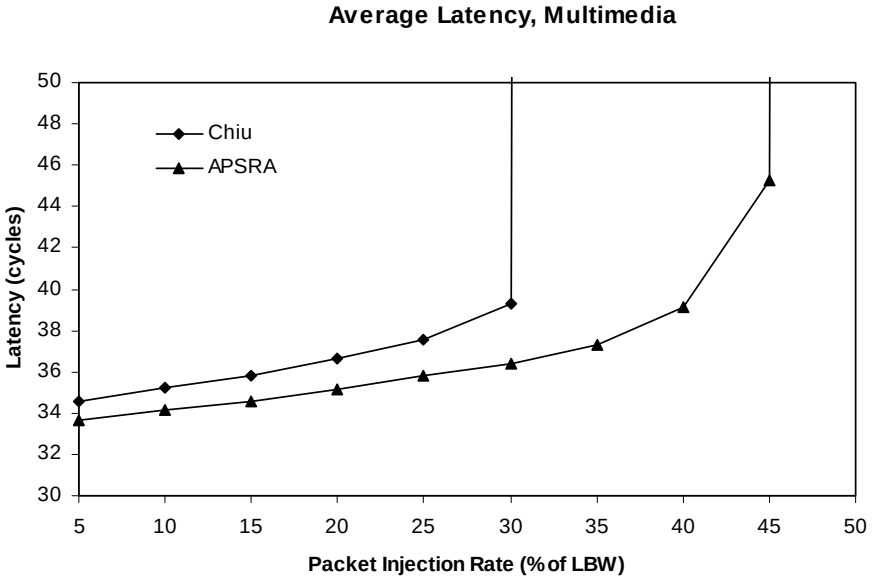


Figure 6-11: Average latency for multimedia application, vs. injection rate in % of link bandwidth (LBW)

6.6 Discussion and Conclusions

The simulation results show that APSRA, for identical traffic scenarios, has an overall advantage in communication latency. This is due to the effect of two important APSRA

characteristics; unbiased routing and high adaptivity. The unbiased behavior has fewer tendencies to *create* highly congested routes. The adaptivity of the APSRA algorithm also makes it possible to *avoid* congested routes.

This is especially shown in the results of traffic not destined to region, where a large difference is shown between APSRA and Chen and Chiu's algorithm for the region in the centre. Considering traffic to region, the latency is more dominated by the distance from sources to the destinations, which in this case is shorter with a centrally placed region. Since traffic to the region has a probability of 60% this also dominates the average latency when we consider the "all communications" case.

A large difference can be identified when comparing injection rates and saturation between the synthetic and multimedia simulations. This is caused mainly by two different properties of the models. First, the number of communications is larger in the synthetic simulations: on average every node generates traffic to two other nodes. Secondly, the hot-spot traffic causes high local traffic rates, which further increases the risk of congestions.

In summary, results show that an APSRA generated algorithm provides higher communication performance. It should be noted though, that the design objective of Chen and Chiu's algorithm is different and that figures of merit reside in other domains. One example is its ability to adapt to topological changes dynamically during run-time. Further, APSRA routing algorithms are implemented by the use of router tables. These may require relatively more chip area than the algorithmic implementation of Chen and Chiu's algorithm. The use of adaptive routes by an APSRA algorithm can result in out of order delivery of packets. This can incur additional cost if reordering buffers or other special mechanisms are required.

It is evident that routing in NoC with regions is less efficient than in a regular NoC. Since network resources are removed by a region in a particular NoC, the existing routes need more sharing. At some positions, regions acts as blockage and prohibit the use of otherwise minimal paths. Our evaluations show that several parameters may improve the performance of a NoC with regions. Two of the most important in this respect are the positions of regions and the choice of routing algorithm.

7

Conclusions

This chapter gives the main conclusions of the thesis work. The first section provides a summary of the contributions. After this follows a section that discusses important limitations and how these may affect the validity of the results. A few ideas on future work are given in the last section.

7.1 Summary of Contributions

7.1.1 Elaboration and Evaluation of the Region Concept

The regular 2-D mesh topology has several advantages for NoC implementation. One serious drawback though, is the area wastage due to cores of various sizes. This thesis elaborated on the region concept, which enables the use of oversized cores. Several aspects on NoC with regions were outlined and discussed:

Routing: Regions adversely affect several aspects of network routing. In particular, the deadlock freedom requirement impacts on the availability of routing algorithms for NoC with regions.

Applications: Regions have the potential of much wider use than only handling oversized cores. The region concept provides the ability to handle parts of a network in

different ways. This allows for reuse of sub-networks, new opportunities for power management, effective QoS implementation or network partitioning.

Design space: The design space of NoC architectures is broadened by allowing regions. Dynamically created regions, access point trade-offs and core/application mapping with regions provide new design space exploration parameters.

Performance: The general communication performance of a NoC decreases when including regions. The magnitude of the performance reduction is dependent on both the routing algorithm and the region placement.

In all, it is difficult to conclude whether the use of regions is good or bad for NoC design. Nevertheless, this work has identified important drawbacks, advantages and possibilities of the region concept.

7.1.2 Correction of the Fault-Tolerant Routing Algorithm

Even minor topological changes of the 2-D mesh topology, combined with the deadlock freedom requirement, may turn efficient routing algorithms for regular mesh unusable. Other types of routing algorithms are therefore necessary. Fault-tolerant routing algorithms for mesh often assume a fault model with faulty blocks that are similarly shaped as regions.

During our work on the region concept, erroneous behavior was identified in a fault-tolerant routing algorithm proposed by (Chen & Chiu 1998). Our investigation found that the algorithm, which was claimed deadlock free, was in fact not deadlock free. The algorithm was also incomplete (not connected), since some nodes were unreachable from certain other nodes.

This thesis proposed corrections to the original algorithm, such that deadlock freedom and connectivity are ensured. A theoretical proof was provided for the correctness and completeness of the improved algorithm.

7.1.3 Development of APSRA

Adaptivity in routing algorithms is strictly limited by requirements of deadlock freedom. Implementation of virtual channels can enable higher adaptivity, but will also increase interconnect resource requirements. The application specific nature of System on Chip (SoC) applications gives new possibilities for routing algorithm optimization. Such opportunities are less available for off-chip networks.

Our work in this area has resulted in a methodology for designing deadlock-free, highly adaptive *application specific* routing algorithms. A task communication graph and the channel dependency graph (CDG) of the topology, are used for constructing an application specific channel dependency graph (ASCDG). The ASCDG is likely to have fewer cycles as compared to the CDG. Cycles that do exist are removed with the objective

of maximizing adaptivity. Once an acyclic ASCDG is obtained, router tables are created and programmed into network routers.

In conclusion, APSRA provides a new way to utilize application knowledge for optimizing deadlock-free routing algorithms in NoC systems.

7.1.4 Evaluation of Two Routing Algorithms

The two proposed routing algorithms for mesh NoC with regions were evaluated in a comparative study. The algorithms differ on several important aspects. The table-based implementation required by APSRA may utilize more resources, especially for large networks since table size grows with the size of the network. The adaptivity provided by APSRA is high, but, on the other hand dependent on the application.

Network simulations showed that algorithms designed using the APSRA methodology outperform Chen and Chiu's algorithm in communication performance. Two main reasons were identified. First, by only considering actual communication it is possible to select more favorable routes at design time. Secondly, high adaptivity at run-time provides the possibility of using alternate routes to avoid congestion. The evaluation also shows that the position of region is important, and in some cases influences performance more than the choice of routing algorithm.

We conclude that APSRA undoubtedly provides the highest routing performance. On the other hand, Chen and Chiu's algorithm is more attractive with respect to objectives like fault tolerance or general communication support.

7.2 Limitations

The work in this thesis has a few limitations. The severity of these issues, with respect to the validity of the presented results is varying. They are also related to what claims and conclusions are made in the thesis. A general limitation is that the ideas and results in the thesis are quite abstract. Moreover, the main work in the thesis is devoted to the area waste/design productivity trade-offs and performance. The important issues on power consumption are less discussed and evaluated. A few more specific limitations are the following:

Region concept: No concrete implementations have been made to verify that a mesh NoC with region is usable in practice. The advantages, possibilities and drawbacks are discussed with no concrete example of a real application.

Evaluated routing algorithms: The region evaluations are only performed with two routing algorithms. The parameters of the routing algorithms are not implementation based. For example, the algorithms are assumed to have identical delays in routers.

Simulation traffic: The synthetic evaluations are performed with only uniform or hot-spot random traffic. The communication model in the real application is only *real* with

respect to communication pairs. Packet injection rates, for example, are synthetic and the existence of other traffic than what is given in the communication graph is ignored.

With respect to the region concept, most of the work is not devoted to finding an optimal solution. Instead, it is more of an investigation to identify problems and possibilities.

Several issues relate to the simulation experiments. It is hard to include and foresee all possible parameters in simulation based evaluation. To get 100% valid results it is actually necessary to model all versions of a system first and then evaluate. Which of course, in turn, make the use of simulation based evaluation at this stage highly questionable.

The advantage of simulations is that certain aspects of network behavior are quickly investigated. Several publications have shown that the effectiveness of different algorithms vary with respect to traffic type. The random destination model used in this thesis is applied to investigate very general behavior. Other synthetic models like transpose are not easily defined for a network with regions. The hot-spot model was used to reflect that a larger region can be assumed to generate and receive more traffic as compared to other cores.

Increased validity of the comparative results could be gained by using more accurate timing models, obtained from actual router implementation. Still, such values validate one type of implementation, which is dependent on, for example, designer expertise. Also, it is common practice that comparative studies of routing algorithms assume identical simulator parameters.

7.3 Future Work

Several issues related to the work in this thesis can be further explored. The following are some examples of possible future work:

Implementation and evaluation: Implementation of a NoC with region prototype, or realistic system model, could provide useful insights on the applicability of the region concept. Especially if implementation aspects are compared with other types of interconnects, both regular as well as irregular.

Design exploration: A lot of research has been done in the area of task and core mapping on regular NoC. Allowing regions in such experiments would add interesting trade-offs. For example, the advantages of grouping few cores within a region with bus-based interconnect, might out-weigh the negative traffic effects of a region. Allowing a few larger, more powerful, memory enhanced multi-threaded cores can provide new challenges in application mapping. Placement of larger sized regions provides a new parameter to core mapping techniques.

Routing in NoC with regions: The use of regions motivates new features in routing algorithms. It is already shown that adaptivity to traffic is positive with respect to performance. Mechanisms that incorporate region placement information and optimize traffic distribution could have high potential.

Routing algorithm optimization: Deadlock freedom constrains the design of routing algorithms. Application knowledge could further be exploited to optimize routing algorithms. APSRA could, for example, be based on other techniques than CDG. The impact of real-time requirements, programming model and communication protocol can also be studied.

Low power: The region concept could be applied with respect to power management. It would, for example, be possible to redirect packets on the region border even if the internal region is powered down. Exploration activities with APSRA may consider using low power as an objective in the procedure of breaking ASCDG cycles.

Appendix I:

Average Distance in a Mesh Network

1. Introduction

The term *average distance* here refers to the average number of routers (or hops) a message traverse in a network. It may be calculated both with respect to individual nodes as well as over all the network nodes. Average distance can be used for estimation of average *zero-load* latency, which is a statistical lower bound for average packet latency, assuming non-interfering traffic.

Average zero-load latency can serve as verification of simulation results of uniform traffic at low load conditions. Furthermore, if zero-load average latency is used in elaborate analysis of network performance, correct representation is of high importance.

This appendix discusses and proposes formulae for calculating average distances in one and two-dimensional mesh topologies. The equations assume minimal routes and uniformly random selected traffic destinations. Several books and papers provide formulae for obtaining average distance in meshes. Still, they are ambiguous and absence of complete details makes them hard to validate and use.

APPENDIX I

1.1. A Simple Example

First we give an introductory example of average distances. Consider the three node one-dimensional mesh in Figure 1. Assume that for each node, each of the other nodes has equal probability to be selected as destination.

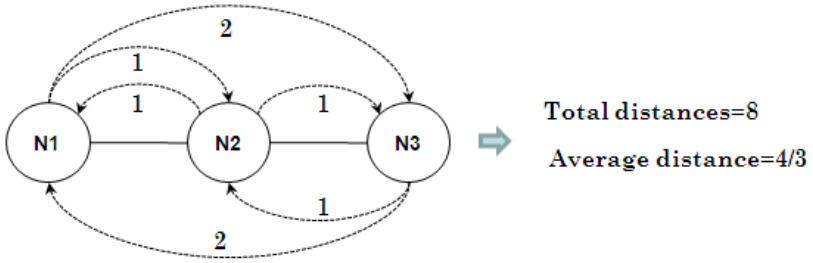


Figure 1. Distances in a three node network

The total distances for N1 is the distance to N2 plus the distance to N3 $(1 + 2)$. The number of destinations for N1 is 2, thus the average distance is $(1 + 2)/2 = 3/2$.

For N2, the total distances are $(1 + 1)$ and the average distance is $(1 + 1)/2 = 2/2$. The distances for N3 are the same as for N1. The total distances in the network is the sum of the total distances for all nodes $(3 + 2 + 3)$. The average distance in the network is the average of the distances for all node-pairs $(3 + 2 + 3)/(2 * 3) = 4/3$.

The source node often adds latency to packet paths. Therefore, another quite logical way to compute average distance would be to add the source node to the distance. Then the average distance would be $(5 + 4 + 5)/(2 * 3) = 7/3$. Note that the same result is returned by adding 1 to the average distance obtained in the previous way $(4/3 + 3/3 = 7/3)$.

1.2. Textbook Formulae

This section reviews the literature for related work on average distance calculations. The average minimum distance H_{min} in number of hops, for uniform traffic in a regular n-dimensional k-ary mesh is proposed in (Dally and Towles 2003), to be

$$H_{min} = \begin{cases} \dfrac{nk}{3}, & k\ even \\ n\left(\dfrac{k}{3} - \dfrac{1}{3k}\right), & k\ odd \end{cases}$$

Note that the formula assumes that the numbering of nodes ranges from 0 to k, which would result in k+1 number of nodes. Applying the formula to the network in Figure 1, with n=1, k=2 yields an average distance of 2/3. Assuming that there are k number of nodes, i.e. k=3, the average distance is 8/9.

Agarwal (Agarwal 1991) proposes the average distance formula in mesh to be (in the same terminology as above)

$$H_{min} = n\left(\frac{k}{3} - \frac{1}{3k}\right)$$

As can be seen, Agarwals (Agarwal 1991) formula is identical to the odd version of (Dally and Towles 2003) with an average distance of 8/9.

Liu et al. (Liu et al. 1997) determine the average distance $\overline{D}$ in an *m x n* mesh as

$$\overline{D} = \frac{1}{3}\left(m - \frac{1}{m}\right) + \frac{1}{3}\left(n - \frac{1}{n}\right)$$

In comparison with Agarwal (Agarwal 1991), the formula provided by Liu et al. (Liu et al. 1997) returns the average distance also for non-quadratic meshes, but is on the other hand restricted to one and two-dimensional meshes.

As shown in Figure 2, Liu et al. (Liu et al. 1997) provide the derivation of the formula. Note that ACD in the derivation stands for Average Communication Distance.

The average communication distance from processor $P(i, j)$ to all other nodes in the same partition of $m \times n$ processors becomes, denoted as $D(i, j)$,

$$\begin{aligned} D(i,j) &= \frac{1}{mn-1}\sum_{l=1}^{n}\sum_{k=1}^{m}(|i-k|+|j-l|) \\ &\approx \frac{1}{mn}(n\sum_{l=1}^{n}|i-k| + m\sum_{k=1}^{m}|j-l|) \\ &= \frac{1}{n}\sum_{l=1}^{n}|j-l| + \frac{1}{m}\sum_{k=1}^{m}|i-k| \end{aligned} \quad (1)$$

The ACD of such a partition, i.e. the average distance between a pair of nodes in a partition, denoted as $\overline{D}$ can be determined as

$$\begin{aligned} \overline{D} &= \frac{1}{m*n}\sum_{i=1}^{m}\sum_{j=1}^{n} D(i,j) \\ &= \frac{1}{m^2}\sum_{i=1}^{m}\sum_{k=1}^{m}|i-k| + \frac{1}{n^2}\sum_{j=1}^{n}\sum_{l=1}^{n}|j-l| \\ &= \frac{1}{3}(m - \frac{1}{m}) + \frac{1}{3}(n - \frac{1}{n}) \end{aligned} \quad (2)$$

Figure 2. Average distance formula in (Liu et al. 1997)

From the derivation, we can see that the average distance of a single node is calculated over all other nodes except itself.

Strangely, an un-explained approximation (denominator *mn*-1 to *mn*) is made such that the resulting formula is not consistent with the initial assumption. Therefore, the formula can be said to reflect the average distance when the source nodes are included with a distance of zero hops.

2. Terminology and Definitions

2.1. Definitions of Mesh Network Properties

An $R\ X\ C$ mesh network M_{RC} consists of R rows and C columns of interconnected nodes. A node $m_{rc} = (r, c)$ is indicated by its row r and column c coordinates.

In general, each node m_{rc}, $1 \le r \le R, 1 \le c \le C$, is directly connected to nodes $m_{(r-1)c}, m_{(r+1)c}$, $m_{r(c-1)}$, $m_{r(c+1)}$ as depicted in Figure 3. Boundary nodes have one or two fewer connections.

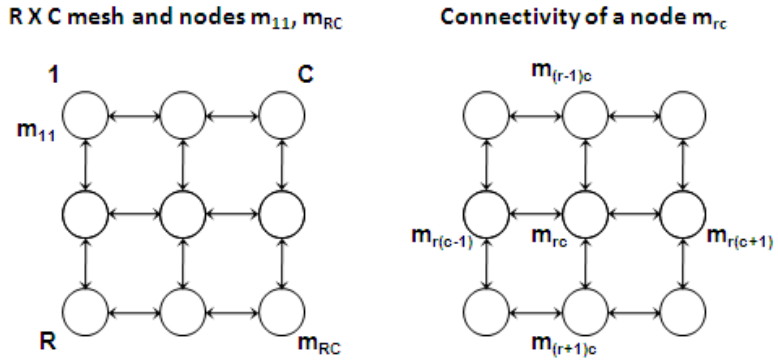


Figure 3. Properties of *R X C* mesh network

2.2. Distances related to a Single Node

The minimum distance $h_{min}(s, d)$ in number of hops from a source node $s = (r_s, c_s)$ to a destination node $d = (r_d, c_d)$, is

$$h_{min}(s, d) = |r_d - r_s| + |c_d - c_s|$$

Let D_s represent the set of all destination nodes for node s. The total of distances $H(s)$ from node s to all other nodes is then

$$H(s) = \sum_{\forall d \in D_s} h_{min}(s, d)$$

The average distance $\overline{H}(s)$ from node s to another node is

$$\overline{H}(s) = \frac{H(s)}{|D_s|}$$

2.3. Definitions of Aggregated Distances

Let S be the set of all source nodes and D_s the set of destination nodes for a node s in an M_{RC} mesh. The total aggregated distances H_{RC} is then

$$H_{RC} = \sum_{\forall s \in S} H(s)$$

The average distance $\bar{H}_{RC}$ is defined as

$$\bar{H}_{RC} = \frac{H_{RC}}{\sum_{\forall s \in S} |D_s|}$$

Given that each of the RC source nodes has RC-1 destinations, the average distance $\bar{H}_{RC}$ is equivalently

$$\bar{H}_{RC} = \frac{H_{RC}}{RC(RC-1)}$$

Note that the source node s is excluded to be its own destination.

3. Average Distance for One and Two-Dimensional Meshes

3.1. Average Distance in 1-D Mesh

This section derives the average distance for 1-D mesh, M_{1C} (R=1). The same logic will be used for obtaining the formulas for 2-D mesh. First, consider the average distance from one node to all other nodes.

Study the node m_{1C} in the four node mesh M_{1C}: $C = 4$, in Figure 4. The distance from this node to any other node is the distance of the previous node, $m_{1(C-1)}$ plus 1.

The total distance for $m_{1(C-1)}$ is 2+1 and the number of destination nodes for m_{1C} is 3, thus the total distances for m_{1C} is (2+1) + 3*1 = 6.

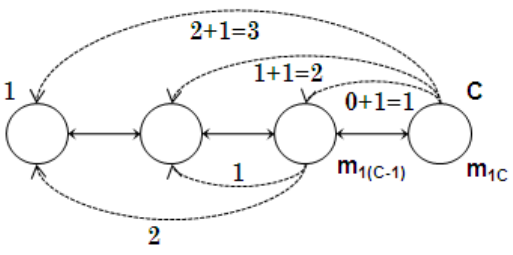


Figure 4. Total distance of node m_{1C} to all other nodes

As the number of destinations for a node m_{1C} is always C-1, the total distance $H(m_{1C})$ can be expressed as

$$H(m_{1C}) = H(m_{1(C-1)}) + C - 1$$

Solving the above recurrence relation, we obtain

APPENDIX I

$$H(m_{1C}) = \sum_{C=1}^{C} (C-1)$$

$$\Rightarrow H(m_{1C}) = \frac{C(C-1)}{2}$$

The total distance $H(m_{1C})$ is used for calculating the total distance of all nodes to all other nodes.

Note in the four-node mesh example in Figure 5, that the total distance H_{1C} of a mesh M_{1C} is always the total distances $H_{1(C-1)}$ of the one node smaller mesh, $M_{1(C-1)}$, plus the total distances $H(m_{1C})$ of the node m_{1C} to all other nodes, plus the distances (which is also $H(m_{1C})$) of all other nodes to m_{1C}.

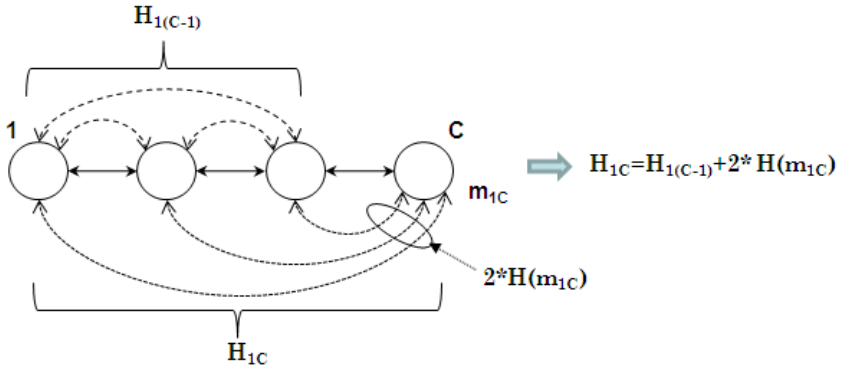


Figure 5. Total distance of node m_{1C} to all other nodes

Thus, the total distance of all nodes to all other nodes, H_{1C} can be expressed by

$$H_{1C} = H_{1(C-1)} + 2H(m_{1C})$$

Inserting the newly obtained $H(m_{1C})$ gives the equation

$$H_{1C} = H_{1(C-1)} + C(C-1)$$

Solving above recurrence relation provides the closed form expression

$$H_{1C} = \sum_{C=1}^{C} C(C-1)$$

$$= \frac{C(C+1)(2C+1)}{6} - \frac{C(C+1)}{2}$$

$$\Rightarrow H_{1C} = \frac{C(C+1)(C-1)}{3}$$

Dividing by the number of nodes-pairs gives the average distance $\bar{H}_{1C}$

$$\bar{H}_{1C} = \frac{C(C+1)(C-1)}{C(C-1)3}$$

$$\Rightarrow \bar{H}_{1C} = \frac{C+1}{3}$$

3.2. Average Distance in 2-D Mesh

Similar to the 1-D mesh case, in the case of 2-D mesh the distances from one node to all other nodes is first calculated.

Consider a two-dimensional mesh, M_{RC} depicted in Figure 6, and a node in an arbitrary row r, column C, $m_{rC} = (r, C)$.

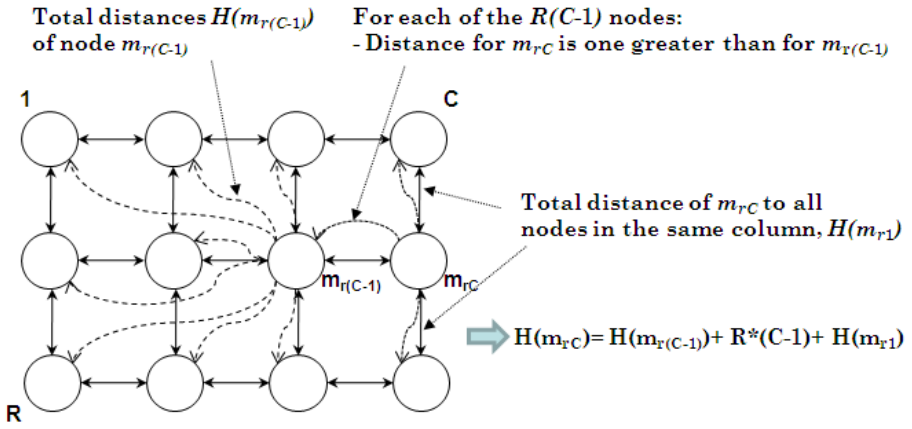


Figure 6. Total distance of node m_{rC} to all other nodes

As seen in Figure 6, the total distances $H(m_{rC})$ of node m_{rC} is $R(C-1) + H(m_{r1})$ larger than the total distances of a node from the same row in column C-1, $m_{r(C-1)}$.

The equation for total distance $H(m_{rC})$ is then

$$H(m_{rC}) = H(m_{r(C-1)}) + R(C-1) + H(m_{r1})$$

Solving the recurrence equation gives the summation expression

$$H(m_{rC}) = R\sum_{C=1}^{C}(C - 1 + H(m_{r1}))$$

$$\Rightarrow H(m_{rC}) = \frac{RC(C-1)}{2} + CH(m_{r1})$$

For all R nodes $(R(m_{rC}))$ in column C, the total distance is

$$HR(m_{rC}) = R\left(\frac{RC(C-1)}{2} + CH(m_{r1})\right)$$

APPENDIX I

The total column distance for each node $H(m_{r1})$ varies with row coordinate *r*. But, considering that the total distances of all nodes to all other nodes within a column, $RH(m_{r1})$, is H_{R1} (total distances in a 1-d mesh) we get

$$HR(m_{rC}) = \frac{RRC(C-1)}{2} + CH_{R1}$$

Similar to the 1-D mesh analysis, the total distance H_{RC} is given by the total distances of the mesh of size C-1, $H_{R(C-1)}$ plus the total distances of nodes in column C to all other nodes $HR(m_{rC})$, and the total distances from all other nodes to nodes in column C, which is also $HR(m_{rC})$.

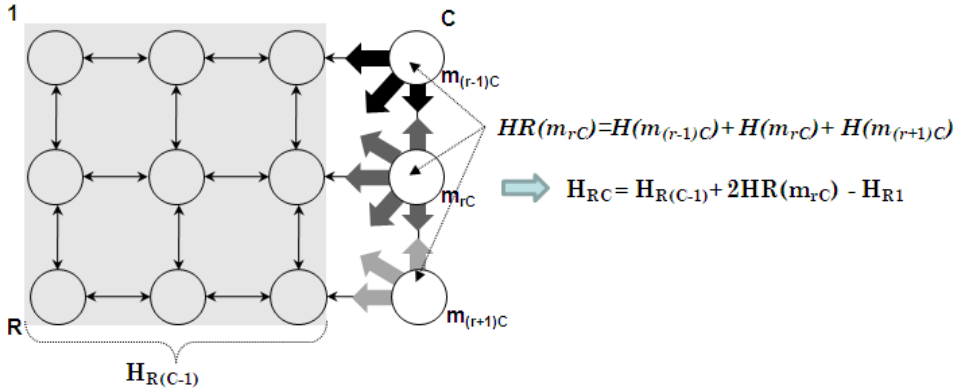


Figure 7. Derivation of total distance in 2-D mesh

Note in Figure 7 that the total distances in column C itself is counted *twice* in $2 * HR(m_{rC})$, therefore one H_{R1} needs to be subtracted. The final equation forH_{RC} is

$$\begin{aligned} H_{RC} &= H_{R(C-1)} + 2HR(m_{rC}) - H_{R1} \\ &= H_{R(C-1)} + 2\left(\frac{RRC(C-1)}{2} + CH_{R1}\right) - H_{R1} \\ \Rightarrow H_{RC} &= H_{R(C-1)} + RRC(C-1) + H_{R1}(2C-1) \end{aligned}$$

Above equation can be expressed as a summation over C columns

$$\begin{aligned} H_{RC} &= RR\sum_{C=1}^{C} C(C-1) + H_{R1}\sum_{C=1}^{C}(2C-1) \\ &= RR\frac{C(C+1)(2C+1)}{6} - \frac{C(C+1)}{2} + H_{R1}(C(C+1)-C) \\ &= RR\left(\frac{C(C+1)(2C+1)}{6} - \frac{C(C+1)}{2}\right) + \left(\frac{R(R+1)(R-1)}{3}\right)CC \\ \Rightarrow H_{RC} &= RC\left(\frac{(RC-1)(R+C)}{3}\right) \end{aligned}$$

Dividing by the number of source-destination pairs, the average distance $\overline{H}_{RC}$ is simply

$$\overline{H}_{RC} = \frac{H_{RC}}{RC(RC-1)}$$
$$\Rightarrow \overline{H}_{RC} = \frac{R+C}{3}$$

4. Discussion

As shown in Section 1.2, average distance formulae found in literature do not return consistent results. Without details on formulae derivation, it is hard to guess the reason for these discrepancies. Liu et al. (Liu et al. 1997) do provide more detailed information. From the derivation in Section 1.2, we concluded that the average distance reflects that the source node is treated as its own destination, with a distance of zero hops.

Let us compare our formula with the formula of Liu et al. (Liu et al. 1997) for a mesh of size 4 x 3. First our formula:

$$\overline{H}_{RC} = \frac{R+C}{3} = \frac{4+3}{3} = \frac{7}{3}$$

Next the Liu et al. (Liu, Lin, and Song 1997) formula:

$$\overline{D} = \frac{1}{3}\left(m - \frac{1}{m}\right) + \frac{1}{3}\left(n - \frac{1}{n}\right) = \left(\frac{16}{12} - \frac{1}{12}\right) + \left(\frac{9}{9} - \frac{1}{9}\right) = \frac{77}{36}$$

Adjusting the Liu et al. (Liu et al. 1997) result by removing the source as destination gives:

$$\frac{77}{36} * \frac{4*3}{4*3-1} = \frac{7}{3}$$

It is quite clear that our formula is simpler, but the question is: Is it the most suitable? Our answer is that it depends on the system that is modeled. If each source node is itself a member of its own set of destinations, the formula by Liu et al. (Liu et al. 1997) is most appropriate. If it is not, our formula would most correctly reflect the average distance.

Both the main as well as the intermediate results on average distance may be subject to future work. Two examples for future extensions are average distance for an arbitrary single node and average distances for higher dimensional meshes.

REFERENCES

Agarwal, A., 1991. Limits on interconnection network performance. *Parallel and Distributed Systems, IEEE Transactions on*, 2(4), 398-412.

Andreasson, D. & Kumar, S., 2005. Slack-time aware routing in NoC systems. In *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*. pp. 2353-2356 Vol. 3.

Anjan, K. & Pinkston, T., 1995. DISHA: a deadlock recovery scheme for fully adaptive routing. In *Parallel Processing Symposium, 1995. Proceedings., 9th International*. pp. 537-543.

Arns, R., 1998. The other transistor: early history of the metal-oxide semiconductor field-effect transistor. *Engineering Science and Education Journal*, 7(5), 233-240.

Ascia, G. et al., 2004. Multi-objective mapping for mesh-based NoC architectures. In *Hardware/Software Codesign and System Synthesis, 2004. CODES + ISSS 2004. International Conference on*. pp. 182-187.

Ascia, G. et al., 2006. Neighbors-on-Path: A New Selection Strategy for On-Chip Networks. In *Embedded Systems for Real Time Multimedia, Proceedings of the 2006 IEEE/ACM/IFIP Workshop on*. pp. 79-84.

Ashouei, M. et al., 2006. Statistical estimation of correlated leakage power variation and its application to leakage-aware design. In *VLSI Design, 2006. Held jointly with 5th International Conference on Embedded Systems and Design., 19th International Conference on*. p. 7 pp.

Banerjee, A. et al., 2007. A Power and Energy Exploration of Network-on-Chip Architectures. In *Proc. of the ACM/IEEE Int. Symp. on Networks-on-Chip (NOCS)*.

Banker, J. et al., 1993. Physical design tradeoffs for ASIC technologies. In *ASIC Conference and Exhibit, 1993. Proceedings., Sixth Annual IEEE International*. pp. 70-78.

Benini, L. & Bertozzi, D., 2005. Network-on-chip architectures and design methods. *Computers and Digital Techniques, IEE Proceedings* -, 152(2), 261-272.

Bertozzi, D. et al., 2005. NoC Synthesis Flow for Customized Domain Specific Mutliprocessor Systems-on-Chip. *IEEE Transactions on Parallel and Distributed Systems*, 16(2), 113–129.

Birnbaum, M. & Sachs, H., 1999. How VSIA answers the SOC dilemma. *Computer*, 32(6), 42-50.

Bolotin, E. et al., 2004. QNoC: QoS Architecture and Design Process for Network on Chip. *Journal of Systems Architecture, special issue on Network on Chip*, 50, 105–128.

Bolotin, E. et al., 2004. Automatic hardware-efficient SoC integration by QoS network on chip. In *Electronics, Circuits and Systems, 2004. ICECS 2004. Proceedings of the 2004 11th IEEE International Conference on*. pp. 479-482.

Boppana, R. & Chalasani, S., 1995. Fault-tolerant wormhole routing algorithms for mesh networks. *Computers, IEEE Transactions on*, 44(7), 848-864.

Brightwell, R. et al., 2005. Initial performance evaluation of the Cray SeaStar interconnect. In *High Performance Interconnects, 2005. Proceedings. 13th Symposium on*. pp. 51-57.

Chen, K. & Chiu, G., 1998. Fault-Tolerant Routing Algorithm for Meshes without Using Virtual Channels. *J. Inf. Sci. Eng.*, 14(4), 765-783.

Chien-Chun Su & Shin, K., 1996. Adaptive fault-tolerant deadlock-free routing in meshes and hypercubes. *Computers, IEEE Transactions on*, 45(6), 666-683.

Collins, L., 2003. Chip makers hit heat barrier. *IEE Review*, 49(1), 22-23.

Culler, D. et al., 1998. *Parallel Computer Architecture: A Hardware/Software Approach* 1st ed., Morgan Kaufmann.

Daewook, Kim. et al., 2004. CDMA-based network-on-chip architecture. In *Circuits and Systems, 2004. Proceedings. The 2004 IEEE Asia-Pacific Conference on*. pp. 137-140 vol.1.

Dally, W. & Aoki, H., 1993. Deadlock-free adaptive routing in multicomputer networks using virtual channels. *Parallel and Distributed Systems, IEEE Transactions on*, 4(4), 466-475.

Dally, W. & Seitz, C., 1987. Deadlock-Free Message Routing in Multiprocessor Interconnection Networks. *Computers, IEEE Transactions on*, C-36(5), 547-553.

Dally, W. & Towles, B., 2001. Route packets, not wires: on-chip interconnection networks. In *Design Automation Conference, 2001. Proceedings*. pp. 684-689.

Davis, J. et al., 2001. Interconnect limits on gigascale integration (GSI) in the 21st century. *Proceedings of the IEEE*, 89(3), 305-324.

Dey, S. et al., 2000. Using a soft core in a SoC design: experiences with picoJava. *Design & Test of Computers, IEEE*, 17(3), 60-71.

Di Natale, M., 2000. Scheduling the CAN bus with earliest deadline techniques. In *Real-Time Systems Symposium, 2000. Proceedings. The 21st IEEE*. pp. 259-268.

Duato, J., 1993. A new theory of deadlock-free adaptive routing in wormhole networks . *Parallel and Distributed Systems, IEEE Transactions on*, 4(12), 1320-1331.

Duato, J. et al., 1997. *Interconnection Networks: An Engineering Approach*, IEEE Computer Society Press.

Fleury, E. & Fraigniaud, P., 1998. A general theory for deadlock avoidance in wormhole-routed networks . *Parallel and Distributed Systems, IEEE Transactions on*, 9(7), 626-638.

Flich, J. et al., 2007. Region-Based Routing. An Efficient Routing Mechanism to Tackle Unreliable Hardware in Network on Chips. In *Proc. of the ACM/IEEE Int. Symp. on Networks-on-Chip (NOCS)*.

Frazzetta, D. et al., 2008. Efficient Application Specific Routing Algorithms for NoC Systems utilizing Partially Faulty Links. In *Digital System Design Architectures, Methods and Tools, 2008. DSD '08. 11th EUROMICRO Conference on*. pp. 18-25.

Gaughan, P. & Yalamanchili, S., 1993. Adaptive routing protocols for hypercube interconnection networks . *Computer*, 26(5), 12-23.

Ge-Ming Chiu, 2000. The odd-even turn model for adaptive routing. *Parallel and Distributed Systems, IEEE Transactions on*, 11(7), 729-738.

Glass, C. & Ni, L., 1992. The Turn Model for Adaptive Routing. In Computer Architecture, 1992. Proceedings., The 19th Annual International Symposium on. pp. 278-287.

Glass, C.J. & Ni, L.M., 1996. Fault-Tolerant Wormhole Routing in Meshes without Virtual Channels. *IEEE Trans. Parallel Distrib. Syst.*, 7(6), 620-636.

Goossens, K. et al., 2004. Interconnect and memory organization in SOCs for advanced set-top boxes and TV — evolution, analysis, and trends. *In Jari Nurmi, Hannu Tenhunen, Jouni Isoaho, and Axel Jantsch, editors, Interconnect-Centric Design for Advanced SoC and NoC, chapter 15*, 2004, 399--423.

Gries, M., 2004. Methods for evaluating and covering the design space during early design development. *Integration, the VLSI Journal*, 38(2), 131-183.

Guerrier, P. & Greiner, A., 2000. A generic architecture for on-chip packet-switched interconnections . In *Design, Automation and Test in Europe Conference and Exhibition 2000. Proceedings*. pp. 250-256.

Havemann, R. & Hutchby, J., 2001. High-performance interconnects: an integration overview. *Proceedings of the IEEE*, 89(5), 586-601.

Hemani, A. et al., 2000. Network on a Chip: An architecture for billion transistor era. *Proc. of IEEE NorChip Conference*.

Henkel, J., 2003. Closing the SoC design gap. *Computer*, 36(9), 119-121.

Henkel, J. et al., 2004. On-chip networks: a scalable, communication-centric embedded system design paradigm. In *VLSI Design, 2004. Proceedings. 17th International Conference on*. pp. 845-851.

Hluchyj, M. & Karol, M., 1988. Queueing in high-performance packet switching. *Selected Areas in Communications, IEEE Journal on*, 6(9), 1587-1597.

Hollstein, T. et al., 2006. Hinoc: A Hierarchical Generic Approach for on-Chip Communication, Testing and Debugging of SoCs. In *VLSI-SOC: From Systems to Chips*. pp. 39-54.

Holsmark, R. & Kumar, S., 2007. Corrections to Chen and Chiu's Fault Tolerant Routing Algorithm for Mesh Networks. *Journal of Information Science and Engineering*, 23(6), 1649-1662.

Holsmark, R. & Kumar, S., 2005. Design Issues and Performance Evaluation of Mesh NoC with Regions. In *NORCHIP Conference, 2005. 23rd*. pp. 1-4.

Holsmark, R. et al., 2006. Deadlock Free Routing Algorithms for Mesh Topology NoC Systems with Regions. In *Proc. 9th EUROMICRO Conference on Digital System Design, Architectures, Methods and Tools (DSD)*.

Holsmark, R. et al., 2008. Deadlock free routing algorithms for irregular mesh topology NoC systems with rectangular regions. *Journal of Systems Architecture*, 54(3-4), 427 - 440.

Ishiwata, S. et al., 2003. A single-chip MPEG-2 codec based on customizable media embedded processor. *Solid-State Circuits, IEEE Journal of*, 38(3), 530-540.

Jayasimha, D. et al., 1996. A foundation for designing deadlock-free routing algorithms in wormhole networks. In *Parallel and Distributed Processing, 1996. Eighth IEEE Symposium on*. pp. 190-197.

Je Wu & Zhen Jiang, 2002. Extended minimal routing in 2D meshes with faulty blocks. In *Distributed Computing Systems Workshops, 2002. Proceedings. 22nd International Conference on*. pp. 49-54.

Jiang Xie & Chin-Tau Lea, 1999. Speedup and buffer division in input/output queueing ATM switches . In *Global Telecommunications Conference, 1999. GLOBECOM '99*. pp. 49-53 vol.1a.

Jie Wu, 2003. A fault-tolerant and deadlock-free routing protocol in 2D meshes based on odd-even turn model. *Computers, IEEE Transactions on*, 52(9), 1154-1169.

Jingcao Hu & Marculescu, R., 2004. DyAD - smart routing for networks-on-chip. In *Design Automation Conference, 2004. Proceedings. 41st*. pp. 260-263.

Jingcao Hu & Marculescu, R., 2005. Energy- and performance-aware mapping for regular NoC architectures. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 24(4), 551-562.

Jipeng Zhou & Lau, F., 2000. Fault-tolerant wormhole routing in 2D meshes. In Parallel Architectures, Algorithms and Networks, 2000. I-SPAN 2000. Proceedings. International Symposium on. pp. 94-101.

Keutzer, K. et al., 2000. System-level design: Orthogonalization of concerns and platform-based design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19, 1523--1543.

Kihong Park, Gitae Kim & Crovella, M., 1996. On the relationship between file sizes, transport protocols, and self-similar network traffic. In *Network Protocols, 1996. Proceedings., 1996 International Conference on*. pp. 171-180.

Kilby, J., 2000. The integrated circuit's early history. *Proceedings of the IEEE*, 88(1), 109-111.

Koibuchi, M. et al., 2001. L-turn routing: an adaptive routing in irregular networks. In *Parallel Processing, International Conference on, 2001.* pp. 383-392.

Kucukcakar, K., 1998. Analysis of emerging core-based design lifecycle. In Computer-Aided Design, 1998. ICCAD 98. Digest of Technical Papers. 1998 IEEE/ACM International Conference on. pp. 445-449.

Kumar, S. et al., 2002. A network on chip architecture and design methodology. In *VLSI, 2002. Proceedings. IEEE Computer Society Annual Symposium on*. pp. 105-112.

Kyeong Keol Ryu, Eung Shin & Mooney, V., 2001. A comparison of five different multiprocessor SoC bus architectures . In *Digital Systems, Design, 2001. Proceedings. Euromicro Symposium on*. pp. 202-209.

Lahiri, K. et al., 2001. System-level performance analysis for designing on-chip communication architectures. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 20(6), 768-783.

Levin, P. & Ludwig, R., 2002. Crossroads for mixed-signal chips. *Spectrum, IEEE*, 39(3), 38-43.

Liu, H. et al., 1997. An efficient processor partitioning and thread mapping strategy for mesh-connected multiprocessor systems. In *Proceedings of the 1997 ACM symposium on Applied computing*, 403-412. San Jose, California, United States.

Loghi, M. et al., 2004. Analyzing on-chip communication in a MPSoC environment. In *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*. pp. 752-757 Vol.2.

Man Lung Mui, Banerjee, K. & Mehrotra, A., 2004. A global interconnect optimization scheme for nanometer scale VLSI with implications for latency, bandwidth, and power dissipation. *Electron Devices, IEEE Transactions on*, 51(2), 195-203.

Manolache, S. et al., 2006. Buffer space optimisation with communication synthesis and traffic shaping for NoCs. In *Proceedings of the conference on Design, automation and test in Europe: Proceedings*. Munich, Germany: European Design and Automation Association, pp. 718-723.

McKeown, N., 1999. The iSLIP scheduling algorithm for input-queued switches. *Networking, IEEE/ACM Transactions on*, 7(2), 188-201.

Mead, C. & Conway, L., 1979. *Introduction to Vlsi Systems*, Addison-Wesley Pub (Sd).

Mejia, A. et al., 2006. Segment-Based Routing: An Efficient Fault-Tolerant Routing Algorithm for Meshes and Tori. In *Parallel and Distributed Processing Symposium, April*.

Millberg, M. et al., 2004. Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network on chip. In *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*. pp. 890-895 Vol.2.

Mohapatra, P., 1998. Wormhole routing techniques for directly connected multicomputer systems. *ACM Comput. Surv.*, 30(3), 374-410.

Moore, G., 1975. Progress in digital integrated electronics. In *Electron Devices Meeting, 1975 International*. pp. 11-13.

Moore, G., 1965. Cramming More Components onto Integrated Circuits. *Electronics*, 38(8), 117, 114.

Murali, S. et al., 2006. A multi-path routing strategy with guaranteed in-order packet delivery and fault-tolerance for networks on chip. In *Proceedings of the 43rd annual conference on Design automation*. San Francisco, CA, USA: ACM, pp. 845-848.

Neeb, C. & Wehn, N., 2006. Designing Efficient Irregular Networks for Heterogeneous Systems-on-Chip. In *Digital System Design: Architectures, Methods and Tools, 2006. DSD 2006. 9th EUROMICRO Conference on*. pp. 665-672.

Ni, L. & McKinley, P., 1993. A survey of wormhole routing techniques in direct networks. *Computer*, 26(2), 62-76.

Nurmi, J. et al., 2005. Interconnect-Centric Design for Advanced SoC and NoC.

Ogras, U. & Marculescu, R., 2006. "It's a small world after all": NoC performance optimization via long-range link insertion. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 14(7), 693-706.

Ogras, U.Y. & Marculescu, R., 2007. Analytical Router Modeling for Networks-on-Chip Performance Analysis. In *Proc. Design, Automation and Test in Europe (DATE) Conf.* Nice, France.

Palesi, M., Holsmark, R. & Kumar, S., 2006. A methodology for design of application specific deadlock-free routing algorithms for NoC systems. In *Proceedings of the 4th international conference on Hardware/software codesign and system synthesis*. ACM New York, NY, USA, pp. 142-147.

Palesi, M. et al., 2006. A Method for Router Table Compression for Application Specific Routing in Mesh Topology NoC Architectures. In *Proc. of the SAMOS VI Workshop: Embedded Computer Systems: Architectures, Modeling, and Simulation*.

Palesi, M. et al., 2007. Exploiting Communication Concurrency for Efficient Deadlock Free Routing in Reconfigurable NoC Platforms. In *14th Reconfigurable Architectures Workshop March*. pp. 27-28.

Palesi, M. et al., 2008. Design of Bandwidth Aware and Congestion Avoiding Efficient Routing Algorithms for Networks-on-Chip Platforms. In *Networks-on-Chip, 2008. NoCS 2008. Second ACM/IEEE International Symposium on*. pp. 97-106.

Palesi, M. et al., 2009. Application Specific Routing Algorithms for Networks on Chip. *Parallel and Distributed Systems, IEEE Transactions on*, 20(3), 316-330.

Pande, P. et al., 2003. Design of a switch for network on chip applications. In *Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on*. pp. V-217-V-220 vol.5.

Pasricha, S. et al., 2008. Fast exploration of bus-based communication architectures at the CCATB abstraction. *Trans. on Embedded Computing Sys.*, 7(2), 1-32.

Qian, Y. et al., 2009. Analysis of communication delay bounds for network on chips. In *Proceedings of the 2009 Asia and South Pacific Design Automation Conference*. Yokohama, Japan: IEEE Press, pp. 7-12.

Rabaey, J. et al., 2006. Embedding Mixed-Signal Design in Systems-on-Chip. *Proceedings of the IEEE*, 94(6), 1070-1088.

Rabaey, J., 2005. System-on-Chip-Challenges in the Deep-Sub-Micron Era. In *Interconnect-Centric Design for Advanced SoC and NoC*. pp. 3-24.

Riordan, M., 2004. The lost history of the transistor. *Spectrum, IEEE*, 41(5), 44-49.

Saastamoinen, I. et al., 2002. Interconnect IP node for future system-on-chip designs. In *Electronic Design, Test and Applications, 2002. Proceedings. The First IEEE International Workshop on*. pp. 116-120.

Sancho, J. et al., 2004. An effective methodology to improve the performance of the up*/down* routing algorithm. *Parallel and Distributed Systems, IEEE Transactions on*, 15(8), 740-754.

Schroeder, M. et al., 1991. Autonet: a high-speed, self-configuring local area network using point-to-point links. *Selected Areas in Communications, IEEE Journal on*, 9(8), 1318-1335.

Schwiebert, L. & Jayasimha, D., 1993. Optimal fully adaptive wormhole routing for meshes. In *Supercomputing '93. Proceedings*. pp. 782-791.

Silla, F. et al., 1998. Virtual channel multiplexing in networks of workstations with irregular topology. In *High Performance Computing, 1998. HIPC '98. 5th International Conference On*. pp. 147-154.

Singh, A. et al., 2004. Globally Adaptive Load-Balanced Routing on Tori. *Computer Architecture Letters*, 3(1), 2.

Spangenberg, K.R., 1948. *Vacuum Tubes*, McGraw-Hill.

Srinivasan, K. & Chatha, K.S., 2006. A low complexity heuristic for design of custom network-on-chip architectures. In *In Proc. Design, Automation and Test in Europe (DATE)*.

Srinivasan, K. et al., 2006. Linear-programming-based techniques for synthesis of network-on-chip architectures. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 14(4), 407-420.

Sun, Y. et al., 2002. Simulation and Evaluation of a Network on Chip Architecture Using Ns-2. In *Proceedings of the IEEE NorChip Conference*.

Sung-Rok Yoon, Jin Lee & Sin-Chong Park, 2006. Case Study : NoC based Next-generation WLAN receiver design in Transaction Level. In *Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference*. pp. 1125-1128.

Sylvester, D. & Keutzer, K., 1998. Getting to the bottom of deep submicron. In Computer-Aided Design, 1998. ICCAD 98. Digest of Technical Papers. 1998 IEEE/ACM International Conference on. pp. 203-211.

Thid, R. et al., 2003. Evaluating NoC communication backbones with simulation. In *Proceedings of the IEEE NorChip Conference*.

Tredennick, N., 1996. Microprocessor-based computers. *Computer*, 29(10), 27-37.

Turley, J., 2002. Embedded.com - The Two Percent Solution. [Accessed August 7, 2009].

Vaidya, A. et al., 1999. LAPSES: a recipe for high performance adaptive router design. In *High-Performance Computer Architecture, 1999. Proceedings. Fifth International Symposium On*. pp. 236-243.

Yu Cao et al., 2003. Yield optimization with energy-delay constraints in low-power digital circuits. In *Electron Devices and Solid-State Circuits, 2003 IEEE Conference on*. pp. 285-288.

LINKÖPINGS UNIVERSITET

Avdelning, institution
Division, department

Institutionen för datavetenskap

Department of Computer and Information Science

Datum
Date

2009-09-14

Språk
Language

[ ] Svenska/Swedish
[X] Engelska/English

[ ] _____

Rapporttyp
Report category

[X] Licentiatavhandling
[ ] Examensarbete
[ ] C-uppsats
[ ] D-uppsats
[ ] Övrig rapport

_____

URL för elektronisk version

ISBN 978-91-7393-559-3

ISRN LiU-Tek-Lic-2009:18

Serietitel och serienummer
Title of series, numbering

ISSN 0280-7971

Linköping Studies in Science and Technology

Thesis No. 1410

Titel
Deadlock Free Routing in Mesh Networks on Chip with Regions

Författare
Rickard Holsmark

Sammanfattning
Abstract

There is a seemingly endless miniaturization of electronic components, which has enabled designers to build sophisticated computing structures on silicon chips. Consequently, electronic systems are continuously improving with new and more advanced functionalities. Design complexity of these Systems on Chip (SoC) is reduced by the use of pre-designed *cores*. However, several problems related to the interconnection of cores remain. Network on Chip (NoC) is a new SoC design paradigm, which targets the interconnect problems using classical network concepts. Still, SoC cores show large variance in size and functionality, whereas several NoC benefits relate to regularity and homogeneity.

This thesis studies some network aspects which are characteristic to NoC systems. One is the issue of area wastage in NoC due to cores of various sizes. We elaborate on using oversized *regions* in regular mesh NoC and identify several new design possibilities. Adverse effects of regions on communication are outlined and evaluated by simulation.

Deadlock freedom is an important region issue, since it affects both the usability and performance of routing algorithms. The concept of faulty blocks, used in deadlock free fault-tolerant routing algorithms has similarities with rectangular regions. We have improved and adopted one such algorithm to provide deadlock free routing in NoC with regions. This work also offers a methodology for designing topology agnostic, deadlock free, highly adaptive *application specific* routing algorithms. The methodology exploits information about communication among tasks of an application. This is used in the analysis of deadlock freedom, such that fewer deadlock preventing routing restrictions are required.

A comparative study of the two proposed routing algorithms shows that the application specific algorithm gives significantly higher performance. But, the fault-tolerant algorithm may be preferred for systems requiring support for general communication. Several extensions to our work are proposed, for example in areas such as core mapping and efficient routing algorithms. The region concept can be extended for supporting reuse of a pre-designed NoC as a component in a larger hierarchical NoC.

Nyckelord
Keywords
Networks on Chip, Mesh Topology, Routing Algorithms, Wormhole Switching, Deadlock, Application Specific Routing

Department of Computer and Information Science
Linköpings universitet

Linköping Studies in Science and Technology
Faculty of Arts and Sciences - Licentiate Theses

No 17 **Vojin Plavsic:** Interleaved Processing of Non-Numerical Data Stored on a Cyclic Memory. (Available at: FOA, Box 1165, S-581 11 Linköping, Sweden. FOA Report B30062E)

No 28 **Arne Jönsson, Mikael Patel:** An Interactive Flowcharting Technique for Communicating and Realizing Algorithms, 1984.

No 29 **Johnny Eckerland:** Retargeting of an Incremental Code Generator, 1984.

No 48 **Henrik Nordin:** On the Use of Typical Cases for Knowledge-Based Consultation and Teaching, 1985.

No 52 **Zebo Peng:** Steps Towards the Formalization of Designing VLSI Systems, 1985.

No 60 **Johan Fagerström:** Simulation and Evaluation of Architecture based on Asynchronous Processes, 1985.

No 71 **Jalal Maleki:** ICONStraint, A Dependency Directed Constraint Maintenance System, 1987.

No 72 **Tony Larsson:** On the Specification and Verification of VLSI Systems, 1986.

No 73 **Ola Strömfors:** A Structure Editor for Documents and Programs, 1986.

No 74 **Christos Levcopoulos:** New Results about the Approximation Behavior of the Greedy Triangulation, 1986.

No 104 **Shamsul I. Chowdhury: S**tatistical Expert Systems - a Special Application Area for Knowledge-Based Computer Methodology, 1987.

No 108 **Rober Bilos:** Incremental Scanning and Token-Based Editing, 1987.

No 111 **Hans Block:** SPORT-SORT Sorting Algorithms and Sport Tournaments, 1987.

No 113 **Ralph Rönnquist:** Network and Lattice Based Approaches to the Representation of Knowledge, 1987.

No 118 **Mariam Kamkar, Nahid Shahmehri:** Affect-Chaining in Program Flow Analysis Applied to Queries of Programs, 1987.

No 126 **Dan Strömberg:** Transfer and Distribution of Application Programs, 1987.

No 127 **Kristian Sandahl:** Case Studies in Knowledge Acquisition, Migration and User Acceptance of Expert Systems, 1987.

No 139 **Christer Bäckström:** Reasoning about Interdependent Actions, 1988.

No 140 **Mats Wirén:** On Control Strategies and Incrementality in Unification-Based Chart Parsing, 1988.

No 146 **Johan Hultman:** A Software System for Defining and Controlling Actions in a Mechanical System, 1988.

No 150 **Tim Hansen:** Diagnosing Faults using Knowledge about Malfunctioning Behavior, 1988.

No 165 **Jonas Löwgren:** Supporting Design and Management of Expert System User Interfaces, 1989.

No 166 **Ola Petersson:** On Adaptive Sorting in Sequential and Parallel Models, 1989.

No 174 **Yngve Larsson:** Dynamic Configuration in a Distributed Environment, 1989.

No 177 **Peter Åberg:** Design of a Multiple View Presentation and Interaction Manager, 1989.

No 181 **Henrik Eriksson:** A Study in Domain-Oriented Tool Support for Knowledge Acquisition, 1989.

No 184 **Ivan Rankin:** The Deep Generation of Text in Expert Critiquing Systems, 1989.

No 187 **Simin Nadjm-Tehrani:** Contributions to the Declarative Approach to Debugging Prolog Programs, 1989.

No 189 **Magnus Merkel:** Temporal Information in Natural Language, 1989.

No 196 **Ulf Nilsson:** A Systematic Approach to Abstract Interpretation of Logic Programs, 1989.

No 197 **Staffan Bonnier:** Horn Clause Logic with External Procedures: Towards a Theoretical Framework, 1989.

No 203 **Christer Hansson:** A Prototype System for Logical Reasoning about Time and Action, 1990.

No 212 **Björn Fjellborg:** An Approach to Extraction of Pipeline Structures for VLSI High-Level Synthesis, 1990.

No 230 **Patrick Doherty:** A Three-Valued Approach to Non-Monotonic Reasoning, 1990.

No 237 **Tomas Sokolnicki:** Coaching Partial Plans: An Approach to Knowledge-Based Tutoring, 1990.

No 250 **Lars Strömberg:** Postmortem Debugging of Distributed Systems, 1990.

No 253 **Torbjörn Näslund:** SLDFA-Resolution - Computing Answers for Negative Queries, 1990.

No 260 **Peter D. Holmes:** Using Connectivity Graphs to Support Map-Related Reasoning, 1991.

No 283 **Olof Johansson:** Improving Implementation of Graphical User Interfaces for Object-Oriented Knowledge-Bases, 1991.

No 298 **Rolf G Larsson:** Aktivitetsbaserad kalkylering i ett nytt ekonomisystem, 1991.

No 318 **Lena Strömbäck:** Studies in Extended Unification-Based Formalism for Linguistic Description: An Algorithm for Feature Structures with Disjunction and a Proposal for Flexible Systems, 1992.

No 319 **Mikael Pettersson:** DML-A Language and System for the Generation of Efficient Compilers from Denotational Specification, 1992.

No 326 **Andreas Kågedal:** Logic Programming with External Procedures: an Implementation, 1992.

No 328 **Patrick Lambrix:** Aspects of Version Management of Composite Objects, 1992.

No 333 **Xinli Gu:** Testability Analysis and Improvement in High-Level Synthesis Systems, 1992.

No 335 **Torbjörn Näslund:** On the Role of Evaluations in Iterative Development of Managerial Support Sytems, 1992.

No 348 **Ulf Cederling:** Industrial Software Development - a Case Study, 1992.

No 352 **Magnus Morin:** Predictable Cyclic Computations in Autonomous Systems: A Computational Model and Implementation, 1992.

No 371 **Mehran Noghabai:** Evaluation of Strategic Investments in Information Technology, 1993.

No 378 **Mats Larsson:** A Transformational Approach to Formal Digital System Design, 1993.

No 380 **Johan Ringström:** Compiler Generation for Parallel Languages from Denotational Specifications, 1993.

No 381 **Michael Jansson:** Propagation of Change in an Intelligent Information System, 1993.

No 383 **Jonni Harrius:** An Architecture and a Knowledge Representation Model for Expert Critiquing Systems, 1993.

No 386 **Per Österling:** Symbolic Modelling of the Dynamic Environments of Autonomous Agents, 1993.

No 398 **Johan Boye:** Dependency-based Groudness Analysis of Functional Logic Programs, 1993.

No 402 **Lars Degerstedt:** Tabulated Resolution for Well Founded Semantics, 1993.
No 406 **Anna Moberg:** Satellitkontor - en studie av kommunikationsmönster vid arbete på distans, 1993.
No 414 **Peter Carlsson:** Separation av företagsledning och finansiering - fallstudier av företagsledarutköp ur ett agentteoretiskt perspektiv, 1994.
No 417 **Camilla Sjöström:** Revision och lagreglering - ett historiskt perspektiv, 1994.
No 436 **Cecilia Sjöberg:** Voices in Design: Argumentation in Participatory Development, 1994.
No 437 **Lars Viklund:** Contributions to a High-level Programming Environment for a Scientific Computing, 1994.
No 440 **Peter Loborg:** Error Recovery Support in Manufacturing Control Systems, 1994.
FHS 3/94 **Owen Eriksson:** Informationssystem med verksamhetskvalitet - utvärdering baserat på ett verksamhetsinriktat och samskapande perspektiv, 1994.
FHS 4/94 **Karin Pettersson:** Informationssystemstrukturering, ansvarsfördelning och användarinflytande - En komparativ studie med utgångspunkt i två informationssystemstrategier, 1994.
No 441 **Lars Poignant:** Informationsteknologi och företagsetablering - Effekter på produktivitet och region, 1994.
No 446 **Gustav Fahl:** Object Views of Relational Data in Multidatabase Systems, 1994.
No 450 **Henrik Nilsson:** A Declarative Approach to Debugging for Lazy Functional Languages, 1994.
No 451 **Jonas Lind:** Creditor - Firm Relations: an Interdisciplinary Analysis, 1994.
No 452 **Martin Sköld:** Active Rules based on Object Relational Queries - Efficient Change Monitoring Techniques, 1994.
No 455 **Pär Carlshamre:** A Collaborative Approach to Usability Engineering: Technical Communicators and System Developers in Usability-Oriented Systems Development, 1994.
FHS 5/94 **Stefan Cronholm:** Varför CASE-verktyg i systemutveckling? - En motiv- och konsekvensstudie avseende arbetssätt och arbetsformer, 1994.
No 462 **Mikael Lindvall:** A Study of Traceability in Object-Oriented Systems Development, 1994.
No 463 **Fredrik Nilsson:** Strategi och ekonomisk styrning - En studie av Sandviks förvärv av Bahco Verktyg, 1994.
No 464 **Hans Olsén:** Collage Induction: Proving Properties of Logic Programs by Program Synthesis, 1994.
No 469 **Lars Karlsson:** Specification and Synthesis of Plans Using the Features and Fluents Framework, 1995.
No 473 **Ulf Söderman:** On Conceptual Modelling of Mode Switching Systems, 1995.
No 475 **Choong-ho Yi:** Reasoning about Concurrent Actions in the Trajectory Semantics, 1995.
No 476 **Bo Lagerström:** Successiv resultatavräkning av pågående arbeten. - Fallstudier i tre byggföretag, 1995.
No 478 **Peter Jonsson:** Complexity of State-Variable Planning under Structural Restrictions, 1995.
FHS 7/95 **Anders Avdic:** Arbetsintegrerad systemutveckling med kalkylkprogram, 1995.
No 482 **Eva L Ragnemalm:** Towards Student Modelling through Collaborative Dialogue with a Learning Companion, 1995.
No 488 **Eva Toller:** Contributions to Parallel Multiparadigm Languages: Combining Object-Oriented and Rule-Based Programming, 1995.
No 489 **Erik Stoy:** A Petri Net Based Unified Representation for Hardware/Software Co-Design, 1995.
No 497 **Johan Herber:** Environment Support for Building Structured Mathematical Models, 1995.
No 498 **Stefan Svenberg:** Structure-Driven Derivation of Inter-Lingual Functor-Argument Trees for Multi-Lingual Generation, 1995.
No 503 **Hee-Cheol Kim:** Prediction and Postdiction under Uncertainty, 1995.
FHS 8/95 **Dan Fristedt:** Metoder i användning - mot förbättring av systemutveckling genom situationell metodkunskap och metodanalys, 1995.
FHS 9/95 **Malin Bergvall:** Systemförvaltning i praktiken - en kvalitativ studie avseende centrala begrepp, aktiviteter och ansvarsroller, 1995.
No 513 **Joachim Karlsson:** Towards a Strategy for Software Requirements Selection, 1995.
No 517 **Jakob Axelsson:** Schedulability-Driven Partitioning of Heterogeneous Real-Time Systems, 1995.
No 518 **Göran Forslund:** Toward Cooperative Advice-Giving Systems: The Expert Systems Experience, 1995.
No 522 **Jörgen Andersson:** Bilder av småföretagares ekonomistyrning, 1995.
No 538 **Staffan Flodin:** Efficient Management of Object-Oriented Queries with Late Binding, 1996.
No 545 **Vadim Engelson:** An Approach to Automatic Construction of Graphical User Interfaces for Applications in Scientific Computing, 1996.
No 546 **Magnus Werner :** Multidatabase Integration using Polymorphic Queries and Views, 1996.
FiF-a 1/96 **Mikael Lind:** Affärsprocessinriktad förändringsanalys - utveckling och tillämpning av synsätt och metod, 1996.
No 549 **Jonas Hallberg:** High-Level Synthesis under Local Timing Constraints, 1996.
No 550 **Kristina Larsen:** Förutsättningar och begränsningar för arbete på distans - erfarenheter från fyra svenska företag. 1996.
No 557 **Mikael Johansson:** Quality Functions for Requirements Engineering Methods, 1996.
No 558 **Patrik Nordling:** The Simulation of Rolling Bearing Dynamics on Parallel Computers, 1996.
No 561 **Anders Ekman:** Exploration of Polygonal Environments, 1996.
No 563 **Niclas Andersson:** Compilation of Mathematical Models to Parallel Code, 1996.
No 567 **Johan Jenvald:** Simulation and Data Collection in Battle Training, 1996.
No 575 **Niclas Ohlsson:** Software Quality Engineering by Early Identification of Fault-Prone Modules, 1996.
No 576 **Mikael Ericsson:** Commenting Systems as Design Support—A Wizard-of-Oz Study, 1996.
No 587 **Jörgen Lindström:** Chefers användning av kommunikationsteknik, 1996.
No 589 **Esa Falkenroth:** Data Management in Control Applications - A Proposal Based on Active Database Systems, 1996.
No 591 **Niclas Wahllöf:** A Default Extension to Description Logics and its Applications, 1996.
No 595 **Annika Larsson:** Ekonomisk Styrning och Organisatorisk Passion - ett interaktivt perspektiv, 1997.
No 597 **Ling Lin:** A Value-based Indexing Technique for Time Sequences, 1997.

No 598	**Rego Granlund:** C^3Fire - A Microworld Supporting Emergency Management Training, 1997.
No 599	**Peter Ingels:** A Robust Text Processing Technique Applied to Lexical Error Recovery, 1997.
No 607	**Per-Arne Persson:** Toward a Grounded Theory for Support of Command and Control in Military Coalitions, 1997.
No 609	**Jonas S Karlsson:** A Scalable Data Structure for a Parallel Data Server, 1997.
FiF-a 4	**Carita Åbom:** Videomötesteknik i olika affärssituationer - möjligheter och hinder, 1997.
FiF-a 6	**Tommy Wedlund**: Att skapa en företagsanpassad systemutvecklingsmodell - genom rekonstruktion, värdering och vidareutveckling i T50-bolag inom ABB, 1997.
No 615	**Silvia Coradeschi**: A Decision-Mechanism for Reactive and Coordinated Agents, 1997.
No 623	**Jan Ollinen:** Det flexibla kontorets utveckling på Digital - Ett stöd för multiflex? 1997.
No 626	**David Byers:** Towards Estimating Software Testability Using Static Analysis, 1997.
No 627	**Fredrik Eklund:** Declarative Error Diagnosis of GAPLog Programs, 1997.
No 629	**Gunilla Ivefors:** Krigsspel coh Informationsteknik inför en oförutsägbar framtid, 1997**.**
No 631	**Jens-Olof Lindh:** Analysing Traffic Safety from a Case-Based Reasoning Perspective, 1997
No 639	**Jukka Mäki-Turja:**. Smalltalk - a suitable Real-Time Language, 1997.
No 640	**Juha Takkinen**: CAFE: Towards a Conceptual Model for Information Management in Electronic Mail, 1997.
No 643	**Man Lin**: Formal Analysis of Reactive Rule-based Programs, 1997.
No 653	**Mats Gustafsson**: Bringing Role-Based Access Control to Distributed Systems, 1997.
FiF-a 13	**Boris Karlsson:** Metodanalys för förståelse och utveckling av systemutvecklingsverksamhet. Analys och värdering av systemutvecklingsmodeller och dess användning, 1997.
No 674	**Marcus Bjäreland:** Two Aspects of Automating Logics of Action and Change - Regression and Tractability, 1998.
No 676	**Jan Håkegård**: Hiera rchical Test Architecture and Board-Level Test Controller Synthesis, 1998.
No 668	**Per-Ove Zetterlund**: Normering av svensk redovisning - En studie av tillkomsten av Redovisningsrådets rekommendation om koncernredovisning (RR01:91), 1998.
No 675	**Jimmy Tjäder**: Projektledaren & planen - en studie av projektledning i tre installations- och systemutvecklingsprojekt, 1998.
FiF-a 14	**Ulf Melin**: Informationssystem vid ökad affärs- och processorientering - egenskaper, strategier och utveckling, 1998.
No 695	**Tim Heyer**: COMPASS: Introduction of Formal Methods in Code Development and Inspection, 1998.
No 700	**Patrik Hägglund:** Programming Languages for Computer Algebra, 1998.
FiF-a 16	**Marie-Therese Christiansson:** Inter-organistorisk verksamhetsutveckling - metoder som stöd vid utveckling av partnerskap och informationssystem, 1998.
No 712	**Christina Wennestam:** Information om immateriella resurser. Investeringar i forskning och utveckling samt i personal inom skogsindustrin, 1998.
No 719	**Joakim Gustafsson:** Extending Temporal Action Logic for Ramification and Concurrency, 1998.
No 723	**Henrik André-Jönsson:** Indexing time-series data using text indexing methods, 1999.
No 725	**Erik Larsson:** High-Level Testability Analysis and Enhancement Techniques, 1998.
No 730	**Carl-Johan Westin:** Informationsförsörjning: en fråga om ansvar - aktiviteter och uppdrag i fem stora svenska organisationers operativa informationsförsörjning, 1998.
No 731	**Åse Jansson:** Miljöhänsyn - en del i företags styrning, 1998.
No 733	**Thomas Padron-McCarthy:** Performance-Polymorphic Declarative Queries, 1998.
No 734	**Anders Bäckström:** Värdeskapande kreditgivning - Kreditriskhantering ur ett agentteoretiskt perspektiv, 1998.
FiF-a 21	**Ulf Seigerroth:** Integration av förändringsmetoder - en modell för välgrundad metodintegration, 1999.
FiF-a 22	**Fredrik Öberg:** Object-Oriented Frameworks - A New Strategy for Case Tool Development, 1998.
No 737	**Jonas Mellin:** Predictable Event Monitoring, 1998.
No 738	**Joakim Eriksson:** Specifying and Managing Rules in an Active Real-Time Database System, 1998.
FiF-a 25	**Bengt E W Andersson:** Samverkande informationssystem mellan aktörer i offentliga åtaganden - En teori om aktörsarenor i samverkan om utbyte av information, 1998.
No 742	**Pawel Pietrzak:** Static Incorrectness Diagnosis of CLP (FD), 1999.
No 748	**Tobias Ritzau:** Real-Time Reference Counting in RT-Java, 1999.
No 751	**Anders Ferntoft:** Elektronisk affärskommunikation - kontaktkostnader och kontaktprocesser mellan kunder och leverantörer på producentmarknader,1999.
No 752	**Jo Skåmedal:** Arbete på distans och arbetsformens påverkan på resor och resmönster, 1999.
No 753	**Johan Alvehus:** Mötets metaforer. En studie av berättelser om möten, 1999.
No 754	**Magnus Lindahl:** Bankens villkor i låneavtal vid kreditgivning till högt belånade företagsförvärv: En studie ur ett agentteoretiskt perspektiv, 2000.
No 766	**Martin V. Howard:** Designing dynamic visualizations of temporal data, 1999.
No 769	**Jesper Andersson:** Towards Reactive Software Architectures, 1999.
No 775	**Anders Henriksson:** Unique kernel diagnosis, 1999.
FiF-a 30	**Pär J. Ågerfalk:** Pragmatization of Information Systems - A Theoretical and Methodological Outline, 1999.
No 787	**Charlotte Björkegren:** Learning for the next project - Bearers and barriers in knowledge transfer within an organisation, 1999.
No 788	**Håkan Nilsson:** Informationsteknik som drivkraft i granskningsprocessen - En studie av fyra revisionsbyråer, 2000.
No 790	**Erik Berglund:** Use-Oriented Documentation in Software Development, 1999.
No 791	**Klas Gäre:** Verksamhetsförändringar i samband med IS-införande, 1999.
No 800	**Anders Subotic:** Software Quality Inspection, 1999.
No 807	**Svein Bergum**: Managerial communication in telework, 2000.

No 809 **Flavius Gruian:** Energy-Aware Design of Digital Systems, 2000.
FiF-a 32 **Karin Hedström:** Kunskapsanvändning och kunskapsutveckling hos verksamhetskonsulter - Erfarenheter från ett FOU-samarbete, 2000.
No 808 **Linda Askenäs:** Affärssystemet - En studie om teknikens aktiva och passiva roll i en organisation, 2000.
No 820 **Jean Paul Meynard:** Control of industrial robots through high-level task programming, 2000.
No 823 **Lars Hult:** Publika Gränsytor - ett designexempel, 2000.
No 832 **Paul Pop:** Scheduling and Communication Synthesis for Distributed Real-Time Systems, 2000.
FiF-a 34 **Göran Hultgren:** Nätverksinriktad Förändringsanalys - perspektiv och metoder som stöd för förståelse och utveckling av affärsrelationer och informationssystem, 2000.
No 842 **Magnus Kald:** The role of management control systems in strategic business units, 2000.
No 844 **Mikael Cäker:** Vad kostar kunden? Modeller för intern redovisning, 2000.
FiF-a 37 **Ewa Braf**: Organisationers kunskapsverksamheter - en kritisk studie av ”knowledge management”, 2000.
FiF-a 40 **Henrik Lindberg:** Webbaserade affärsprocesser - Möjligheter och begränsningar, 2000.
FiF-a 41 **Benneth Christiansson:** Att komponentbasera informationssystem - Vad säger teori och praktik?, 2000.
No. 854 **Ola Pettersson:** Deliberation in a Mobile Robot, 2000.
No 863 **Dan Lawesson**: Towards Behavioral Model Fault Isolation for Object Oriented Control Systems, 2000.
No 881 **Johan Moe:** Execution Tracing of Large Distributed Systems, 2001.
No 882 **Yuxiao Zhao:** XML-based Frameworks for Internet Commerce and an Implementation of B2B e-procurement, 2001.
No 890 **Annika Flycht-Eriksson:** Domain Knowledge Management inInformation-providing Dialogue systems, 2001.
FiF-a 47 **Per-Arne Segerkvis**t: Webbaserade imaginära organisationers samverkansformer: Informationssystemarkitektur och aktörssamverkan som förutsättningar för affärsprocesser, 2001.
No 894 **Stefan Svarén:** Styrning av investeringar i divisionaliserade företag - Ett koncernperspektiv, 2001.
No 906 **Lin Han**: Secure and Scalable E-Service Software Delivery, 2001.
No 917 **Emma Hansson:** Optionsprogram för anställda - en studie av svenska börsföretag, 2001.
No 916 **Susanne Odar:** IT som stöd för strategiska beslut, en studie av datorimplementerade modeller av verksamhet som stöd för beslut om anskaffning av JAS 1982, 2002.
FiF-a-49 **Stefan Holgersson:** IT-system och filtrering av verksamhetskunskap - kvalitetsproblem vid analyser och beslutsfattande som bygger på uppgifter hämtade från polisens IT-system, 2001.
FiF-a-51 **Per Oscarsson:**Informationssäkerhet i verksamheter - begrepp och modeller som stöd för förståelse av informationssäkerhet och dess hantering, 2001.
No 919 **Luis Alejandro Cortes:** A Petri Net Based Modeling and Verification Technique for Real-Time Embedded Systems, 2001.
No 915 **Niklas Sandell:** Redovisning i skuggan av en bankkris - Värdering av fastigheter. 2001.
No 931 **Fredrik Elg:** Ett dynamiskt perspektiv på individuella skillnader av heuristisk kompetens, intelligens, mentala modeller, mål och konfidens i kontroll av mikrovärlden Moro, 2002.
No 933 **Peter Aronsson:** Automatic Parallelization of Simulation Code from Equation Based Simulation Languages, 2002.
No 938 **Bourhane Kadmiry**: Fuzzy Control of Unmanned Helicopter, 2002.
No 942 **Patrik Haslum**: Prediction as a Knowledge Representation Problem: A Case Study in Model Design, 2002.
No 956 **Robert Sevenius:** On the instruments of governance - A law & economics study of capital instruments in limited liability companies, 2002.
FiF-a 58 **Johan Petersson:** Lokala elektroniska marknadsplatser - informationssystem för platsbundna affärer, 2002.
No 964 **Peter Bunus:** Debugging and Structural Analysis of Declarative Equation-Based Languages, 2002.
No 973 **Gert Jervan:** High-Level Test Generation and Built-In Self-Test Techniques for Digital Systems, 2002.
No 958 **Fredrika Berglund:** Management Control and Strategy - a Case Study of Pharmaceutical Drug Development, 2002.
FiF-a 61 **Fredrik Karlsson:** Meta-Method for Method Configuration - A Rational Unified Process Case, 2002.
No 985 **Sorin Manolache:** Schedulability Analysis of Real-Time Systems with Stochastic Task Execution Times, 2002.
No 982 **Diana Szentiványi:** Performance and Availability Trade-offs in Fault-Tolerant Middleware, 2002.
No 989 **Iakov Nakhimovski:** Modeling and Simulation of Contacting Flexible Bodies in Multibody Systems, 2002.
No 990 **Levon Saldamli:** PDEModelica - Towards a High-Level Language for Modeling with Partial Differential Equations, 2002.
No 991 **Almut Herzog:** Secure Execution Environment for Java Electronic Services, 2002.
No 999 **Jon Edvardsson:** Contributions to Program- and Specification-based Test Data Generation, 2002
No 1000 **Anders Arpteg:** Adaptive Semi-structured Information Extraction, 2002.
No 1001 **Andrzej Bednarski:** A Dynamic Programming Approach to Optimal Retargetable Code Generation for Irregular Architectures, 2002.
No 988 **Mattias Arvola:** Good to use! : Use quality of multi-user applications in the home, 2003.
FiF-a 62 **Lennart Ljung:** Utveckling av en projektivitetsmodell - om organisationers förmåga att tillämpa projektarbetsformen, 2003.
No 1003 **Pernilla Qvarfordt:** User experience of spoken feedback in multimodal interaction, 2003.
No 1005 **Alexander Siemers:** Visualization of Dynamic Multibody Simulation With Special Reference to Contacts, 2003.
No 1008 **Jens Gustavsson:** Towards Unanticipated Runtime Software Evolution, 2003.
No 1010 **Calin Curescu:** Adaptive QoS-aware Resource Allocation for Wireless Networks, 2003.
No 1015 **Anna Andersson:** Management Information Systems in Process-oriented Healthcare Organisations, 2003.
No 1018 **Björn Johansson:** Feedforward Control in Dynamic Situations, 2003.
No 1022 **Traian Pop:** Scheduling and Optimisation of Heterogeneous Time/Event-Triggered Distributed Embedded Systems, 2003.
FiF-a 65 **Britt-Marie Johansson:** Kundkommunikation på distans - en studie om kommunikationsmediets betydelse i affärstransaktioner, 2003.

No 1024 **Aleksandra Tešanovic:** Towards Aspectual Component-Based Real-Time System Development, 2003.
No 1034 **Arja Vainio-Larsson**: Designing for Use in a Future Context - Five Case Studies in Retrospect, 2003.
No 1033 **Peter Nilsson:** Svenska bankers redovisningsval vid reservering för befarade kreditförluster - En studie vid införandet av nya redovisningsregler, 2003.
FiF-a 69 **Fredrik Ericsson:** Information Technology for Learning and Acquiring of Work Knowledge, 2003.
No 1049 **Marcus Comstedt:** Towards Fine-Grained Binary Composition through Link Time Weaving, 2003.
No 1052 **Åsa Hedenskog:** Increasing the Automation of Radio Network Control, 2003.
No 1054 **Claudiu Duma:** Security and Efficiency Tradeoffs in Multicast Group Key Management, 2003.
FiF-a 71 **Emma Eliason:** Effektanalys av IT-systems handlingsutrymme, 2003.
No 1055 **Carl Cederberg:** Experiments in Indirect Fault Injection with Open Source and Industrial Software, 2003.
No 1058 **Daniel Karlsson:** Towards Formal Verification in a Component-based Reuse Methodology, 2003.
FiF-a 73 **Anders Hjalmarsson:** Att etablera och vidmakthålla förbättringsverksamhet - behovet av koordination och interaktion vid förändring av systemutvecklingsverksamheter, 2004.
No 1079 **Pontus Johansson:** Design and Development of Recommender Dialogue Systems, 2004.
No 1084 **Charlotte Stoltz:** Calling for Call Centres - A Study of Call Centre Locations in a Swedish Rural Region, 2004.
FiF-a 74 **Björn Johansson:** Deciding on Using Application Service Provision in SMEs, 2004.
No 1094 **Genevieve Gorrell:** Language Modelling and Error Handling in Spoken Dialogue Systems, 2004.
No 1095 **Ulf Johansson:** Rule Extraction - the Key to Accurate and Comprehensible Data Mining Models, 2004.
No 1099 **Sonia Sangari:** Computational Models of Some Communicative Head Movements, 2004.
No 1110 **Hans Nässla:** Intra-Family Information Flow and Prospects for Communication Systems, 2004.
No 1116 **Henrik Sällberg:** On the value of customer loyalty programs - A study of point programs and switching costs, 2004.
FiF-a 77 **Ulf Larsson:** Designarbete i dialog - karaktärisering av interaktionen mellan användare och utvecklare i en systemutvecklingsprocess, 2004.
No 1126 **Andreas Borg:** Contribution to Management and Validation of Non-Functional Requirements, 2004.
No 1127 **Per-Ola Kristensson:** Large Vocabulary Shorthand Writing on Stylus Keyboard, 2004.
No 1132 **Pär-Anders Albinsson:** Interacting with Command and Control Systems: Tools for Operators and Designers, 2004.
No 1130 **Ioan Chisalita:** Safety-Oriented Communication in Mobile Networks for Vehicles, 2004.
No 1138 **Thomas Gustafsson:** Maintaining Data Consistency im Embedded Databases for Vehicular Systems, 2004.
No 1149 **Vaida Jakoniené:** A Study in Integrating Multiple Biological Data Sources, 2005.
No 1156 **Abdil Rashid Mohamed**: High-Level Techniques for Built-In Self-Test Resources Optimization, 2005.
No 1162 **Adrian Pop:** Contributions to Meta-Modeling Tools and Methods, 2005.
No 1165 **Fidel Vascós Palacios:** On the information exchange between physicians and social insurance officers in the sick leave process: an Activity Theoretical perspective, 2005.
FiF-a 84 **Jenny Lagsten:** Verksamhetsutvecklande utvärdering i informationssystemprojekt, 2005.
No 1166 **Emma Larsdotter Nilsson:** Modeling, Simulation, and Visualization of Metabolic Pathways Using Modelica, 2005.
No 1167 **Christina Keller:** Virtual Learning Environments in higher education. A study of students' acceptance of educational technology, 2005.
No 1168 **Cécile Åberg:** Integration of organizational workflows and the Semantic Web, 2005.
FiF-a 85 **Anders Forsman:** Standardisering som grund för informationssamverkan och IT-tjänster - En fallstudie baserad på trafikinformationstjänsten RDS-TMC, 2005.
No 1171 **Yu-Hsing Huang:** A systemic traffic accident model, 2005.
FiF-a 86 **Jan Olausson:** Att modellera uppdrag - grunder för förståelse av processinriktade informationssystem i transaktionsintensiva verksamheter, 2005.
No 1172 **Petter Ahlström**: Affärsstrategier för seniorbostadsmarknaden, 2005.
No 1183 **Mathias Cöster**: Beyond IT and Productivity - How Digitization Transformed the Graphic Industry, 2005.
No 1184 **Åsa Horzella**: Beyond IT and Productivity - Effects of Digitized Information Flows in Grocery Distribution, 2005.
No 1185 **Maria Kollberg**: Beyond IT and Productivity - Effects of Digitized Information Flows in the Logging Industry, 2005.
No 1190 **David Dinka**: Role and Identity - Experience of technology in professional settings, 2005.
No 1191 **Andreas Hansson**: Increasing the Storage Capacity of Recursive Auto-associative Memory by Segmenting Data, 2005.
No 1192 **Nicklas Bergfeldt:** Towards Detached Communication for Robot Cooperation, 2005.
No 1194 **Dennis Maciuszek:** Towards Dependable Virtual Companions for Later Life, 2005.
No 1204 **Beatrice Alenljung:** Decision-making in the Requirements Engineering Process: A Human-centered Approach, 2005
No 1206 **Anders Larsson:** System-on-Chip Test Scheduling and Test Infrastructure Design, 2005.
No 1207 **John Wilander:** Policy and Implementation Assurance for Software Security, 2005.
No 1209 **Andreas Käll:** Översättningar av en managementmodell - En studie av införandet av Balanced Scorecard i ett landsting, 2005.
No 1225 **He Tan:** Aligning and Merging Biomedical Ontologies, 2006.
No 1228 **Artur Wilk:** Descriptive Types for XML Query Language Xcerpt, 2006.
No 1229 **Per Olof Pettersson:** Sampling-based Path Planning for an Autonomous Helicopter, 2006.
No 1231 **Kalle Burbeck:** Adaptive Real-time Anomaly Detection for Safeguarding Critical Networks, 2006.
No 1233 **Daniela Mihailescu:** Implementation Methodology in Action: A Study of an Enterprise Systems Implementation Methodology, 2006.
No 1244 **Jörgen Skågeby:** Public and Non-public gifting on the Internet, 2006.
No 1248 **Karolina Eliasson:** The Use of Case-Based Reasoning in a Human-Robot Dialog System, 2006.
No 1263 **Misook Park-Westman:** Managing Competence Development Programs in a Cross-Cultural Organisation-What are the Barriers and Enablers, 2006.
FiF-a 90 **Amra Halilovic:** Ett praktikperspektiv på hantering av mjukvarukomponenter, 2006.
No 1272 **Raquel Flodström**: A Framework for the Strategic Management of Information Technology, 2006.

No 1277 **Viacheslav Izosimov:** Scheduling and Optimization of Fault-Tolerant Embedded Systems, 2006.
No 1283 **Håkan Hasewinkel:** A Blueprint for Using Commercial Games off the Shelf in Defence Training, Education and Research Simulations, 2006.
FiF-a 91 **Hanna Broberg:** Verksamhetsanpassade IT-stöd - Designteori och metod, 2006.
No 1286 **Robert Kaminski:** Towards an XML Document Restructuring Framework, 2006
No 1293 **Jiri Trnka:** Prerequisites for data sharing in emergency management, 2007.
No 1302 **Björn Hägglund:** A Framework for Designing Constraint Stores, 2007.
No 1303 **Daniel Andreasson**: Slack-Time Aware Dynamic Routing Schemes for On-Chip Networks, 2007.
No 1305 **Magnus Ingmarsson:** Modelling User Tasks and Intentions for Service Discovery in Ubiquitous Computing, 2007.
No 1306 **Gustaf Svedjemo**: Ontology as Conceptual Schema when Modelling Historical Maps for Database Storage, 2007.
No 1307 **Gianpaolo Conte**: Navigation Functionalities for an Autonomous UAV Helicopter, 2007.
No 1309 **Ola Leifler**: User-Centric Critiquing in Command and Control: The DKExpert and ComPlan Approaches, 2007.
No 1312 **Henrik Svensson**: Embodied simulation as off-line representation, 2007.
No 1313 **Zhiyuan He:** System-on-Chip Test Scheduling with Defect-Probability and Temperature Considerations, 2007.
No 1317 **Jonas Elmqvist:** Components, Safety Interfaces and Compositional Analysis, 2007.
No 1320 **Håkan Sundblad:** Question Classification in Question Answering Systems, 2007.
No 1323 **Magnus Lundqvist**: Information Demand and Use: Improving Information Flow within Small-scale Business Contexts, 2007.
No 1329 **Martin Magnusson:** Deductive Planning and Composite Actions in Temporal Action Logic, 2007.
No 1331 **Mikael Asplund:** Restoring Consistency after Network Partitions, 2007.
No 1332 **Martin Fransson:** Towards Individualized Drug Dosage - General Methods and Case Studies, 2007.
No 1333 **Karin Camara:** A Visual Query Language Served by a Multi-sensor Environment, 2007.
No 1337 **David Broman:** Safety, Security, and Semantic Aspects of Equation-Based Object-Oriented Languages and Environments, 2007.
No 1339 **Mikhail Chalabine:** Invasive Interactive Parallelization, 2007.
No 1351 **Susanna Nilsson:** A Holistic Approach to Usability Evaluations of Mixed Reality Systems, 2008.
No 1353 **Shanai Ardi:** A Model and Implementation of a Security Plug-in for the Software Life Cycle, 2008.
No 1356 **Erik Kuiper:** Mobility and Routing in a Delay-tolerant Network of Unmanned Aerial Vehicles, 2008.
No 1359 **Jana Rambusch**: Situated Play, 2008.
No 1361 **Martin Karresand:** Completing the Picture - Fragments and Back Again, 2008.
No 1363 **Per Nyblom:** Dynamic Abstraction for Interleaved Task Planning and Execution, 2008.
No 1371 **Fredrik Lantz:**Terrain Object Recognition and Context Fusion for Decision Support, 2008.
No 1373 **Martin Östlund:** Assistance Plus: 3D-mediated Advice-giving on Pharmaceutical Products, 2008.
No 1381 **Håkan Lundvall:** Automatic Parallelization using Pipelining for Equation-Based Simulation Languages, 2008.
No 1386 **Mirko Thorstensson:** Using Observers for Model Based Data Collection in Distributed Tactical Operations, 2008.
No 1387 **Bahlol Rahimi:** Implementation of Health Information Systems, 2008.
No 1392 **Maria Holmqvist:** Word Alignment by Re-using Parallel Phrases, 2008.
No 1393 **Mattias Eriksson**: Integrated Software Pipelining, 2009.
No 1401 **Annika Öhgren:** Towards an Ontology Development Methodology for Small and Medium-sized Enterprises, 2009.
No 1410 **Rickard Holsmark:** Deadlock Free Routing in Mesh Networks on Chip with Regions, 2009.