# The MANGO Clockless Network-on-Chip: Concepts and Implementation

PhD Thesis
by
Tobias Bjerregaard

Kgs. Lyngby 2005
IMM-PHD-2005-153

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-PHD: ISSN 0909-3192

# Summary

This dissertation addresses aspects of on-chip interconnection networks. The scientific contributions of the thesis are twofold. First, a survey of existing research is made. The survey categorizes, structures and reviews a wide spectrum of work in this new academic field, giving an overview of the state-of-the-art. Secondly, issues are covered which relate to the practical design of a network enabling a modular and scalable design flow for giga scale system-on-chip designs. Proof of concepts is given by their implementation in the MANGO (*Message-passing Asynchronous Network-on-chip providing Guaranteed services over OCP interfaces*) network-on-chip (NoC) architecture, which was developed during the course of the PhD-project.

The main body of the thesis is composed of a set of research papers. One of these is the mentioned survey, while five other papers explain concepts of the MANGO architecture, their implementation, and deployment in a complete MANGO-based system. Preceeding the papers, an introduction provides an overview of the thesis and explains the key features of MANGO, which are: clockless implementation, guaranteed communication services and standard socket access points. Also, the introduction touches upon industrial use of NoC.

# Resumé

Denne afhandling adresserer aspekter af on-chip interkonnektions netværk. Afhandlingens videnskabelige bidrag er todelt. Først er lavet en oversigt over eksisterende forskning. Dette survey kategoriserer og strukturerer et bredt spektrum af arbejde indenfor det nye akademiske område. Derudover dækkes udviklingen af et netværk som muliggør et modulært og skalerbart design flow for giga scale system-on-chip design. Bevis for koncepterne gives ved deres implementering i MANGO (*Message-passing Asynchronous Network-on-chip providing Guaranteed services over OCP interfaces*) network-on-chip (NoC) arkitekturen, som er blevet udviklet under PhD-projektets forløb.

Afhandlingen består i sin hovedvægt af en samling forskningsartikler. Den ene er det nævnte survey, mens fem andre artikler beskriver nøglekoncepter af MANGO arkitekturen, og koncepternes implementering i et komplet, MANGO-baseret system. Forud for disse artikler giver en introduktion en oversigt over afhandlingen, og forklarer hovedideerne ved MANGO: klokløs implementering, garanteret kommunikationsservice og standard socket access punkter. I introduktionen berøres også industriel anvendelse af NoC.

# Preface

This thesis was prepared at Informatics and Mathematical Modelling, at the Technical University of Denmark in partial fulfillment of the requirements for acquiring the PhD degree. The PhD project was supervised by Associate Professor Jens Sparsø.

Though the main focus of the thesis is network-on-chip, the experience of Jens Sparsø within the field of asynchronous circuit design, has helped shape the project from the very beginning, in the recognition of the value of globally asynchronous locally synchronous system-on-chip design.

This final version of the thesis is different from the version submitted in September 2005, in that papers A and C have been revised based on reviewers comments. Though the PhD degree was granted, purely on basis of the original submission, I was encouraged by the thesis examiners to include the most recent versions of the papers.

Lyngby, February 2006

Tobias Bjerregaard

# List of Publications

This section contains a list of the publications written during the course of the PhD project. The following research papers are included in the thesis:

1. Paper A: Tobias Bjerregaard and Shankar Mahadevan. A Survey of Research and Practices of Network-on-Chip. *ACM Computing Surveys*. Accepted.

2. Paper B: Tobias Bjerregaard and Jens Sparsø. A Router Architecture for Connection-Oriented Service Guarantees in the MANGO Clockless Network-on-Chip. *Proceedings of the Design, Automation and Test in Europe Conference*, IEEE 2005. Published.

3. Paper C: Tobias Bjerregaard and Jens Sparsø. Implementation of Guaranteed Services in the MANGO Clockless Network-on-Chip. *IEE Proceedings: Computers and Digital Techniques*. Submitted.

4. Paper D: Tobias Bjerregaard and Jens Sparsø. A Scheduling Discipline for Latency and Bandwidth Guarantees in Asynchronous Network-on-Chip. *Proceedings of the 11th IEEE International Symposium on Advanced Research in Asynchronous Circuits and Systems*, IEEE 2005. Published.

5. Paper E: Tobias Bjerregaard, Shankar Mahadevan, Rasmus Grøndahl Olsen and Jens Sparsø. An OCP Compliant Network Adapter for GALS-based SoC Design using the MANGO Network-on-Chip. *Proceedings of the International Symposium on System-on-Chip*, IEEE 2005. Published.

6. Paper F: Tobias Bjerregaard. Programming and Using Connections in the MANGO Network-on-Chip. *To be submitted.*

The following research papers, which were also written during the course of the PhD project, are not included in the thesis:

7. Girish Venkataramani, Tobias Bjerregaard, Tiberiu Chelcea and Seth Copen Goldstein. Hardware Compilation of Memory Abstractions. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. Accepted.

8. Girish Venkataramani, Tobias Bjerregaard, Tiberiu Chelcea and Seth Copen Goldstein. SOMA: A Tool for Synthesizing and Optimizing Memory Accesses in ASICs. *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis*, ACM/IEEE 2005. Published.

9. Tobias Bjerregaard and Jens Sparsø. Virtual Channel Designs for Guaranteeing Services in Asynchronous Network-on-Chip. *Proceedings of the 22nd IEEE Norchip Conference*, IEEE 2004. Published.

10. Tobias Bjerregaard, Shankar Mahadevan and Jens Sparsø. A Channel Library for Asynchronous Circuit Design Supporting Mixed-Mode Modeling. *Proceedings of the 14th International Workshop of Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation*, Springer 2004. Published.

In addition to the above research papers, the following patents have been submitted by the University:

11. Tobias Bjerregaard and Jens Sparsø. A Network, a System and a Node for use in the Network or System. *DK and US patents submitted*, 2005.

12. Tobias Bjerregaard. A Method of and a System for Controlling Access to a Shared Resource. *DK and US patents submitted*, 2005.

13. Tobias Bjerregaard. A Method and an Apparatus for Providing Timing Signals to a Number of Circuits, an Integrated Circuit and a Node. *DK and US patents submitted*, 2005.

# List of Abbreviations

This section provides a list of short hands used in the thesis.

**NoC** - Network-on-Chip

**SoC** - System-on-Chip

**IP core** - Intellectual Property core (functional unit in a SoC)

**GALS** - Globally Asynchronous Locally Synchronous (systems which do not have a global clock, but wherein each submodule is independently clocked)

**OCP** - Open Core Protocol (socket for on-chip integration of IP cores)

**MANGO** - Message-passing Asynchronous Network-on-chip providing Guaranteed services over OCP interfaces

**QoS** - Quality of Service

**GS** - Guaranteed Services (refers to routing services)

**BE** - Best-Effort (refers to routing services)

**ALG** - Asynchronous Latency Guarantees (scheduling discipline)

**TDM** - Time Division Multiplexing

**VC** - Virtual Channel

**DMA** - Direct Memory Access

**TFlops** - Tera Floating-point Operations

**CMOS** - Complementary Metal Oxide Semiconductor

# Acknowledgements

A number of people have been of indispensable help to me, during the writing of this thesis. I want to thank my son Metro for his love and for being such a cool and beautiful boy, and my whole family for their unlimited support. My fellow PhD student, collaborator and office mate Shankar Mahadevan for company and hefty (at times *very* much so) and fruitful discussions. My supervisor Jens Sparsø for constructive feedback and for his efforts in motivating me to structure and focus my ideas. Per Friis for keeping the servers running, and Maria Jensen and Helle Job for keeping track of all the papers. During the project, I also had the great pleasure of co-supervising a number of students during their Master Thesis projects within the subject area, all of which helped inspire this work. I thank all of these students Mikkel Stensgaard, Rasmus Grøndahl Olsen, Mathias Nicolajsen Kjærgaard, Juliana Zhou and Thomas Christensen.

Finally, I thank my fingers for doing the typing, my brain cells for doing the thinking and the moon for shining its creative light. Now the question of what it is worth to humanity remains.

*Tobias Bjerregaard*
*Lyngby, September 2005.*

# Contents

Chapter 1

# Introduction

On-chip networks constitute a viable solution space to emerging system-on-chip (SoC) design challenges [10][6][15]. As a replacement for busses and point-to-point links, they hold the potential for a much more scalable, modular and flexible design flow, as well as addressing fundamental physical-level problems introduced when scaling microchip technologies into deep submicron geometries. This PhD thesis addresses issues of network-on-chip (NoC) design and usage. The focus is on MANGO (*Message-passing Asynchronous Network-on-chip providing Guaranteed services over OCP interfaces*), the NoC architecture developed during the course of the PhD project, by the author and others in the System-on-Chip Group of the department of Informatics and Mathematical Modelling at the Technical University of Denmark. The author's contributions are mainly at the lower levels of abstraction, conceptualizing, implementing and formally proving methods and circuits used in the network adapters, routers and links. These elements provide the functionality, on which higher-level usage of the network is to be based. As NoC represents a rather new field, no major work providing an overview of existing research exists, and in addition to the development of the MANGO architecture, the thesis contributes a survey of research within the field.

The thesis is composed of a set of research papers, which have been written during the PhD project. Most have been published, or accepted for publication, in relevant journals and conference proceedings. In the following sections, I will

first describe key features of MANGO. Then I will provide an overview of the papers included in the thesis. After this the present and future of practical, industrial NoC usage is touched upon. In Chapter 2 I make a conclusion on the thesis work and on MANGO.

There may occur details in this introduction, which are not readily understandable. These however will become more clear, after reading the included papers that follow. The concluding remarks of Chapter 2 can be returned to, after reading the papers. One may also choose to skip Sections 1.3 and 1.4, saving these for after reading the papers. Since the papers are written as stand-alone works, be prepared for overlaps in the introductory and motivating sections. As the scientific weight of the thesis lies in the papers, I will mostly make references in this introdution only when addressing key concepts. For extensive referencing, I refer to the papers, in particular paper A – the survey paper.

## 1.1 The MANGO Network-on-Chip

The central goal addressed with MANGO is the realization of a modular and scalable design flow for giga scale SoC designs. Issues in relation to this goal include the challenge of global synchronization in large chips, the unpredictable performance resulting from complex, dynamic dependencies when using a shared communication media, and the need for increased design productivity in order to exploit the growing amount of on-chip resources available to chip designers. Key features of MANGO addressing these issues are:

**(i) Clockless implementation.** The MANGO links and routers are constructed entirely using clockless, or *asynchronous*, circuits [23]. As such, no global synchronization signal is needed in a MANGO-based system. The IP cores can be clocked individually. This facilitates a globally asynchronous locally synchronous (GALS) system [9][16][18].

**(ii) Guaranteed Communication Services.** By providing hard bandwidth and latency guarantees, over connections in the network, it is possible to get a handle on the complex dynamic performance interdependencies resulting from the use of a shared communication media [22][17][8][13]. Benefits of guaranteed services (GS) are that local changes do not have global effects, that it is feasible to verify a system analytically rather than through simulation, and that real time responsiveness becomes possible from a programming point-of-view. In addition to connection-oriented GS, MANGO also provides connection-less best-effort (BE) routing services.

**(iii) Standard socket access points.** Access points in MANGO adhere to the Open Core Protocol (OCP) [4]. OCP is an industry standard for on-chip integration of IP cores. It provides a flexible family of synchronous core-centric interfaces, based on memory-mapped access. Network adapters in MANGO provide read/write-style OCP transactions based on the primitive message-passing services of the network. Providing standard socket access points [20][21] is a step towards closing the widening design-productivity gap, by allowing design reuse and decoupling of IP cores in the system.

While the papers in this thesis describe novel clockless circuits used in MANGO, the basic concepts are equally applicable in a clocked implementation. Such a network could have the benefit of being synthesizable from a high-level HDL description. However it would not benefit from advantages of clockless circuits as inherent global timing closure, zero dynamic idle power consumption and low forward latency. An additional benefit of using clockless circuit techniques, particularly relevant for NoC, is the fact that while any data communication network needs to implement data driven flow control, this functionality is an integral part of clockless circuits, as these are data driven by nature.

A design decision of MANGO was the implementation of read/write-style (memory-mapped) interfaces. It is presently not fully clear, what the choice of interfaces will be in future SoCs. It seemed natural however to support memory-mapped interfaces in MANGO, since these dominate in computer systems today, leveraging the legacy of busses. The overhead of performing single reads and writes over a NoC is high however, in particular in terms of latency, and one can argue that e.g. media applications may benefit considerably from using streaming interfaces instead. As the backbone of MANGO – the links and routers – makes use of message-passing, the implementation of a streaming interface would be trivial.

## 1.2 Overview of the Included Papers

The included papers should be read in the order that they appear. Paper A is a survey paper, and is meant as an introduction to the field. The remaining papers concern technical details of MANGO. Paper B provides an overview of the MANGO architecture and describes the router. Hereafter, in paper C, follows a detailed circuit level presentation of some of the basic circuits of MANGO: the implementation of virtual channel links and fair-share access to provide bandwidth guarantees. Paper D details a novel link access scheduling scheme, used to provide bandwidth and latency guarantees which are not inversely dependent of each other. Paper E explains the network adapters, which implement the OCP

compliant access points of the network, and paper F provides a perspective on all the previous papers, by presenting a core-centric view, and programming model, of a MANGO-based system. In the following, I will give a short review of each paper.

**Paper A: A Survey of Research and Practices of Network-on-Chip.**
*Tobias Bjerregaard and Shankar Mahadevan*
This paper, which has been accepted to appear in *ACM Computing Surveys*, provides an overview of the state-of-the-art of NoC research. Firstly, it gives motivation for the usage of segmented interconnection networks in system-on-chip (SoC) designs, and explains the basics of NoC. Fundamental building blocks are named, a taxonomy for datagrams is defined and an adaption for NoC of the OSI standard for layered network communication is made. Hereafter a review of existing research is made. Issues range from link implementation to design methodologies, also covering topics like performance analysis and traffic characterization. Finally a number of case studies, of existing NoC solutions, are given. The paper was written as an equal and joint effort between myself and my fellow PhD student Shankar Mahadevan. While Shankar is mostly involved in issues at higher levels of abstraction, such as modeling, and my own work involves circuit design and other low level issues, our involvement in writing the survey has been in all areas. The contribution of this paper is to provide an overview and a structuring of a wide spectrum of state-of-the-art NoC research.

**Paper B: A Router Architecture for Connection-Oriented Service Guarantees in the MANGO Clockless Network-on-Chip.**
*Tobias Bjerregaard and Jens Sparsø*
This paper was published by the IEEE Computer Society Press in the proceedings of the *Design, Automation and Test in Europe Conference (DATE)*, 2005. It details the architecture of the MANGO routers, and explains how these can be used to provide end-to-end service guarantees on connections. The paper overlaps slightly with paper C, however whereas paper C details the links, this paper provides more details on the routers and their programming. The main contribution of the paper is the development of a router architecture, by which local link access arbitration can be used to provide any type of global end-to-end service guarantees.

**Paper C: Implementation of Guaranteed Services in the MANGO Clockless Network-on-Chip.**
*Tobias Bjerregaard and Jens Sparsø*
In this paper, which is an invited submission to *IEE Computers and Digital Techniques* based on a paper published by the authors at the *Norchip IEEE Conference* [7], some fundamental circuits of MANGO are presented. The paper details implementation of delay insensitive inter-router links and circuits used in sharing of physical links between *virtual channels* (VCs). VCs are the

basic building blocks of *virtual circuits*, which are a necessity of establishing GS connections in MANGO. The main contributions of the paper include the development of a clockless VC flow control method which requires only a single control wire, and the implementation of high-performance clockless circuits for providing fair bandwidth share access to links.

**Paper D: A Scheduling Discipline for Latency and Bandwidth Guarantees in Asynchronous Network-on-Chip.**
*Tobias Bjerregaard and Jens Sparsø*
Published in the proceedings of the *IEEE International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, in March 2005, this paper presents a scheduling discipline for link access, called *Asynchronous Latency Guarantees* (ALG), and its clockless implementation. The contribution of the paper is the development of ALG scheduling, which provides latency guarantees which are not inversely dependent on the bandwidth guarantees, as is the case with time division multiplexed scheduling. Formal proof of the ALG discipline is provided. The paper achieved the best paper award of the conference.

**Paper E: An OCP Compliant Network Adapter for GALS-based SoC Design using the MANGO Network-on-Chip.**
*Tobias Bjerregaard, Shankar Mahadevan, Rasmus Grøndahl Olsen and Jens Sparsø*
This paper, which has been published at the *International Symposium on System-on-Chip (SOC)*, November 2005, addresses the third key feature of MANGO: standard socket access points. An OCP compliant network adapter (NA) is presented, which makes it possible to address other cores attached to the network by OCP read and write transactions. The adapter also handles synchronization between the clockless network and the clocked OCP socket, hence enabling GALS-type SoC design. The main contribution of the paper is the mixed clocked/clockless NA architecture, which appropriately leverages the advantages particular to either circuit style.

**Paper F: Programming and Using Connections in the MANGO Network-on-Chip.**
*Tobias Bjerregaard*
The final paper of the thesis provides a perspective on papers B through E, by presenting a core-centric view of a MANGO-based system. The system is build from the building blocks introduced in each of the previous papers, and shows how such a system performs, from the point-of-view of the IP cores. The GS provided by the routers is based on ALG scheduling, and the links are delay insensitive and pipelined. It is shown how connections with end-to-end guarantees are programmed, and how pipelining links has a minimal impact on performance, due to the forward latency of clockless circuits being much

smaller than the cycle time. Thus bandwidth is gained with very little penalty on latency. The contribution of the paper is a simple and intuitive core-centric programming model, by which any master core – such as a microprocessor – in the system can program GS connections into the network, using OCP write commands.

## 1.3 Industrial Use of NoC

However interesting a theoretical work of engineering research may be, the real value of it is not revealed until industrially applied. The survey provided in paper A details mainly academia related NoC research. In the following, a perspective to this will be given by reflecting on the industrial deployment of NoC-type solutions.

Until recently, a widespread use of multiprocessor systems has not been practically feasible. The overhead of designing and using such systems has for a long time exceeded the advantages. CMOS scaling, computation intensive multimedia applications and power constrained mobile systems however, have pushed towards distributed, multi-core systems. In effect, this evolution incurs a segmented communication infrastructure as well. While practical deployment of NoC-type communication structures in commercial chips most often is hidden, the trend is clear, and commercial designs are starting to appear. In the following a few examples of industrial use of NoC are provided. Undoubtly many more are in the pipeline.

In the spring of 2006, Sony Computer Entertainment Incorporated will introduce the Play Station 3 (PS3), a state-of-the-art multimedia entertainment console, with an acclaimed performance of up to 1 TFlops. The PS3 is based on the Cell processor [14] developed as a result of a collaboration between Sony, Toshiba, and IBM. The Cell processor is the first in a new generation of multi-core, general purpose processor architectures. Along with memory and I/O controllers, it consists of 9 processors, connected by a high-speed, segmented interconnection bus – a NoC – in a dual ring topology. This communication architecture was devised in order to adhere to the extremely high bandwidth requirements of present day multimedia applications. A variety of traffic is communicated on this shared network, in the form of memory accesses, DMA streaming, as well as message-passing. To prevent starvation and enhance real time reponsiveness, the network uses a token scheme to allocate bandwidth. In recognition of the complexity of effectively exploiting the raw compute power of such an architecture, a considerable effort went into the development of software tools in parallel with the hardware platform.

Philips research has invested a great deal of resources into their Æthereal NoC [11]. While commercial use of this architecture is still to be publicly announced, one could guess that NoC-type solutions already exist in Philips products, or will appear in the near future. In the context of embedded computing systems, there is an increasing drive towards multi-core SoC design, in order to adhere to the requirements of low power, high performance devices. In [12] Kees Goossens of Philips Research addresses issues related to using NoC in consumer electronics. First, commercial application domains are identified and the basic requirements for these are stated. Application domains are converging and systems are increasingly embedded. System behaviour is often real-time and safety critical, as many systems have a concrete interaction with the real world. Reliable and predictable behaviour of the devices is the norm. Finally, price is a critical factor in consumer products. The paper concludes that NoC has great promise as a design-flow tool by 1) addressing deep submicron challenges and 2) offering a structured view on communication between IP cores. This leads to faster time-to-market, more predictable design cycles and more reliable designs.

Sonics [3] is a company which has focussed entirely on providing solutions for on-chip communication. Based on the MicroNetwork concept [25], their *SMART Interconnects* is a highly configurable, scalable SoC inter-block communication system that integrally manages data, control, debug and test flows. Recent publications detail methods for providing quality-of-service (QoS) [24], as a means to decouple cores from each other in the system.

Recent years have also seen the emergence of a number of start-up companies, seeking to commercialize on novel, NoC-related ideas. In 2003 the French start-up Arteris [1] was founded. Arteris calls itself 'the network-on-chip company', and its products comprise a suite of tools for generating and debugging a NoC, and a library of configurable NoC components. Apart from routers, the component library includes mesochronous links to enable GALS systems, *network spies* which allow for on-the-fly monitoring of network communication, and network interface units providing support for a number of standard interface sockets, such as OCP, AHB and AXI. Another start-up is Silistix [2], a spin-off from the Advanced Processor Technologies Group at Manchester University. Their NoC is based on the clockless CHAIN [5], which targets extremely low power systems, e.g. CHAIN was demonstrated in a smart card implementation. It is not publically known in what direction Silistix is presently taking their NoC development.

The successful deployment of the NoC concept holds the potential to yield benefits in a wide range of systems. To this end, NoC-based design may very well prove to leverage the challenges at hand, in particular by enabling a scalable, modular design flow, but also by providing the performance required by high-end media applications.

## 1.4 Future Challenges

A major question that remains to be answered, concerns the need for quality-of-service in NoC. Hard, connection-oriented service guarantees incur ultimate predictability, and is hence desirable both from a real-time performance as well as from a design verification point-of-view. But it comes at a price, in terms of area overhead, reduced average performance and/or loss of routing flexibility. NoC design trade-offs concern balancing these costs, and different solutions express different types of overhead. In the Æthereal NoC [11] the area of a pure GS router is lower than that of a router which provides BE routing services. The overhead of GS is in terms of increased average latency, due to the need to await a time division multiplexing slot that has been reserved for a particular connection. Hence connections which have reserved few slots have high latency. In MANGO on the other hand, latency can be guaranteed independently of bandwidth. Here the overhead is mainly in terms of area, due to the need to buffer GS connections separately.

Having to specify every connection in the system explicitly also limits the flexibility. Envision a distributed, shared memory system. Each master would need to establish a connection to every slave in the system. Though at a more abstract or *virtual* level, this scenario parallels that of establishing a full mesh of dedicated point-to-point links, clearly a non-scalable solution. To this end, the combination of BE routing services and GS could constitute a viable solution space. In combining different services however, again we encounter the trade-off between features and overhead.

One could also question the need for *hard* guarantees. Possibly *soft* guarantees – QoS based on *statistical* guarantees, as known from connection-less macro networks – will suffice in most applications. Another possibility is the provision of hard guarantees in connection-less networks, based on a global knowledge of the traffic in the network. Currently, research is being conducted in this direction at a number of institutions around the world, including IMEC (Interuniversity MicroElectronics Center) in Belgium and the Department of Electrical and Computer Engineering at the National University of Singapore. Approaches include calculating acceptable injection rates to still allow bandwidth to be guaranteed, also, implementing global congestion detection feedback loops and dynamically controlling injection of packets into the network. The overhead in using BE routing but still providing guarantees would lie mainly in overdimensioning the network, creating a *bandwidth overhead*. This potentially leaves headroom for obtaining hard guarantees based on analysis of BE traffic, or to obtain acceptable statistical routing guarantees.

While *throughput* and *computation* performance improves, CMOS scaling incurs

an increasing communication *latency*. As argued in paper F, this is what incurs the need for communication-centric design approaches. NoC does not solve this problem, as it reflects fundamental physical properties of the fabrication technology. However, NoC has the potential to utilize the available technology as optimally as possible, in spite of the inherent limitations. This is basically done by pipelining and sharing, hence keeping bandwidth and wire utilization up while supporting large systems. Since bandwidth and scalability are the most important metrics of high-end media system design, a driving application domain for microchip systems today, NoC evidently displays advantages over traditional communication architectures based on busses.

Generally speaking, the research focus is shifting from the implementation of NoC, to the investigation of its optimal use. In [19] key research problems in NoC design are identified. Approaches are proposed to each of eight key problems, and open problems are stated. These relate to synthesis of the communication infrastructure, choice of communication paradigm and application mapping and optimization. In addition to problems of this type, challenges that need to be addressed in order to realize a complete and practically feasible NoC framework, relate to the programmability of tightly coupled, highly embedded, heterogeneous, multi-core systems, and their distributed memory subsystems.

Chapter 2

# Concluding Remarks

This thesis concerns the network-on-chip concept of employing a shared, segmented interconnection network for intra-chip communication. The working title of my PhD project was just that; 'Intra-chip Communication'. During the course of the project, it became clear that the future of this topic is embedded in the NoC concept. As stated in paper A, *"...NoC constitutes a unification of current trends of intra-chip communication, rather than an explicit new alternative"*.

As a contribution to the field, I have provided a structured overview of the state-of-the-art of NoC research. Also, I have identified a series of important NoC features, needed in realizing a modular and scalable design flow for giga scale SoC designs, and developed novel solutions to these. As a proof of the concepts, I have deployed the theoretical ground work in a practical implementation: the MANGO clockless network-on-chip architecture. In MANGO:

- A clockless implementation, with syncronization to clocked cores in network adapters, makes global timing closure inherent and enhances IP composability.
- Guaranteed routing services decouple subsystems and make analytical verification possible.

- OCP-compliant standard socket access points enable IP reuse.

All of these features help promoting modularity and scalability. The advantages of a clockless implementation include low forward latency in pipelined links, zero dynamic idle power consumption and inherent support for GALS systems.

The work done for this PhD thesis demonstrates the existence of realistic, low cost, high performance solutions to the challenges of designing giga scale microchips; the design-productivity gap can be bridged, and the fundamental short comings of scaling CMOS technologies can be counteracted.

Paper A

# A Survey of Research and Practices of Network-on-Chip

Tobias Bjerregaard and Shankar Mavadevan

Accepted for publication in *ACM Computing Surveys*.

# A Survey of Research and Practices of Network-on-Chip

TOBIAS BJERREGAARD
and
SHANKAR MAHADEVAN
Technical University of Denmark

The scaling of microchip technologies has enabled large scale systems-on-chip (SoC). Network-on-chip (NoC) research addresses global communication in SoC, involving: (i) a move from computation-centric to communication-centric design and (ii) the implementation of scalable communication structures. This survey presents a perspective on existing NoC research. We define the following abstractions: system, network adapter, network and link; to explain and structure the fundamental concepts. First, research relating to the actual network design is reviewed. Then system level design and modeling are discussed. We also evaluate performance analysis techniques. The research shows that NoC constitutes a unification of current trends of intra-chip communication, rather than an explicit new alternative.

Categories and Subject Descriptors: A.1 [**Introductory and Survey**]: ; B.4.3 [**Input/Output and Data-Communications**]: Interconnections; B.7.1 [**Integrated Circuits**]: Types and Design Styles; C.5.4 [**Computer System Implementation**]: VLSI Systems; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design; C.0 [**General**]: —*System Architectures*

General Terms: Design

Additional Key Words and Phrases: chip-area networks, communication-centric design, communication abstractions, GALS, GSI design, interconnects, network-on-chip, NoC, OCP, on-chip communication, SoC, sockets, system-on-chip, ULSI design

## 1. INTRODUCTION

Chip design has four distinct aspects: computation, memory, communication and I/O. As processing power has increased and data intensive applications have emerged, the challenge of the communication aspect in single-chip systems, Systems-on-Chip (SoC), has had increasing attention. This survey treats a prominent concept for communication in SoC known as Network-on-Chip (NoC). As will become clear in the following, NoC does not constitute an explicit new alternative for intra-chip communication, but is rather a concept which presents a unification of on-chip communication solutions.

This paper is a joint first author effort, authors in alfabetical order.
S. Mahadevan was funded by SoC-MOBINET (IST-2000-30094), Nokia and the Thomas B. Thrige Foundation.
Authors' address: Technical University of Denmark, Informatics and Mathematical Modelling, Richard Petersens Plads, Building 321, DK-2800 Lyngby, Denmark; email:{tob,sm}@imm.dtu.dk
This work is accepted for publication in ACM Computing Surveys.

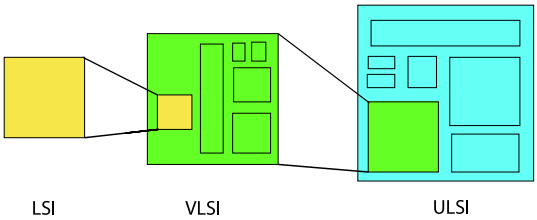



Fig. 1. When a technology matures, it leads to a paradigm shift in system scope. Shown here is the chip scope in LSI, VLSI and ULSI, the sequence of technologies leading to the enabling of SoC designs.

In this section we shall first briefly review the history of microchip technology that has led to a call for NoC based designs. With our minds on intra-chip communication, we will then look at a number of key issues of large-scale chip design, and finally show how the NoC concept provides a viable solution space to the problems presently faced by chip designers.

### 1.1 Intra-SoC Communication

The scaling of microchip technologies has lead to a doubling of available processing resources on a single chip every second year. Even though this is projected to slow down to a doubling every three years in the next few years for fixed chip sizes [ITRS 2003], the exponential trend is still in force. Though the evolution is continuous, the system level focus, or system scope, moves in steps. When a technology matures for a given implementation style, it leads to a paradigm shift. Examples of such shifts are moving from room- to rack-level systems (LSI - 1970s) and later from rack- to board-level systems (VLSI - 1980s). Recent technological advances allowing multi million transistor chips (currently well beyond 100M) have lead to a similar paradigm shift from board- to chip-level systems (ULSI - 1990s). The scope of a single chip has changed accordingly, as illustrated in Figure 1. In LSI systems a chip was a component of a system module (e.g. a bitslice in a bitslice processor), in VLSI systems a chip was a system level module (e.g. a processor or a memory), and in ULSI systems a chip constitutes an entire system (hence the term System-on-Chip or SoC). SoC opens up to the feasibility of a wide range of applications making use of massive parallel processing and tightly interdependent processes, some adhering to real-time requirements, bringing into focus new complex aspects of the underlying communication structure. Many of these aspects are addressed by NoC.

There are multiple ways to approach an understanding of NoC. Readers well versed in macro network theory may approach the concept by adapting proven techniques from multicomputer networks. Much work done in this area during the 80s and 90s can readily be built upon. Layered communication abstraction models, and decoupling of computation and communication are relevant issues. There are however, a number of basic differences between on- and off-chip communication. These generally reflect the difference in the cost ratio between wiring and processing resources.

Historically, computation has been expensive and communication cheap. With scaling microchip technologies this changed. Computation is becoming ever cheaper, while communication encounters fundamental physical limitations such as time-of-flight of electrical signals, power-use in driving long wires/cables, etc. In comparison with off-chip, on-chip communication is significantly cheaper. There is room for lots of wires on a chip. Thus

the shift to single-chip systems has relaxed system communication problems. However on-chip wires do not scale in the same manner as does transistors, and as we shall see in the following, the cost gap between computation and communication is widening. Meanwhile the differences between on- and off-chip wires make the direct scaling down of traditional multicomputer-networks sub-optimal for on-chip use.

In this survey we attempt to incorporate the whole range of design abstractions while relating to the current trends of intra-chip communication. With the *Giga Transistor Chip* era close at hand, the solution space of intra-chip communication is far from trivial. Below we have summarized a number of relevant key issues. Though not new, we find it worthwhile to go through them, as the NoC concept presents a possible unification of solutions for these. In Section 3 and 4, we will look into the details of research being done in relation to these issues, and their relevance for NoC.

—**Electrical wires.** Even though on-chip wires are cheap in comparison with off-chip wires, on-chip communication is becoming still more costly, in terms of both power and speed. As fabrication technologies scale down, wire resistance per mm is increasing while wire capacitance does not change much, the major part of the wire capacitance being due to edge capacitance [Ho et al. 2001]. For CMOS, the approximate point at which wire delays begin to dominate gate delays, was the $0.25\mu$m generation for aluminum, and $0.18\mu$m for copper interconnects, as first projected in [SIA 1997]. Shrinking metal pitches, in order to maintain sufficient routing densities, is appropriate at the local level where wire lengths also decrease with scaling. But global wire lengths do not decrease, and as local processing cycle times decrease, the time spend on global communication, relative to the time spend on local processing, increases drastically. Thus in future deep submicron (DSM) designs the interconnect effect will definitely dominate performance [Sylvester and Keutzer 2000]. Figure 2 taken from the International Technology Roadmap for Semiconductors [ITRS 2001] shows the projected relative delay for local wires, global wires and logic gates of the near future. Another issue of pressing importance concerns signal integrity. In DSM technologies, the wire models are unreliable, due to issues like fabrication uncertainties, crosstalk, noise sensitivity etc. These issues are especially applicable to long wires.

Due to these effects of scaling, it has become necessary to differentiate between local and global communication, and as transistors shrink the gap is increasing. The need for global communication schemes supporting single-chip systems has emerged.

—**System synchronization.** As chip technologies scale and chip speeds increase, it is becoming harder to achieve global synchronization. The drawbacks of the predominant design style of digital integrated circuits, strict global synchrony, are growing relative to the advantages. The clocktree needed to implement a globally synchronized clock is demanding increasing portions of the power and area budget, and even so the clock skew is claiming an ever larger relative part of the total cycle time available [Oklobdzija and Sparsø 2002][Oberg 2003]. This has triggered work on skew tolerant circuit design [Nedovic et al. 2003], which deals with clockskew by relaxing the need for timing margins, and on the use of optical waveguides for on-chip clock distribution [Piguet et al. 2004], the main purpose being to minimize power usage. Still power hungry skew adjustment techniques such as phase locked loops (PLL) and delay locked loops (DLL), traditionally used for chip-to-chip synchronization, are finding their way into single-chip systems [Kurd et al. 2001][Xanthopoulos et al. 2001].

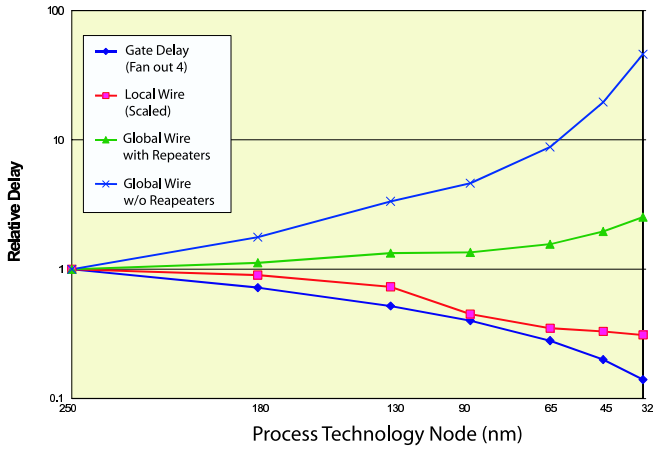



Fig. 2. Projected relative delay for local and global wires and for logic gates of near future technologies [ITRS 2001].

As a reaction to the inherent limitations of global synchrony, alternative concepts such as GALS (Globally Asynchronous Locally Synchronous systems) are being introduced. A GALS chip is made up of locally synchronous islands which communicate asynchronously [Chapiro 1984][Meincke et al. 1999][Muttersbach et al. 2000]. There are two main advantageous aspects of this method. One is the reducing of the synchronization problem to a number of smaller subproblems. The other relates to the integration of different IP (Intellectual Property) cores, easing the building of larger systems from individual blocks with different timing characteristics.

—**Design productivity.** The exploding amount of processing resources available in chip design together with a requirement for shortened design cycles have pushed the productivity requirements on chip designers. Between 1997 and 2002 the market demand reduced the typical design cycle by 50%. As a result of increased chip sizes, shrinking geometries and the availability of more metal layers, the design complexity increased 50 times in the same period [OCPIP 2003a]. To keep up with these requirements, IP reuse is pertinent. A new paradigm for design methodology is needed, which allows the design effort to scale linearly with system complexity.
Abstraction at register transfer level (RTL) was introduced with the ASIC design flow during the 90s, allowing synthesized standard cell design. This made it possible to design large chips within short design cycles, and synthesized RTL design is at present the defacto standard for making large chips quickly. But the availability of on-chip resources is outgrowing the productivity potential of even the ASIC design style. In order to utilize the exponential growth in number of transistors on each chip, even higher levels of abstraction must be applied. This can be done by introducing higher level communication abstractions, making for a layered design methodology enabling a partitioning of the design effort into minimally interdependent subtasks. Support for this at the hardware level includes standard communication sockets, allowing IP cores from different vendors to be plugged effortlessly together. This is particularly pertinent in complex multi-processor system-on-chip (MPSoC) designs. Also, the development of design

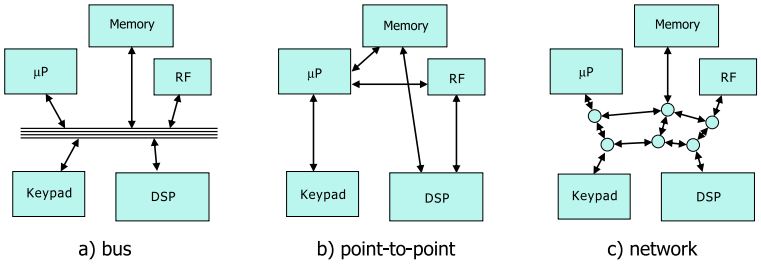



Fig. 3. Examples of communication structures in Systems-on-Chip. a) traditional bus-based communication, b) dedicated point-to-point links, c) a chip area network.

techniques to further increase the productivity of designers, is important. Electronic system level (ESL) design tools are necessary for supporting a design flow which make efficient use of such communication abstraction and design automation techniques, and which make for seamless iterations across all abstraction levels. Pertaining to this, the complex, dynamic interdependency of data streams – arising when using a shared media for data traffic – threatens to foil the efforts of obtaining minimal interdependence between IP cores. Without special quality-of-service (QoS) support, the performance of data communication may become unwarrantly arbitrary [Goossens et al. 2005].

To ensure the effective exploitation of technology scaling, intelligent use of the available chip design resources is necessary, at the physical as well as at the logical design level. Enabling means are the development of effective and structured design methods and ESL tools.

As seen above, the major driving factors for the development of global communication schemes are the ever increasing density of on-chip resources, and the drive to utilize these resources with a minimum of effort, as well as the need to counteract physical effects of DSM technologies. The trend is towards a subdivision of processing resources into manageable pieces. This helps reduce design cycle time since the entire chip design process can be divided into minimally interdependent subproblems. This also allows the use of modular verification methodologies, i.e. verification at low abstraction level of cores (and communication network) individually, and at high abstraction level of the system as a whole. Working at a high abstraction level allows a great degree of freedom from lower level issues. It also lends towards a differentiation of local and global communication. As inter-core communication is becoming the performance bottleneck in many multicore applications, the shift in design focus is from a traditional processing-centric to a communication-centric one. One top level aspect of this involves the possibility to save on global communication resources at the application level by introducing communication aware optimization algorithms in compilers [Guo et al. 2000]. System level effects of technology scaling are further discussed in [Catthoor et al. 2004].

A standardized global communication scheme, together with standard communication sockets for IP cores, would make Lego-brick-like plug-and-play design styles possible, allowing good use of the available resources and fast product design cycles.

Table I. Bus-versus-network arguments (adapted from [Guerrier and Greiner 2000]).

| Bus Pros & Cons | | | Network Pros & Cons |
|---|---|---|---|
| Every unit attached adds parasitic capacitance, therefore electrical performance degrades with growth. | - | + | Only point-to-point one-way wires are used, for all network sizes, thus local performance is not degraded when scaling. |
| Bus timing is difficult in a deep submicron process. | - | + | Network wires can be pipelined because links are point-to-point. |
| Bus arbitration can become a bottleneck. The arbitration delay grows with the number of masters. | - | + | Routing decisions are distributed, if the network protocol is made non-central. |
| The bus arbiter is instance-specific. | - | + | The same router may be reinstantiated, for all network sizes. |
| Bus testability is problematic and slow. | - | + | Locally placed dedicated BIST is fast and offers good test coverage. |
| Bandwidth is limited and shared by all units attached. | - | + | Aggregated bandwidth scales with the network size. |
| Bus latency is wire-speed once arbiter has granted control. | + | - | Internal network contention may cause a latency. |
| Any bus is almost directly compatible with most available IPs, including software running on CPUs. | + | - | Bus-oriented IPs need smart wrappers. Software needs clean synchronization in multiprocessor systems. |
| The concepts are simple and well understood. | + | - | System designers need reeducation for new concepts. |

## 1.2 NoC in SoC

Figure 3 show some examples of basic communication structures in a sample SoC, e.g. a mobile phone. Since the introduction of the SoC concept in the 90s, the solutions for SoC communication structures have generally been characterized by custom designed ad hoc mixes of buses and point-to-point links [Lahiri et al. 2001]. The bus builds on well understood concepts and is easy to model. In a highly interconnected multicore system however, it can quickly become a communication bottleneck. As more units are added to it, the power usage per communication event grows as well, due to more attached units leading to higher capacitive load. For multi-master busses, the problem of arbitration is also not trivial. Table I summarizes the pros and cons of buses and networks. A crossbar overcomes some of the limitations of the buses. However, it is not ultimately scalable and as such an intermediate solution. Dedicated point-to-point links are optimal in terms of bandwidth availability, latency and power usage, as they are designed especially for the given purpose. Also, they are simple to design and verify, and easy to model. But the number of links needed increases exponentially as the number of cores increases. Thus an area and possibly a routing problem develops.

From the point of view of design-effort one may argue that in small systems of less than 20 cores an ad hoc communication structure is viable. But as the systems grow and the design cycle time requirements decrease, the need for more generalized solutions becomes pressing. For maximum flexibility and scalability, it is generally accepted that a move towards a shared, segmented global communication structure is needed. This notion translates into a data-routing network consisting of communication links and routing nodes, being implemented on the chip. In contrast to traditional SoC communication methods outlined above, such a distributed communication media scales well with chip size and

complexity. Additional advantages include increased aggregated performance by exploiting parallel operation.

From a technological perspective, a similar solution is reached: in DSM chips, long wires must be segmented in order to avoid signal degradation, and busses are implemented as multiplexed structures in order to reduce power and increase responsiveness. Hierarchical bus structures are also common, as a means to adhere to the given communication requirements. The next natural step is to increase throughput by pipelining these structures. Wires become pipelines and bus-bridges become routing nodes. Expanding on a structure using these elements, one gets a simple network.

A common concept for segmented SoC communication structures is based on networks. This is what is known as Network-on-Chip (NoC) [Agarwal 1999][Guerrier and Greiner 2000][Dally and Towles 2001][Benini and Micheli 2002][Jantsch and Tenhunen 2003]. As seen above, the distinction between different communication solutions is fading. NoC is seen to be a unifying concept rather than an explicit new alternative. In the research community, there are two widely held perceptions of NoC: (i) NoC as a subset of SoC, and (ii) NoC as an extension of SoC. In the first view, NoC is defined strictly as the data-forwarding communication fabric, i.e. the network and methods used in accessing the network. In the second view NoC is defined more broadly, also to encompass issues dealing with the application, system architecture, and its impact on communication or vice versa.

### 1.3 Outline

The purpose of this survey is to clarify the NoC concept and to map the scientific efforts made into the area of NoC research. We will identify general trends, and explain a range of issues which are important for state-of-the-art global chip-level communication. In doing so we primarily take the first view of NoC, i.e. it being a subset of SoC, to focus and structure the diverse discussion. From our perspective, the view of NoC as an extension of SoC muddles the discussion with topics common to any large-scale IC design effort such as: partitioning and mapping application, hardware/software co-design, compiler choice, etc.

The rest of the survey is organized as follows. In Section 2 we will discuss the basics of NoC. We will give a simple NoC example, address some relevant system level architectural issues, and relate the basic building blocks of NoC to abstract network layers, and to research areas. In Section 3 we will go into more details of existing NoC research. This section is partitioned according to the research areas defined in Section 2. In Section 4 we discuss high abstraction level issues such as design space exploration and modeling. These are issues often applicable to NoC only in the view of it being an extension of SoC, but we treat specifically issues of relevance to NoC-based designs and not to large scale IC designs in general. In Section 5 performance analysis is addressed. Section 6 presents a set of case studies, describing a number of specific NoC implementations, and Section 7 summarizes the survey.

## 2. NOC BASICS

In this section the basics of NoC are uncovered. First a component based view will be presented, introducing the basic building blocks of a typical NoC. Then we shall look at system level architectural issues relevant to NoC-based SoC designs. After this, a layered abstraction based view will be presented, looking at network abstraction models, in particular OSI, and the adaption of such for NoC. Using the foundations established in this

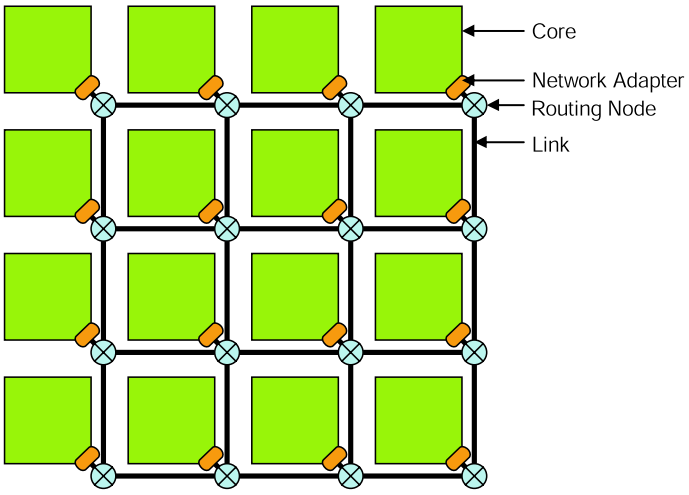



Fig. 4. Topological illustration of a 4-by-4 grid structured NoC, indicating the fundamental components.

section, we will go into further details of specific NoC research in Section 3.

### 2.1 A Simple NoC Example

Figure 4 shows a sample NoC structured as a 4-by-4 grid, which provides global chip-level communication. Instead of busses and dedicated point-to-point links, a more general scheme is adapted, employing a grid of *routing nodes* spread out across the chip, connected by communication *links*. For now we will adapt a simplified perspective in which the NoC contains the following fundamental components:

—**Network Adapters** implement the interface by which *cores* (IP blocks) connect to the NoC. Their function is to decouple computation (the cores) from communication (the network).

—**Routing Nodes** route the data according to chosen protocols. They implement the routing strategy.

—**Links** connect the nodes, providing the raw bandwidth. They may consist of one or more logical or physical channels.

Figure 4 covers only the topological aspects of the NoC. The NoC in the figure could thus employ packet or circuit switching or something entirely different, and be implemented using asynchronous, synchronous or other logic. In Section 3 we will go into details of specific issues with an impact on the network performance.

### 2.2 Architectural Issues

The diversity of communication in the network is affected by architectural issues such as system composition and clustering. These are general properties of SoC, but since they have direct influence on the design of the system level communication infrastructure we find it worthwhile to go through them here.

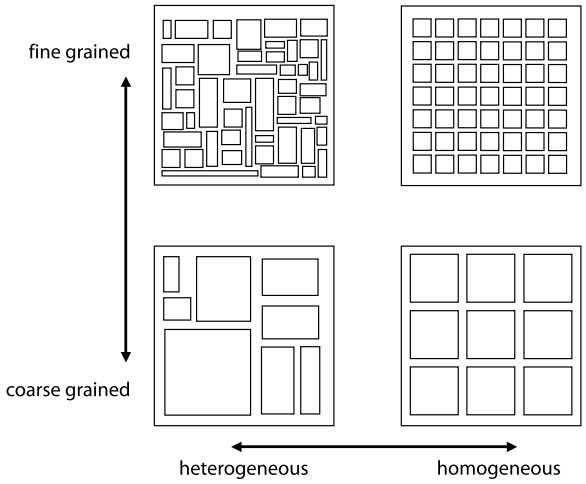



Fig. 5. System composition categorized along the axes of homogenity and granularity of system components.

Figure 5 illustrates how system composition may be categorized along the axes of *homogenity* and *granularity* of system cores. The figure also clarifies a basic difference between NoC and networks for more traditional parallel computers; the latter have generally been homogeneous and coarse grained, where as NoC-based systems implement a much higher degree of variety in composition, and in traffic diversity.

*Clustering* deals with the localization of portions of the system. Such localization may be logical or physical. Logical clustering can be a valuable programming tool. It can be supported by the implementation of hardware primitives in the network, e.g. flexible addressing schemes or virtual connections. Physical clustering, based on pre-existing knowledge of traffic patterns in the system, can be used to minimize global communication, thereby minimizing the total cost of communicating, power- and performance-wise.

Generally speaking, *reconfigurability* deals with the ability to allocate available resources for specific purposes. In relation to NoC-based systems, reconfigurability concerns how the NoC, a flexible communication structure, can be used to make the system reconfigurable from an application point of view. A configuration can be established e.g. by programming connections into the NoC. This resembles the reconfigurability of an FPGA, though NoC-based reconfigurability is most often of coarser granularity. In NoC, the reconfigurable resources are the routing nodes and links rather than wires.

Much research work has been done on architecturally oriented projects, in relation to NoC-based systems. The main issue in architectural decisions is the balancing of flexibility, performance and hardware costs of the system as a whole. As the underlying technology advances, the trade-off spectrum is continually shifted, and the viability of the NoC concept has opened up to a communication-centric solution space, which is what current system level research explores.

At one corner of the architecural space outlined in Figure 5, is the Pleiades architecture [Zhang et al. 2000] and its instantiation the Maia processor. A microprocessor is combined with a relatively fine grained heterogeneous collection of ALUs, memories, FP-

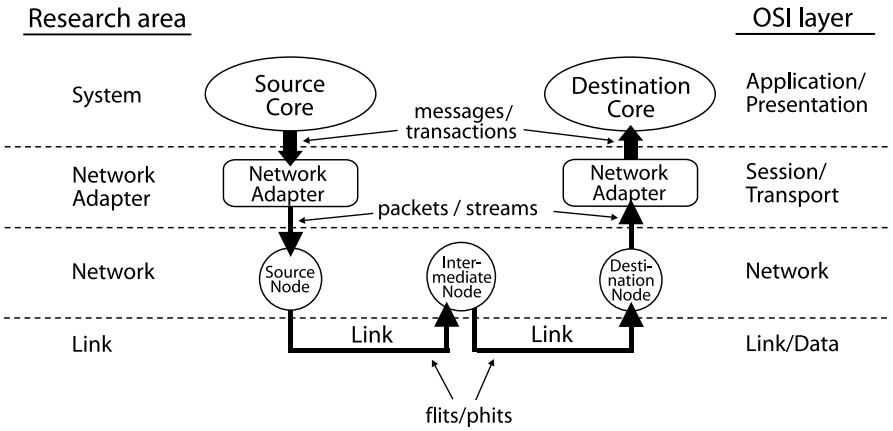



Fig. 6. The flow of data from source to sink, through the NoC components, with an indication of the types of datagrams and research area.

GAs, etc. An interconnection network allows arbitrary communication between modules of the system. The network is hierarchical and employs clustering in order to provide the required communication flexibility while maintaining good energy-efficiency.

At the opposite corner are a number of works, implementing homogeneous, coarse grained multiprocessors. In the Smart Memories [Mai et al. 2000] a hierarchical network is used, with physical clustering of four processors. The flexibility of the local cluster network is used as a means for reconfigurability, and the effectiveness of the platform is demonstrated by mimicking two machines on far ends of the architectural spectrum, the Imagine streaming processor and Hydra multiprocessor, with modest performance degradation. The global NoC is not described however. In the RAW architecture [Taylor et al. 2002] on the other hand, the NoC which interconnects the processor tiles is described in detail. It consists of a static network, in which the communication is preprogrammed cycle by cycle, and a dynamic network. The reason for implementing two physically separate networks is to accommodate different types of traffic in general purpose systems (see Section 4.3 concerning traffic characterization). The Eclipse [Forsell 2002] is another similarly distributed multiprocessor architecture, in which the interconnection network plays an important role. Here, the NoC is a key element in supporting a sofisticated parallel programming model.

## 2.3 Network Abstraction

The term NoC is used in research today in a very broad sense ranging from gate-level physical implementation, across system layout aspects and applications, to design methodologies and tools. A major reason for the wide-spread adaptation of network terminology lies in the readily available and widely accepted abstraction models for networked communication. The OSI model of layered network communication can easily be adapted for NoC usage, as done in [Benini and Micheli 2001] and [Arteris 2005]. In the following we will look at network abstraction, and make some definitions to be used later in the survey.

To better understand the approaches of different groups involved in NoC, we have par-

titioned the spectrum of NoC research into four areas: 1) System, 2) Network Adapter, 3) Network and 4) Link research. Figure 6 shows the flow of data through the network, indicating the relation between these research areas, the fundamental components of NoC and the OSI layers. Also indicated is the basic datagram terminology.

The *System* encompasses applications (processes) and architecture (cores and network). At this level, most of the network implementation details may still be hidden. Much research done at this level is applicable to large scale SoC design in general. The *Network Adapter* (NA) decouples the cores from the network. It handles the end-to-end flow control, encapsulating the *messages* or *transactions* generated by the cores for the routing strategy of the Network. These are broken into *packets* which contain information about their destination, or connection-oriented *streams* which do not, but have had a path setup prior to transmission. The NA is the first level which is 'network aware'. The *Network* consists of the routing nodes, links, etc, defining the topology and implementing the protocol and the node-to-node flow control. The lowest level is the *Link* level. At this level, the basic datagram are *flits* (**fl**ow control un**its**), node level atomic units from which packets and streams are made up. Some researchers operate with yet another subdivision, namely *phits* (**ph**ysical un**its**), which are the minimum size datagram that can be transmitted in one link transaction. Most commonly flits and phits are equivalent, though in a network employing highly serialized links, each flit could be made up of a sequence of phits. Link level research deals mostly with encoding and synchronization issues. The presented datagram terminology seems to be generally accepted, though no standard exists.

In a NoC, the layers are generally more closely bound than in a macro network. Issues arising often have a more physically related flavor, even at the higher abstraction levels. OSI specifies a protocol stack for multicomputer networks. Its aim is to shield higher levels of the network from issues of lower levels, in order to allow communication between independently developed systems, e.g. of different manufacturers, and to allow on-going expansion of systems. In comparison with macro networks, NoC benefits from the system composition being completely static. The network can be designed based on knowledge of the cores to be connected, and possibly also on knowledge of the characteristics of the traffic to be handled, as demonstrated in e.g. [Bolotin et al. 2004] and [Goossens et al. 2005]. Awareness of lower levels can be beneficial, as it can lead to higher performance. The OSI layers, which are defined mainly on a basis of pure abstraction of communication protocols, thus cannot be directly translated into the research areas defined here. With this in mind, the relation established in Figure 6 is to be taken as a conceptual guideline.

## 3. NOC RESEARCH

In this section we provide a review of the approaches of various research groups. Figure 7 illustrates a simplified classification of this research. The text is structured based on the layers defined in Section 2.3. Since we consider NoC as a subset of SoC, system level research is dealt with separately in Section 4.

### 3.1 Network Adapter

The purpose of the Network Adapter (NA) is to interface the core to the network, and make communication services transparently available with a minimum of effort from the core. At this point, the boundary between computation and communication is specified.

As illustrated in Figure 8, the NA component implements a Core Interface (CI) at the core side and a Network Interface (NI) at the network side. The function of the NA is

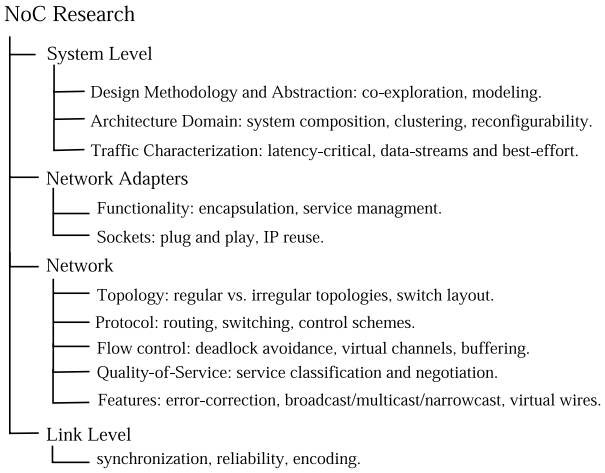



Fig. 7. NoC Research Area Classification. This classification, which also forms the structure of Section 3, is meant as a guideline to evaluate NoC research, and not as a technical categorization.

to provide high-level communication services to the core by utilizing primitive services provided by the network hardware. Thus the NA *decouples* the core from the network, implementing the network end-to-end flow control, facilitating a layered system design approach. The level of decoupling may vary. A high level of decoupling allows for easy reuse of cores. This makes possible a utilization of the exploding resources available to chip designers, and greater design productivity is achieved. On the other hand, a lower level of decoupling (a more network aware core) has the potential to make more optimal use of the network resources.

In this section, we first address the use of standard sockets. We then discuss the abstract functionality of the NA. Finally, we talk about some actual NA implementations, which also address issues related to timing and synchronization.

3.1.1 *Sockets.* The CI of the NA may be implemented to adhere to a SoC socket standard. The purpose of a socket is to orthogonalize computation and communication. Ideally a socket should be completely NoC implementation agnostic. This will facilitate the greatest degree of reusability, because the core adheres to the specification of the socket alone, independently of the underlying network hardware. One commonly used socket is the *Open Core Protocol* (OCP) [OCPIP 2003b][Haverinen et al. 2002]. The OCP specification defines a flexible family of memory-mapped, core-centric protocols for use as native core interface in on-chip systems. The three primary properties envisioned in OCP include: (i) architecture independent design reuse, (ii) feature specific socket implementation, and (iii) simplification of system verification and testing. OCP addresses not only data-flow signaling, but also uses related to errors, interrupts, flags and software flow control, control and status, and test. Another proposed standard is the *Virtual Component Interface* (VCI) [VSI Alliance 2000] used in the SPIN [Guerrier and Greiner 2000] and Proteo [Siguenza-Tortosa et al. 2004] NoCs. In [Radulescu et al. 2004] support for the Advanced eXtensible Interface (AXI) [ARM 2004] and Device Transaction Level (DTL) [Philips Semiconduc-

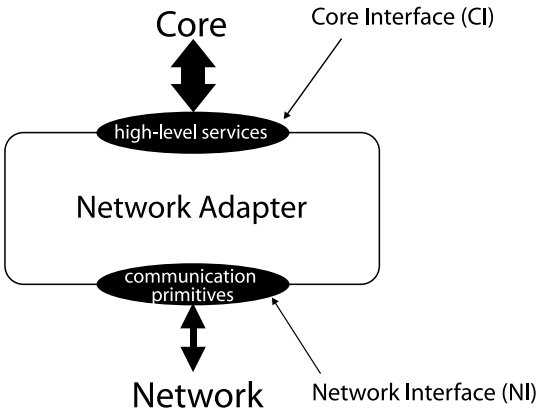



Fig. 8. The Network Adapter (NA) implements two interfaces, the Core Interface (CI) and the Network Interface (NI).

tors 2002] protocols was also implemented in an NA design.

3.1.2 *NA Services.* Basically, the NA provides *encapsulation* of the traffic for the underlying communication media and *management* of services provided by the network. Encapsulation involves handling of end-to-end flow control in the network. This may include global addressing and routing tasks, re-order buffering and data acknowledgement, buffer management to prevent network congestion, e.g. based on credits, packet creation in a packet-switched network, etc.

Cores will content for network resources. These may be provided in terms of service quantification, e.g. bandwidth and/or latency guarantees (see also Sections 3.2.4 and 5). Service management concerns setting up circuits in a circuit-switched network, book keeping tasks such as keeping track of connections, and matching responses to requests. Another task of the NA could be to negotiate the service needs between the core and the network.

3.1.3 *NA Implementations.* A clear understanding of the role of the NA is essential to successful NoC design. Muttersbach, Villiger and Fichtner [Muttersbach et al. 2000] address synchronization issues, proposing a design of an asynchronous wrapper for use in a practical GALS design. Here the synchronous modules are equipped with asynchronous wrappers which adapt their interfaces to the self-timed environment. The packetization occurs within the synchronous module. The wrappers are assembled from a concise library of pre-designed technology-independent elements and provide high speed data transfer. Another mixed asynchronous/synchronous NA architecture is proposed in [Bjerregaard et al. 2005]. Here, a synchronous OCP interface connects to an asynchronous, message-passing NoC. Packetization is performed in the synchronous domain, while sequencing of flits is done in the asynchronous domain. This makes the sequencing independent of the speed of the OCP interface, while still taking advantage of synthesized synchronous design, for maintaining a flexible packet format. Thus the NA leverages the advantages particular to either circuit design style. In [Radulescu et al. 2004] a complete NA design for the ÆTHEREAL NoC is presented, which also offers a shared-memory abstraction to

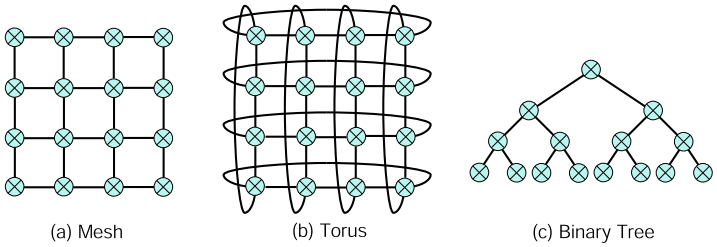



Fig. 9. Regular forms of topologies scale predictably with regards to area and power. Examples are (a) 4-ary 2-cube mesh, (b) 4-ary 2-cube torus and (c) binary (2-ary) tree.

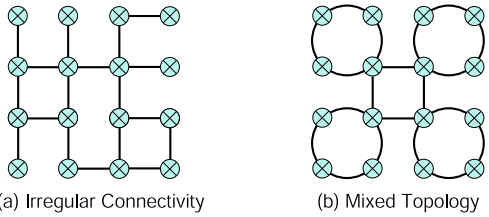



Fig. 10. Irregular forms of topologies are derived by altering the connectivity of a regular structure such as shown in (a) where certain links from a mesh have been removed, or by mixing different topologies such as in (b) where a ring co-exists with a mesh.

the cores. It provides compatibility to existing on-chip protocols such as AXI, DTL and OCP, and allows easy extension to other future protocols as well.

However, the cost of using standard sockets is not trivial. As demonstrated in the HERMES NoC [Ost et al. 2005], the introduction of OCP makes the transactions upto 50% slower compared to the native core interface. An interesting design trade-off issue is the partitioning of the NA functions between software (possibly in the core) and hardware (most often in the NA). In [Bhojwani and Mahapatra 2003] a comparison of software and hardware implementations of the packetization task was undertaken, the software taking 47 cycles to complete, while the hardware version taking only 2 cycles. In [Radulescu et al. 2004] a hardware implementation of the entire NA introduces a latency overhead of between 4 and 10 cycles, pipelined to maximize throughput. The NA in [Bjerregaard et al. 2005] takes advantage of the low forward latency of clockless circuit techniques, introducing an end-to-end latency overhead of only 3 to 5 cycles for writes and 6 to 8 cycles for reads, which include data return.

### 3.2 Network Level

The job of the network is to deliver messages from their source to their designated destination. This is done by providing the hardware support for basic communication primitives. A well-built network, as noted by Dally and Towles [Dally and Towles 2001], should appear as a logical wire to its clients. An on-chip network is defined mainly by its topology and the protocol implemented by it. Topology concerns the layout and connectivity of the nodes and links on the chip. Protocol dictates how these nodes and links are used.

3.2.1 *Topology.* One simple way to distinguish different regular topologies is in terms of *k-ary n-cube* (grid-type), where *k* is the degree of each dimension and *n* is the number of dimensions (Figure 9), first described by Dally [Dally 1990] for multicomputer networks. The *k-ary tree* and the *k-ary n-dimensional* fat tree are two alternate regular forms of networks explored for NoC. The network area and power consumption scales predictably for increasing size of regular forms of topology. Most NoCs implement regular forms of network topology, that can be laid out on a chip surface (a 2-dimensional plane) e.g. k-ary 2-cube, commonly known as grid-based topologies. The Octagon NoC demonstrated in [Karim et al. 2001][Karim et al. 2002] is an example of a novel regular NoC topology. Its basic configuration is a ring of 8 nodes connected by 12 bi-directional links, which provides two-hop communication between any pair of nodes in the ring, and a simple, shortest-path routing algorithm. Such rings are then connected edge to edge, to form a larger, scalable network. For more complex structures such as trees, finding the optimal layout is a challenge in its own right.

Besides the form, the nature of links adds an additional aspect to the topology. In k-ary 2-cube networks, popular NoC topologies based on the nature of link are: the mesh which uses bidirectional links, and torus using unidirectional links. For a torus, a folding can be employed to reduce long wires. In the NOSTRUM NoC presented in [Millberg et al. 2004] a folded torus is discarded in favor of a mesh, with the argument that it has longer delays between routing nodes. Figure 9 shows examples of regular forms of topology. Generally, mesh topology makes better use of links (utilization) while tree-based topologies are useful for exploiting locality of traffic.

Irregular forms of topologies are derived by mixing different forms, in a hierarchical, hybrid or asymmetric fashion, as seen in Figure 10. Irregular forms of topologies scale non-linearly with regards to area and power. These are usually based on the concept of clustering. A small private/local network often implemented as a bus, [Mai et al. 2000] and [Wielage and Goossens 2002], for local communication with k-ary 2-cube global communication is a favored solution. In [Pande et al. 2005], the impact of clustering on five NoC topologies is presented. It shows 20% to 40% reduction in bit-energy for the same amount of throughput, due to traffic localization.

With regards to the presence of a local traffic source or sink connected to the node, *direct networks* are those that have at least one core attached to each node, *indirect networks* on the other hand have a subset of nodes not connected to any core, performing only network operations; as is generally seen in tree-based topology where cores are connected at the leaf nodes. The examples of indirect tree-based networks are fat-tree in SPIN [Guerrier and Greiner 2000] and butterfly in [Pande et al. 2003]. The fat-tree used in SPIN is proven in [Leiserson 1985] to be most hardware efficient compared to any other network.

For alternate classifications of topology the reader is referred to [Aggarwal and Franklin 2002], [Jantsch 2003] and [Culler et al. 1998]. Culler in [Culler et al. 1998] combines protocol and geometry, to bring out a new type of classification which is defined as topology.

With regards to the routing nodes, a layout trade-off is the *thin switch* vs *square switch* presented by Kumar et al [Kumar et al. 2002]. Figure 11 illustrates the difference between these two layout concepts. A thin switch is distributed around the cores and wires are routed across them. A square switch is placed on the crossings of dedicated wiring channels between the cores. It was found that the square switch is better for performance and bandwidth while the thin switch requires relatively low area. The area overhead required to

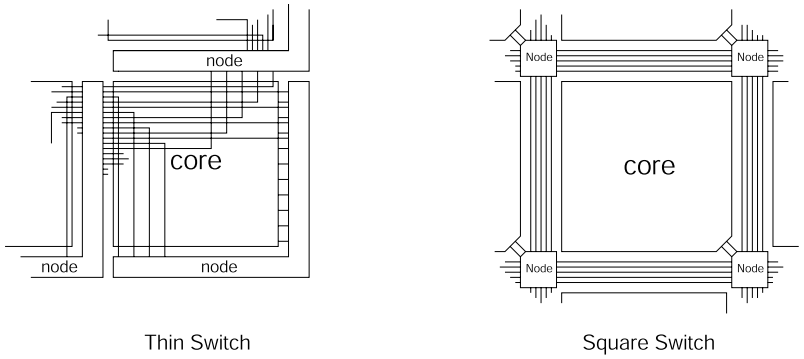



Fig. 11. Two layout concepts. The thin switch is distributed around the cores and wires are routed across it. The square switch is placed on the crossings in dedicated channels between the cores.

implement a NoC is in any case expected to be modest. The processing logic of the router, for a packet switched network, is estimated to be approximately between 2.0% [Pande et al. 2003] to 6.6% [Dally and Towles 2001] of the total chip area. In addition to this, the wiring uses a portion of the upper two wiring layers.

3.2.2 *Protocol.* The protocol concerns the strategy of moving data through the NoC. We define switching as the mere transport of data, while routing is the intelligence behind, i.e. it determines the path of the data transport. This is in accordance with Culler et al [Culler et al. 1998]. In the following these and other aspects of protocol commonly addressed in NoC research, are discussed:

—**Circuit vs packet switching:** Circuit switching involves the circuit from source to destination being setup and reserved until the transport of data is complete. Packet switched traffic on the other hand is forwarded on a per-hop basis, each packet containing routing information as well as data.

—**Connection-oriented vs connection-less:** Connection-oriented mechanisms involve a dedicated (logical) connection path being established prior to data transport. The connection is then terminated upon completion of communication. In connection-less mechanisms the communication occurs in a dynamic manner, with no prior arrangement between the sender and the receiver. Thus circuit switched communication is always connection-oriented, whereas packet switched communication may be either connection-oriented or connection-less.

—**Deterministic vs adaptive routing:** In a deterministic routing strategy, the traversal path is determined by its source and destination alone. Popular deterministic routing schemes for NoC are source routing and X-Y routing (2-d dimension order routing). In source routing, the source core specifies the route to the destination. In X-Y routing the packet follow the rows first then along the columns toward destination or vice versa. In an adaptive routing strategy the routing path is decided on a per-hop basis. Adaptive schemes involve dynamic arbitration mechanisms, e.g. based on local link congestion. This results in more complex node implementations, but offers benefits like dynamic load balancing.

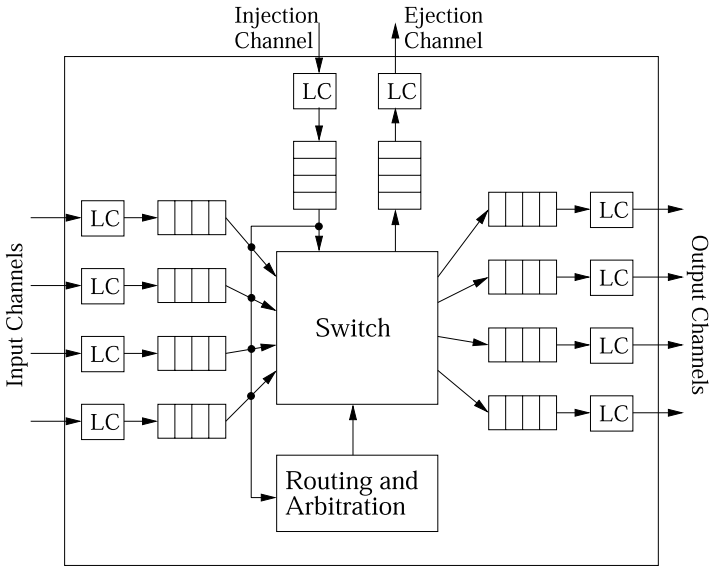



Fig. 12. Generic router model. LC = link controller (reprinted from [Duato et al. 2003] by Jose Duato, Sudhakar Yalamanchili and Lionel Ni, Fig. 2.1, ©2003, with permission from Elsevier).

—**Minimal vs non-minimal routing:** A routing algorithm is minimal if it always chooses among shortest paths toward the destination; otherwise it is non-minimal.

—**Delay vs loss:** In the delay model datagrams are never dropped. This means that the worst that can happen is the data being delayed. In the loss model datagrams can be dropped. In this case the data needs to be retransmitted. The loss model introduces some overhead in that the state of the transmission, successful or failed, must somehow be communicated back to the source. There are however some advantages involved in dropping datagrams, e.g. as a means of resolving network congestion.

—**Central vs distributed control:** In centralized control mechanisms, routing decisions are made globally, e.g. bus arbitration. In distributed control, most common for segmented interconnection networks, the routing decisions are made locally.

The protocol defines the use of the available resources, and thus the node implementation reflects design choices based on the above listed terms. In Figure 12, taken from [Duato et al. 2003], Duato et al. have clearly identified the major components of any routing node i.e.: buffers, switch, routing and arbitration unit and link controller. The switch connects the input buffers to the output buffers, while the routing and arbitration unit implements the algorithm that dictates these connections. In a centrally controlled system, the routing control would be common for all nodes, and a strategy might be chosen which guarantees no traffic contention. Thus no arbitration unit would be necessary. Such a scheme can be employed in a NoC in which all nodes have a common sense of time, as done in [Millberg et al. 2004]. Here the NOSTRUM NoC implements an explicit time division multiplexing mechanism which the authors call *Temporally Disjoint Networks* (TDN). Packets cannot collide, if they are in different TDNs. This is similar to the slot allocation mechanism in

the ÆTHEREAL NoC [Goossens et al. 2005].

The optimal design of the switching fabric itself relates to the services offered by the router. In [Kim et al. 2005] a crossbar switch is proposed, which offers adaptive bandwidth control. This is facilitated by adding an additional bus, allowing the crossbar to be bypassed during periods of congestion. Thus, the switch is shown to improve the throughput and latency of the router by up to 27% and 41% respectively, at a modest area and power overhead of 21% and 15% respectively. In [Bjerregaard and Sparsø 2005a] on the other hand, a non-blocking switch is proposed, which allows for hard performance guarantees, when switching connections within the router (more details in Section 3.2.4). By utilizing the knowledge that only a limited number of flits can enter the router through each input port, the switch can be made to scale linearly rather than exponentially, with the number of connections on each port. In [Leroy et al. 2005] a switch similarly provides guaranteed services. This switch however switches individual wires on each port, rather than virtual connections.

A quantitative comparison of connection-oriented and connection-less scheme for MPEG-2 Video Decoder is presented in [Harmanci et al. 2005]. The connection-oriented scheme is based on ÆTHEREAL, while the connection-less scheme is based on DiffServ - a priority based packet scheduling NoC. The conclusions of tests, conducted in the presence of background traffic noise, show that (i) the individual end-to-end delay is lower in connection-less than in connection-oriented one, due to better adaptation of the first approach to variable bit-rates of the MPEG video flows, and (ii) the connection-less schemes present a higher stability towards a wrong decision in the type of service to be assigned to a flow.

Concerning the merits of adaptive routing, versus deterministic, there are different opinions. In [Neeb et al. 2005] a comparison of deterministic (dimension-order) and adaptive (negative first and planar-adaptive) routing, applied to mesh, torus and cube networks, was made. For chips performing interleaving in high throughput channel decoder wireless applications, dimension-order routing scheme was found to be inferior compared to adaptive schemes, when using lower dimension NoCs topologies. However, it was shown to be the best-choice, due to low area and high thoughput characteristics, for higher dimension NoC topologies. The impact on area and throughput, of input and output buffer queues in the router, was also discussed. In [de Mello et al. 2004], the performance of minimal routing protocols in the HERMES [Moraes et al. 2004] NoC were investigated: one deterministic protocol (XY-routing) and three partially adaptive protocols (west-first, north-last and negative-first routing). While the adaptive protocols can potentially speed up the delivery of individual packets, it was shown that the deterministic protocol was superior to the adaptive ones from a global point. The reason is that adaptive protocols tend to concentrate the traffic in the center of the network, resulting in increased congestion here.

The wide majority of NoC research is based on packet switching networks. In addition, most are delay-based since the overhead of keeping account of packets being transmitted, and of retransmitting dropped packets is high. In [Gaughan et al. 1996] however, a routing scheme is presented which accomodates dropping packets when errors are detected. Most often connection-less routing is employed for best-effort (BE) traffic (Section 4.3), while connection-oriented routing is used to provide service guarantees (Section 3.2.4). In SoCBUS [Sathe et al. 2003] a different approach is taken, in that a connection-oriented strategy is used to provide BE traffic routing. Very simple routers establish short lived

Table II. Cost and stalling for different routing protocols.

| Protocol | Per router cost | | Stalling |
|---|---|---|---|
| | Latency | Buffering | |
| store-and-forward | packet | packet | at two nodes and the link between them |
| wormhole | header | header | at all nodes and links spanned by the packet |
| virtual cut-through | header | packet | at the local node |

connections, set up using BE routed control packets, which provide a very high throughput of 1.2 GHz in a 0.18 $\mu$m CMOS process. Drawbacks are the time spent during the setup phase, which requires a path acknowledge, and the fact that only a single connection can be active on each link at any given time. A similarly connection-oriented NoC is aSoC [Liang et al. 2000], which implements a small *reconfigurable communication processor* in each node. This processor has interconnect memory that programs the crossbar for data transfer from different sources across the node on each communication cycle.

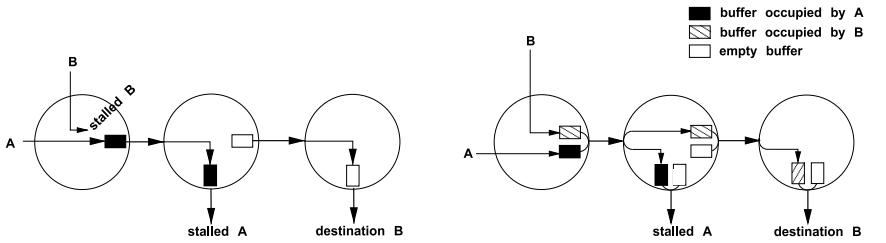
The most common forwarding strategies are store-and-forward, wormhole and virtual cut-through. These will be explained below. Table II summarizes the latency penalty and storage cost in each node for each of these schemes.

**Store-and-forward.** Store-and-forward routing is a packet switched protocol, in which the node stores the complete packet and forwards it based on the information within its header. Thus the packet may stall if the router in the forwarding path does not have sufficient buffer space. The CLICHE [Kumar et al. 2002] is an example of a store-and-forward NoC.

**Wormhole.** Wormhole routing combines packet switching with the data streaming quality of circuit switching, to attain a minimal packet latency. The node looks at the header of the packet to determine its next hop and immediately forwards it. The subsequent flits are forwarded as they arrive. This causes the packet to *worm* its way through the network, possibly spanning a number of nodes, hence the name. The latency within the router is not that of the whole packet. A stalling packet however has the unpleasantly expensive side effect of occupying all the links that the worm spans. In Section 3.2.3 we shall see how virtual channels can relieve this side effect at a marginal cost. In [Al-Tawil et al. 1997] a well structured survey of wormhole routing techniques is provided and a comparison between a number of schemes is made.

**Virtual cut-through.** Virtual cut-through routing has a forwarding mechanism similar to that of wormhole routing. But before forwarding the first flit of the packet, the node waits for a guarantee that the next node in the path will accept the entire packet. Thus if the packet stalls, it aggregates in the current node without blocking any links.

While macro networks usually employ store-and-forward routing, the prevailing scheme for NoC is wormhole routing. Advantages are low latency and the avoidance of area costly buffering queues. A special case of employing single flit packets is explored in [Dally and Towles 2001]. Here the data and header bits of the packets are transmitted separately and in parallel across a link, and the data path is quite wide (256 bits). Each flit is thus a packet in its own right, holding information about its destination. Hence, unlike wormhole routing, the stream of flits may be interlaced with other streams and stalling is restricted to the local node. Still single flit latency is achieved. The cost is a higher header-to-payload ratio, resulting in larger bandwidth overhead.




Incomming data stream B is stalled by stream A

Virtual channels allow stream B to pass stalled stream A

Fig. 13. Using virtual channels, independently buffered logical channels sharing a physical link, to prevent stalls in the network. Streams on different VCs can pass each other, while streams sharing buffer queues may stall.

3.2.3 *Flow Control.* Peh and Dally have defined flow control as the mechanism that determines the packet movement along the network path [Peh and Dally 2001]. Thus it encompasses both global and local issues. Flow control mainly addresses the issue of ensuring correct operation of the network. In addition, it can be extended to include also issues on utilizing network resources optimally and providing predictable performance of communication services. Flow control primitives thus also form the basis of differentiated communication services. This will be discussed further in Section 3.2.4

In the following, we first discuss the concept of virtual channels and their use in flow control. We then discuss a number of works in the area, and finally we address buffering issues.

**Virtual channels (VCs):** VCs is the sharing of a physical channel by several logically separate channels with individual and independent buffer queues. Generally, between 2 and 16 VCs per physical channel have been proposed for NoC. Their implementation results in an area and possibly also power, and latency overhead, due to the cost of control and buffer implementation. There are however a number of advantageous uses. Among these are:

—**Avoiding deadlocks.** Since VCs are not mutually dependent on each other, adding VCs to links, and choosing the routing scheme properly, one may break cycles in the resource dependency graph [Dally and Seitz 1987].

—**Optimizing wire utilization.** In future technologies, wire costs are projected to dominate over transistor costs [ITRS 2003]. Letting several logical channels share the physical wires, the wire utilization can be greatly increased. Advantages include reduced leakage power and wire routing congestion.

—**Improving performance.** VCs can generally be used to relax the inter-resource dependencies in the network, thus minimizing the frequency of stalls. In [Dally 1992] it is shown that dividing a fixed buffer size across a number of VCs improve the network performance at high loads. In [Duato and Pinkston 2001] the use of VCs to implement adaptive routing protocols is presented. [Vaidya et al. 2001] and [Cole et al. 2001] discusses the impact and benefit of supporting VCs.

—**Providing diffentiated services.** Quality-of-service (QoS, see Section 3.2.4) can be used as a tool to optimize application performance. VCs can be used to implement such services, by allowing high priority data streams to over take those of lower priority [Felicijan and Furber 2004][Rostislav et al. 2005][Beigne et al. 2005], or by providing guaranteed service levels on dedicated connections [Bjerregaard and Sparsø 2005a].

To ensure correct operation, the flow control of the network must first and foremost avoid deadlock and livelock. Deadlock occurs when network resources (e.g. link bandwidth or buffer space) are suspended waiting for each other to be released, i.e. where one path is blocked leading to other being blocked in a cyclic fashion [Dally and Seitz 1987]. It can be avoided by breaking cyclic dependencies in the resource dependency graph. Figure 13 illustrates how VCs can be used to prevent stalls due to dependencies on shared network resources. It is seen how in a network without VCs, stream B is stalled by stream A. In a network with VCs however, stream B is assigned to a different VC with a separate buffer queue. Thus even though stream A is stalled stream B is enabled to pass.

Livelock occurs when resources constantly change state waiting for other to finish. Livelock is less common but may be expected in networks where packets are reinjected into the network, or where back-stepping is allowed, e.g. during non-minimal adaptive routing.

Methods to avoid deadlock and livelock can be applied either locally at the nodes with support from service primitives e.g. implemented in hardware, or globally by ensuring logical separation of data streams by applying end-to-end control mechanisms. While local control is most widespread, the latter was done in [Millberg et al. 2004] using the concept of Temporally Disjoint Networks which was described in Section 3.2.2. As mentioned above, dimension-ordered routing is a popular choice for NoC, because it provides freedom from deadlock, without the need to introduce VCs. The *turn model* [Glass and Ni 1994] also does this, but allows more flexibility in routing. A related approach is the *odd-even turn model* [Chiu 2000] for designing partially adaptive deadlock-free routing algorithms. Unlike the turn model, which relies on prohibiting certain turns in order to achieve freedom from deadlock, this model restricts the *locations* where some types of turns can be taken. As a result, the degree of routing adaptiveness provided is more even for different source-destination pairs. The ANoC [Beigne et al. 2005] implements this routing scheme.

The work of José Duato has addressed the mathematical foundations of routing algorithms. His main interests have been in the area of adaptive routing algorithms for multicomputer networks. Most of the concepts are directly applicable to NoC. In [Duato 1993] the theoretical foundation for deadlock-free adaptive routing in wormhole networks is given. This builds on early work by Dally, which showed that avoiding cyclic dependencies in the channel dependency graph of a network, deadlock-free operation is assured. Duato expands the theory to allow adaptive routing, and furthermore shows that the absence of cyclic dependencies is too restrictive. It is enough to require the existence of a channel subset which defines a connected routing subfunction with no cycles in its *extended* channel dependency graph. The extended channel dependency graph is defined in [Duato 1993] as a graph for which the arcs are not only pairs of channels for which there is a direct dependency, but also pairs of channels for which there is an indirect dependency. In [Duato 1995] and [Duato 1996] this theory is refined and extended to cover also cut-through and store-and-forward routing. In [Duato and Pinkston 2001] a general theory is presented, which glues together several of the previously proposed theories into a single theoretical framework.

In [Dally and Aoki 1993] Dally has investigated a hybrid of adaptive and deterministic routing algorithms using VCs. Packets are routed adaptively until a certain number of hops have been made in a direction away from the destination. There after, the packets are routed deterministically, in order to be able to guarantee deadlock-free operation. Thus the benefits of adaptive routing schemes are approached, while keeping the simplicity and

predictability of deterministic schemes.

Other research has addressed flow control approaches purely for improving performance. In [Peh and Dally 1999] and [Kim et al. 2005] look-ahead arbitration schemes are used to allocate link and buffer access ahead of data arrival, thus reducing the end-to-end latency. This results in increased bandwidth utilization as well. Peh and Dally use virtual channels, and their approach is compared with simple virtual-channel flow control, as described in [Dally 1992]. It shows an improvement in latency of about 15%, across the entire spectrum of background traffic load, and network saturation occurs at a load 20% higher. Kim et al do not use virtual channels. Their approach is shown to improve latency considerably (by 42%) when network load is low (10%) with much less improvement (13%) when network load is high (50%). In [Mullins and Moore 2004] a virtual-channel router architecture for NoC is presented, which optimizes routing latency by hiding control overheads, in a single cycle implementation.

**Buffering:** Buffers are an integral part of any network router. In by far the most NoC architectures, buffers account for the main part of the router area. As such, it is a major concern to minimize the amount of buffering necessary, under given performance requirements. There are two main aspects of buffers: (i) their size and (ii) their location within the router. In [Kumar et al. 2002] it is shown that increasing the buffer size is not a solution towards avoiding congestion. At best, it delays the onset of congestion, since the throughput is not increased. The performance improved marginally in relation to the power and area overhead. On the other hand, buffers are useful to absorb bursty traffic, thus leveling the bursts.

Tamir and Frazier [Tamir and Frazier 1988] have provided an comprehensive overview of advantages and disadvantages of different buffer configurations (size and location) and additionally proposed a buffering strategy called dynamically allocated multi-queue (DAMQ) buffer. In the argument of input vs. output buffers, for equal performance the queue length in a system with output port buffering is always found to be shorter than the queue length in an equivalent system with input port buffering. This is so, since in a routing node with input buffers, a packet is blocked if it is queued behind a packet whose output port is busy (head-of-the-line-blocking). With regards to centralized buffer pools shared between multiple input and output ports vs distributed dedicated FIFOs, the centralized buffer implementations are found to be expensive in area due to overhead in control implementation, and become bottlenecks during periods of congestion. The DAMQ buffering scheme allows independent access to the packets destined for each output port, while applying its free space to any incoming packet. DAMQ shows better performance than FIFO or statically-allocated shared buffer space per input-output port due to better utilization of the available buffer space especially for non-uniform traffic. In [Rijpkema et al. 2001] a somewhat similar concept called virtual output queuing is explored. It combines moderate cost with high performance at the output queues. Here independent queues are designated to the output channels thus enhancing the link utilization by bypassing blocked packets.

In [Hu and Marculescu 2004a] the authors present an algorithm which sizes the (input) buffers in a mesh-type NoC, on basis of the traffic characteristics of a given application. For three audio/video benchmarks it was shown how such intelligent buffer allocation resulted in about 85% savings in buffering resources, in comparison with uniform buffer sizes, without any reduction in performance.

3.2.4 *Quality of Service (QoS).* QoS is defined as service quantification that is provided by the network to the demanding core. Thus it involves two aspects: (i) defining the services represented by a certain quantification and (ii) negotiating the services. The services could be low latency, high through-put, low power, bounds on jitter, etc. Negotiating implies balancing the service demands of the core with the services available from the network.

In [Jantsch and Tenhunen 2003](pgs: 61-82), Goossens et al characterize the nature of QoS in relation to NoC. They identify two basic QoS classes, best-effort services (BE) which offer no commitment, and guaranteed services (GS) which do. They also present different levels of commitment, and discuss their effect on predictability of the communication behavior: 1) *correctness* of the result, 2) *completion* of the transaction, 3) *bounds* on the performance. In [Rijpkema et al. 2001] argumentation for the necessity of a combination of BE and GS in NoC is provided. Basically, GS incur predictability, a quality which is often desirable e.g. in real-time systems, while BE improves the average resource utilization [Jantsch and Tenhunen 2003](pgs: 61-82)[Goossens et al. 2002][Rijpkema et al. 2003]. More details of the advantages of GS from a design flow and system verification perspective are given in [Goossens et al. 2005], in which a framework for the development of NoC-based SoC, using the ÆTHEREAL NoC, is described.

Strictly speaking, BE refers to communication for which no commitment can be given, whatsoever. In most NoC related works however, BE covers the traffic for which only correctness and completion are guaranteed while GS is traffic for which additional guarantees are given, i.e. on the performance of a transaction. In macro networks, service guarantees are often of a statistical nature. In tightly bound systems such as SoC, hard guarantees are often preferred. GS allows analytical system verification, and hence a true decoupling of sub-systems. In order to give hard guarantees, GS communication must be logically independent of other traffic in the system. This requires connection-oriented routing. Connections are instantiated as virtual circuits which use logically independent resources, thus avoiding contention. The virtual circuits can be implemented by either virtual channels, time-slots, parallel switch fabric, etc. As the complexity of the system increases and as GS requirements grow, so does the number of virtual circuits and resources (buffers, arbitration logic, etc) needed to sustain them.

While hard service guarantees provide an ultimate level of predictability, soft (statistical) GS or GS/BE hybrids have also been the focus of some research. In [Bolotin et al. 2004], [Felicijan and Furber 2004], [Beigne et al. 2005] and [Rostislav et al. 2005] NoCs providing prioritized BE traffic classes are presented. SoCBUS [Sathe et al. 2003] provides hard, short-lived GS connections, however since these are setup using BE routed packets, and torn down once used, this can also be categorized as soft GS.

ÆTHEREAL [Goossens et al. 2005], NOSTRUM [Millberg et al. 2004], MANGO [Bjerregaard and Sparsø 2005a], SONICS [Weber et al. 2005], aSOC [Liang et al. 2004], and also the NoCs presented in [Liu et al. 2004], in [Leroy et al. 2005], and the static NoC used in the RAW multiprocessor architecture [Taylor et al. 2002], are examples of NoCs implementing hard GS. While most NoCs that implement hard GS use variants of time division multiplexing (TDM) to implement connection-oriented packet routing, thus guaranteeing bandwidth on connections, the clockless NoC MANGO uses sequences of virtual channels to establish virtual end-to-end connections. Hence limitations of TDM, such as bandwidth and latency guarantees being inversely proportional, can be overcome

by appropriate scheduling. In [Bjerregaard and Sparsø 2005b] a scheme for guaranteeing latency, independently of bandwidth, is presented. In [Leroy et al. 2005] an approach for allocating individual wires on the link for different connections, is proposed. The authors call this *spatial division multiplexing*, as opposed to TDM.

For readers interested in exploitation of GS (in terms of throughput) virtual circuits during idle times, in [Andreasson and Kumar 2005] and [Andreasson and Kumar 2004] the concept of slack-time aware routing is introduced. A producer manages injection of BE packets during the slacks in time-slots reserved for GS packets, thereby mixing GS and BE traffic at the source which is unlike in other scheme discussed so far where it is done in the routers. In [Andreasson and Kumar 2005] the impact of variation of output buffer on BE latency is investigated, while in [Andreasson and Kumar 2004] the change of injection control mechanism for fixed buffer size is documented. QoS can also be handled by controlling the injection of packets into a BE network. In [Tortosa and Nurmi 2004] scheduling schemes for packet injection in a NoC with a ring topology were investigated. While a basic scheduling, which always farvors traffic already in the ring, provided the highest total bandwidth, weighted scheduling schemes were much more fair in their serving of different cores in the system.

In addition to the above, QoS may also cover special services such as:

—**Broadcast, multicast, narrowcast:** These features allow simultaneous communication from one source to all, i.e. broadcast, or select destinations, as is shown in ÆTHEREAL [Jantsch and Tenhunen 2003](pgs: 61-82), where a master can perform read or write operations on an address-space distributed among many slaves. In a connection oriented environment, the master request is channeled to a single slave for execution in narrowcast, while the master request is replicated for execution at all slaves in multicast. APIs are available within the NA to realize these types of transactions [Radulescu et al. 2004]. An alternate mulitcast implementation is discussed in [Millberg et al. 2004], where a virtual circuit meanders through all the destinations.

—**Virtual wires:** This refers to the use of network message-passing services to emulate direct pin-to-pin connection. In [Bjerregaard et al. 2005] such techniques are used to support a flexible interrupt scheme, in which the interrupt of a slave core can be programmed to trigger any master attached to the network, by sending a trigger packet.

—**Complex operations:** Complex functionality such as test-and-set, issued by a single command across the network, can be used to provide support for e.g. semaphores.

### 3.3 Link Level

Link level research regards the node-to-node links. These links consist of one or more channels, which can be either virtual or physical. In this section, we present a number of areas of interest for link level research: synchronization, implementation, reliability and encoding.

3.3.1 *Synchronization.* For link level synchronization in a multi clock domain SoC, Chelcea and Nowick have presented a mixed-time FIFO design [Chelcea and Nowick 2001]. The FIFO employs a ring of storage elements in which tokens are used to indicate full or empty state. This simplifies detection of the state of the FIFO (full or empty) and thus makes synchronization robust. In addition, the definitions of full and empty are extended so that full means 0 or 1 cell being unused, while empty means only 0 or 1 cells

being used. This helps in hiding the synchronization delay introduced between the state detection and the input/output handshaking. The FIFO design introduced can be made arbitrarily robust, with regards to metastability, as settling time and latency can be traded-off.

With the emerging of the GALS concept of *globally asynchronous locally synchronous* systems [Chapiro 1984][Meincke et al. 1999], implementing links using asynchronous circuit techniques [Sparsø and Furber 2001][Hauck 1995] is an obvious possibility. A major advantage of asynchronous design styles, relevant for NoC, is the fact that apart from leakage, no power is consumed when the links are idle. Thus, the design style addresses also the problematic issue of increasing power usage by large chips. Another advantage is the potentially very low forward latency, in uncongested data paths, leading to direct performance benefits. Examples of NoCs based on asynchronous circuit techniques are CHAIN [Bainbridge and Furber 2002][Amde et al. 2005], MANGO [Bjerregaard and Sparsø 2005a], ANoC [Beigne et al. 2005], and QNoC [Rostislav et al. 2005]. Asynchronous logic incorporates some area and dynamic power overhead compared with synchronous logic, due to local handshake control. The 1-of-4 encodings discussed in Section 3.3.4 below, generalized to 1-of-N, is much used in asynchronous links [Bainbridge and Furber 2001].

On the other hand, resynchronization of an incoming asynchronous transmission is also not trivial. It costs both time and power, and bit errors may be introduced. In [Dobkin et al. 2004], resynchronization techniques are described, and a method for achieving high throughput across an asynchronous to synchronous boundary is proposed. The work is based on the use of stoppable clocks, a scheme in which the clock of a core is stopped while receiving data on an asynchronous input port. Limitations to this technique are discussed, and the proposed method involves only the clock on the input register being controlled. In [Ginosaur 2003] a number of synchronization techniques are reviewed, and the pitfalls of the topic are addressed.

The trade-offs in the choice of synchronization scheme in a globally asynchronous or multiclocked system, is sensitive to the latency requirements of the system, the expected network load during normal usage, the node complexity, etc.

3.3.2 *Implementation issues.* As chip technologies scale into the DSM domain, the effect of wires on link delays and power consumption increase. Aspects and effects on wires of technology scaling are presented in [Ho et al. 2001], [Lee 1998], [Havemann and Hutchby 2001] and [Sylvester and Keutzer 2000]. In [Liu et al. 2004] these issues are covered specifically from a NoC point-of-view, projecting the operating frequency and size of IP cores in NoC-based SoC designs for future CMOS technologies, down to 0.05 $\mu$m. In the following, we will discuss a number of physical level issues relevant to the implementation of on-chip links.

**Wire segmentation.** At the physical level, the challenge lies in designing fast, reliable and low power point-to-point interconnects, ranging across long distances. Since the delay of long on-chip wires is characterized by distributed RC charging, it has been standard procedure for some time to apply segmentation of long wires by inserting *repeater* buffers at regular intervals, in order to keep the delay linearly dependent on the length of the wire. In [Dobbelaere et al. 1995] an alternative type of repeater is proposed. Rather than splitting and inserting a buffer in the path of the wire, it is based on a scheme of sensing and pulling the wire using a keeper device attached to the wire. The method is shown to improve the delay of global wires by up to 2 times compared with conventional repeaters.

**Pipelining.** Partitioning long interconnects into pipeline stages, as an alternative to wire

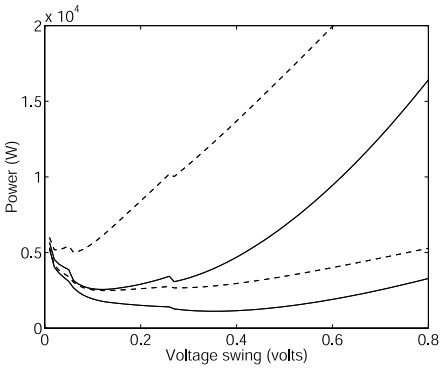



Fig. 14. Total power versus voltage swing for long (5-10 mm) on-chip interconnect. Solid line case 1: power supply generated off-chip by high efficiency DC-DC converter. Dashed line case 2: power supply generated internally on-chip. Upper curves for data activity of 0.25, lower curves 0.05 (reprinted from [Svensson 2001] Fig. 2, ©2001, with permission from Christer Svensson).

segmentation, is an effective way of increasing throughput. The flow control handshake loop is shorter in a pipelined link, making the critical loop faster. This is at the expense of latency of the link and circuit area, since pipeline stages are more complex than repeater buffers. But the forward latency in an asynchronous pipeline handshake cycle can be minimized to a few gate delays, so as wire effects begin to dominate performance in DSM technologies, the overhead of pipelining as opposed to buffering will dwindle. In [Singh and Nowick 2000] several high-throughput clockless pipeline designs were implemented using dynamic logic. Completion detection was employed at each stage to generate acknowledge signals, which were then used to control the precharging and evaluation of the dynamic nodes. The result was a very high throughput of up to 1.2 GDI/s (giga data items per second) for single rail designs, in a 0.6 $\mu$m CMOS technology. In [Mizuno et al. 2001] a hybrid of wire segmentation and pipelining was shown, in that a channel was made with segmentation buffers implemented as latches. A congestion signal traveling backwards through the channel compresses the data in the channel, storing it in the latches, until the congestion is resolved. Thus a back pressure flow control scheme was employed, without the cost of full pipeline elements.

**Low swing drivers.** In an RC charging system, the power consumption is proportional to the voltage shift squared. One way of lowering the power consumption for long on-chip interconnects is by applying low-swing signaling techniques, which are also widely used for off-chip communication lines. Such are presented and analyzed in [Zhang et al. 1999]. Basically the power usage is lowered at the cost of the noise margin. However, a differential transmission line (2 wires), on which the voltage swing is half that of a given single-ended transmission line, has differential mode noise characteristics comparable to the single-ended version. This is so, because the voltage difference between the two wires is the same as that between the single-ended wire and a mid-point between supply and ground. As an approximation, it uses only half the power however, since the two wires working at half the swing each consume one-fourth the power. The common mode noise immunity of the differential version is also greatly improved, and it is thus less sensitive to crosstalk and ground bounces, important sources of noise in on-chip environments as

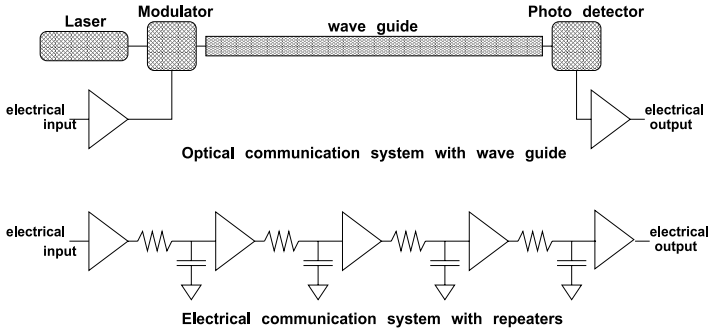



Fig. 15. Model of electrical and optical signaling systems for on-chip communication, showing the basic differences.

discussed in the reliability section below. In [Ho et al. 2003] the design of a low-swing, differential on-chip interconnect for the Smart Memories [Mai et al. 2000] is presented and validated with a test chip.

In [Svensson 2001] Svensson demonstrated how an optimum voltage swing for minimum power consumption in on- and off-chip interconnects can be found for a given data activity rate. The work takes into account dynamic and static power consumption of driving the wire, as well as in the receiver, which needs to amplify the signal back to full logic level. Calculations are presented for a 0.18 $\mu$m CMOS technology. Figure 14 displays the power consumption versus voltage swing for a global on-chip wire of 5-10 mm, a power supply of 1.3 V, and a clock frequency of 1 GHz. For a data activity rate of 0.25 (random data) it is seen that there is a minimum at 0.12 V. This minimum occurs for a two stage receiver amplifier and corresponds to a power saving of 17x. Using a single stage amplifier in the receiver, there is a minimum at 0.26 V, corresponding to a power saving of 14x.

**Future issues.** In [Heiliger et al. 1997] the use of microstrip transmission lines as waveguides for sub-mm wave on-chip interconnects is analyzed. It is shown that using $SiO_2$ as dielectric exhibit prohibitively large attenuation. However the use of bisbenzocyclobutene-polymer offers favorable line-parameters, with an almost dispersion free behavior at moderate attenuation (=< 1 dB/mm at 100 GHz). In [Kapur and Saraswat 2003] a comparison between electrical and optical interconnects for on-chip signaling and clock distribution is presented. Figure 15 shows the models used in evaluating optical and electrical communication. The delay vs. power and delay vs. interconnect length tradeoffs are analyzed for the two types of signaling. As seen in Figure 16, it is shown that the critical length above which the optical system is faster than the electrical is approximately 3-5 mm, projected for a 50 nm CMOS fabrication technology with copper wiring. The work also shows that for long interconnects (defined as 10 mm and above) the optical communication has a great potential for low power operation. Thus it is projected to be of great use in future clock distribution and global signaling.

3.3.3 *Reliability.* Designing global interconnects in DSM technologies, a number of communication reliability issues become relevant. Noise sources which can have an influence on this are mainly crosstalk, power supply noise such as ground bounce, electromagnetic interference (EMI), and intersymbol interference.

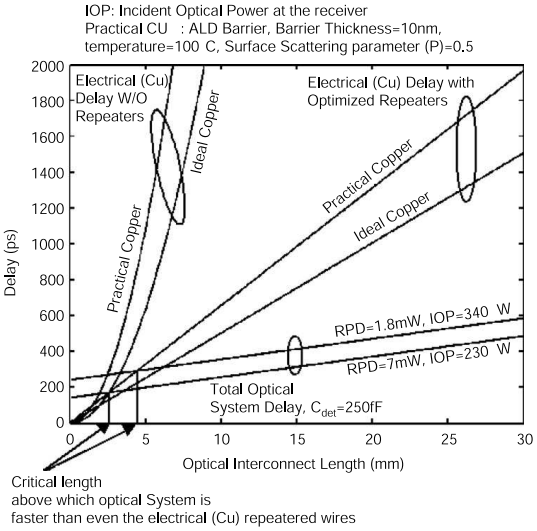



Fig. 16. Delay comparison of optical and electrical interconnect (with and without repeaters) in a projected 50 nm technology (reprinted from [Kapur and Saraswat 2003] by Pawan Kapur and Krishna C. Saraswat, Fig. 13, ©2002, with permission from Elsevier).

Crosstalk is becoming a serious issue due to decreasing supply voltage, increasing wire to wire capacitance, increasing wire inductance (e.g. in power supply lines), and increasing rise times of signaling wires. The wire length at which the peak crosstalk voltage is 10% of the supply voltage, decreases drastically with technology scaling [Jantsch and Tenhunen 2003](chapter 6), and since the length of global interconnects does not scale with technology scaling, this issue is especially relevant to the implementation of NoC links. Power supply noise is worsened by the inductance in the package bonding wires, and the insufficient capacitance in the on-chip power grid. The effect of EMI is worsening as the electric charges moved by each operation in the circuit is getting smaller, making it more susceptible to external influence. Intersymbol interference, i.e. the interference of one symbol on the following symbol on the same wire, is increasing as circuit speeds go up.

In [Jantsch and Tenhunen 2003](chapter 6), the Bertozzi and Benini present and analyze a number of error detecting/correcting encoding schemes in relation to NoC link implementation. Error recovery is a very important issue, since an error in i.e. the header of a packet, may lead to deadlock in the NoC, blocking the operation of the entire chip. This is also recognized in [Zimmer and Jantsch 2003] in which a fault model notation is proposed which can represent multi-wire and multi-cycle faults. This is interesting due to the fact that crosstalk in DSM busses can cause errors across a range of adjacent bits. It is shown that by splitting a wide bus into separate error detection bundles, and interleaving these, the error rate after using single-error correcting and double-error detecting codes can be reduced by several orders of a magnitude. This is because these error-correction schemes function properly when only respectively one or two errors occur in each bundle. When the bundles are interleaved, the probability of multiple errors within the same bundle is greatly reduced.

In [Gaughan et al. 1996] Gaughan et al deal with dynamically occurring errors in networks with faulty links. Their focus is on routing algorithms that can accommodate such errors, assuming that support for the detection of the errors is implemented. For wormhole routing, they present a scheme in which a data transmission is terminated upon detection of an error. A *kill* flit is transmitted backwards, deleting the worm and telling the sender to retransmit it. This naturally presents an overhead, and is not generally representative for excising NoC implementations. It can however prove necessary in mission critical systems. The paper provides formal mathematical proofs of deadlock-freedom.

Another issue with new CMOS technologies is the fact that the delay distribution – due to process variations – flattens with each new generation. While typical delay improves, worst-case delay barely changes. This presents a major problem in todays design methodologies, as these are mostly based on worst-case assumptions. Self-calibrating methods, as used in [Worm et al. 2005], are a way of dealing with unreliability issues of this character. The paper presents a self-calibrating link, and the problem of adaptively controlling its voltage and frequency. The object is to maintain acceptable design trade-offs between power consumption, performance and reliability, when designing on-chip communication systems using DSM technologies.

Redundant transmission of messages in the network is also a way of dealing with fabrication faults. In [Pirretti et al. 2004], two different flooding algorithms and a random walk algorithm are compared. It is shown that the flooding algorithms have an exceedingly large communication overhead, while the random walk offers reduced overhead while maintaining useful levels of fault tolerance.

With the aim of improving fabrication yield, Dally and Towles propose extra wires between nodes, so that defective wires found during post production tests or during self-test at start-up can be bypassed [Dally and Towles 2001]. Another potential advantage of a distributed shared communication structure is the possibility of bypassing entire regions of a chip, if fabrication faults are found.

Dynamic errors are more likely in long wires, and segmenting links into pipeline stages helps keeping the error rate down and the transmission speed up. Since segmentation of the communication infrastructure is one of the core concepts of NoC, it inherently provides solutions to the reliability problems. The segmentation is made possible because NoC-based systems generally imply the use of programming models allowing some degree of latency insensitive communication. Thus it is seen how the issues and solutions at the physical level relate directly to issues and solutions at system level, and vice versa. Another solution towards avoiding dynamic errors is the shielding of signal wires, e.g. by ground wires. This helps to minimize crosstalk from locally interfering wires, at the expense of wiring area.

3.3.4 *Encoding.* Using encoding for on-chip communication has been proposed, the most common objective being to reduce power usage per communicated bit, while maintaining high speed and good noise margin. In [Bogliolo 2001] the proposed encoding techniques are categorized as speed-enhancing or low-power encodings, and it is shown how different schemes in these two categories can be combined to gain the benefits of both. In [Nakamura and Horowitz 1996] a very simple low-weight coding technique was used to reduce dI/dt noise due to simultaneous switching of off-chip I/O drivers. An 8-bit signal was simply converted to a 9-bit signal, the 9th bit indicating whether the other 8 bits should be inverted. The density of 1's was thus reduced, resulting in a reduction

of switching noise by 50% and of power consumption by 18%. Similar techniques could prove useful in relation to long on-chip interconnects. The abundant wire resources available on-chip can also be used to implement more complex M-of-N encodings, thus trading wires for power. A widely used technique, especially in asynchronous implementations, is 1-of-4 encoding. This results in a good power/area tradeoff and low encoding/decoding overhead [Bainbridge and Furber 2001][Bainbridge and Furber 2002].

Another area of encoding, also discussed in Section 3.3.3, relates to error management. This involves the detection and correction of errors that may occur in the network. The mechanism may be observed at different layers of the network, and thus be applicable to either phits, flits, packets or messages. With regards to NoC, the interesting issues involve errors in the links connecting the nodes, since long wires of deep submicron technologies may exhibit unreliable behavior (see Section 3.3.3). ×pipes [Osso et al. 2003] implements flit-level CRC mechanism running in parallel with switching (thus masking its delay) to detect errors. Another common technique is parity-checks. The need here is to balance complexity of error-correction circuits to the urgency of such mechanism.

An interesting result is obtained in [Jantsch and Vitkowski 2005], wherein the authors investigate power consumption in the NOSTRUM NoC. Results are based on a 0.18 $\mu$m implementation, and scaled down to 65 nm. The paper concludes that the major part of the power is spent in the link wires. Power saving encoding however reduces performance, and simply scaling the supply voltage to normalize performance – in non-encoded links – actually results in better power figures than any of the encoding schemes investigated. Subsequently, the authors propose the use of end-to-end data protection, through error correction methods, which allows voltage scaling while maintaining the fault probability without lowering the link speed. In effect this results in better power figures.

In this section we have discussed issues relevant to the lowest level of the NoC, the link level. This concludes the discussion of network design and implementation topics. In the following section we discuss NoC from the view of design approach and modeling, in relation to SoC.

## 4. NOC MODELING

NoC, described as a subset of SoC, is an integral part of SoC design methodology and architecture. Given the vast design space and implementation decisions involved in NoC design, modeling and simulation is important to design flow, integration and verification of NoC concepts. In this section, first we discuss issues related to NoC modeling and then we explore design methodology used to study the system-level impact of the NoC. Finally traffic characterization, which bridges system-level dynamics with NoC requirements is discussed.

### 4.1 Modeling

Modeling the NoC in abstract software models is the first means to approach and understand the required NoC architecture and the traffic within it. Conceptually the purpose of NoC modeling is (i) to explore the vast design and feature space, and (ii) to evaluate tradeoffs between power, area, design-time, etc; while adhering to application requirement on one side and technology constraints on the other side. Modeling NoC has three intertwined aspects: modeling environment, abstraction levels, and result analysis. In the modeling environment section, we present three frameworks to describe NoC. Section 4.1.2 discusses

work done across different levels of NoC abstraction. The result analysis deals with a wide range of issues and is hence dealt with separately in Section 5.

4.1.1 *Modeling Environment.* The NoC models are either analytical or simulation based and can model communication across abstractions.

In a purely abstract framework, a NoC model using allocators, scheduler, and synchronizer is presented in [Madsen et al. 2003] and [Mahadevan et al. 2005]. The allocator translates the path traversal requirements of the message in terms of its resource requirements such as bandwidth, buffers, etc. It attempts to minimize resource conflicts. The scheduler executes the message transfer according to the particular network service requirements. It attempts to minimize resource occupancy. A synchronizer models the dependencies among communicating messages allowing concurrency. Thus these three components are well suited to describe a wide variety of NoC architecture and can be simulated in a multiprocessor real-time environment.

OPNET, a commercial network simulator originally developed for macro-networks, is used as NoC simulator in [Bolotin et al. 2004][Xu et al. 2004][Xu et al. 2005]. OPNET provides a convenient tool for hierarchical modeling of a network, including processes (state machines), network topology description and simulation of different traffic scenarios. However, as noted in [Xu et al. 2004] and [Xu et al. 2005], it needs to be adapted for synchronous environments, requiring explicit design of clocking scheme and a distribution network. [Bolotin et al. 2004] uses OPNET to model a QoS-based NoC architecture and design with irregular network topology.

A VHDL based cycle accurate RTL model for evaluating power and performance of NoC architecture is presented in [Banerjee et al. 2004]. The power and delay are evaluated for fine-grain components of the routers and links using SPICE simulations for a $0.18\mu$m technology and incorporated into the architectural-level blocks. Such modeling enables easy evaluation of dynamic vs leakage power at system-level. As expected, at high injection rate (packets/cycle/node) it was found that dynamic power dominates over leakage power. The Orion power-performance simulator proposed by Wang et al. [Wang et al. 2002], modeled only dynamic power consumption.

Recently, due to increasing size of applications, NoC emulation [Genko et al. 2005] has been proposed as alternative to simulation-based NoC models. It has been shown that FPGA based emulation can take few seconds compared to simulation-based approaches which can take hours to process through many millions of cycles, as would be necessary in any thorough communication co-exploration.

4.1.2 *Noc Modeling at Different Abstraction Levels.* New hardware description languages are emerging, such as SystemC [SystemC 2002], a library of C++, and SystemVerilog [Fitzpatrick 2004], which make simulations at a broad range of abstraction levels readily available, and thus support the full range of abstractions needed in a modular NoC-based SoC design. In [Bjerregaard et al. 2004] mixed-mode asynchronous handshake channels were developed in SystemC, and a mixed abstraction level design flow was used to design two different NoC topologies.

From an architectural point of view, the network topology generally incur the use of a segmented (multi-hop) communication structure, however some researchers working at the highest levels of abstraction, define NoC merely as a multiport blackbox communication structure or core, presenting a number of ports for communication. A message can

Table III. Communication semantics and abstraction for NoC, adapted from [Gerstlauer 2003].

| Layer | Interface semantics | Communication |
|---|---|---|
| Application/ Presentation | IP-to-IP messaging<br>`sys.send(struct myData)`<br>`sys.receive(struct myData)` | Message passing |
| Session/ Transport | IP-to-IP port-oriented messaging<br>`nwk.read(messagepointer*, unsigned len)`<br>`nwk.write(int addr, msgptr*, unsigned len)` | Message passing or shared memory |
| Network | NA-to-NA packet streams<br>`ctrl.send(), ctrl.receive()`<br>`link.read(bit[] path, bit[] data_packet)`<br>`link.write(bit[] path, bit[] data_packet)` | Message passing or shared memory |
| Link | Node-to-Node logical links and shared byte streams<br>`ctrl.send(), ctrl.receive()`<br>`channel.transmit(bit[] link, bit[] data_flit)`<br>`channel.receive(bit[] link, bit[] data_flit)` | Message passing |
| Physical | Pins and wires<br>`A.drive(0), D.sample(), clk.tick()` | Interconnect |

be transmitted from an arbitrary port to any other, allowing maximum flexibility of system communication. At this level, the actual implementation of the NoC is often not considered. Working at this high abstraction level allows a great degree of freedom from lower level issues. Table III adapted from Gerstlauer [Gerstlauer 2003] summarizes, in general, the communication primitives at different levels of abstraction.

At system level, transaction level models (TLM) are typically used for modeling communication behavior. This takes the form of either synchronous or asynchronous *send()/ receive()* message passing semantics, which use unique channels for communication between the source and the destination. One level below this abstraction, for NoCs, additional identifiers such as addressing may be needed to uniquely identify the traversal path or for providing services for end-to-end communication. Control primitives at network and link level, which are representative of actual hardware implementation, model the NoC flow-control mechanisms. In [Gerstlauer 2003], a JPEG encoder and voice encoder/decoder running concurrently were modeled for each and for mixed-levels of abstraction. The results show that the model complexity generally grows exponentially with lower level of abstraction. By extrapolating the result from bus to NoC, interestingly, model complexity at NA level can be found to be higher than at other levels due to slicing of message, connection management, buffer management, and others.

Working between session to network layer, Juurlink and Wijshoff [Juurlink and Wijshoff 1998] have made a comparison of three communication models used in parallel computation; (i) asynchronous communication with fixed message size, (ii) synchronous communication which rewards burst-mode message transfers, and (iii) asynchronous with variable message size communication while also accounting for network load. Cost-benefit analysis shows that, though the software-based messaging layers serve as a very useful function of delinking computation and communication, it creates anywhere between 25% to 200% overhead, as opposed to optimized hardware implementation.

A similar study of parallel computation applications, but with more detailed network model, was undertaken by Vaidya et al. [Vaidya et al. 2001]. Here the network was imple-

mented to use adaptive routing with virtual channels. The applications, running on power-of-two number of processors using grid based network topologies, used shared-memory or message passing for communication thus generating wide range of traffic patterns. They have found that increasing the number of VCs and routing adaptively offer little performance improvement for scalable shared-memory applications. Their observation holds true over a range of systems and problem sizes. The results show that the single most important factor for improving performance in such applications is the router speed, which is likely to provide lasting payoffs. The benefits of a faster router are visible across all applications in a consistent and predictable fashion.

Ahonen et al. [Ahonen et al. 2004] and Lahiri et al. [Lahiri et al. 2001] have associated high level modeling aspects with actual design choices such as: selection of an appropriate topology, selection of communication protocols, specification of architectural parameters (such as bus widths, burst transfer size, priorities, etc), and mapping communications onto the architecture, as requirements to optimize the on-chip communication for application-specific needs. Using a tool called OIDIPUS, Ahonen et al. compare placement of twelve processors in a ring-based topology. It is found that OIDIPUS, which uses physical path taken by the communication channels as cost function, generated topologies are only marginally inferior to human design. Without being restricting to any one topology, Lahiri et al. have evaluated traffic characteristics in a static priority based shared bus, hierarchical bus, two-level time division multiplexed access (TDMA), and ring based communication architecture. It was found that no single architecture uniformly outperforms other.

Wieferink [Wieferink et al. 2004] have demonstrated a processor/communication co-exploration methodology which works cross-abstraction and in a co-simulation platform. Here LISA based IP core descriptions have been integrated with SystemC based bus based transaction level models. A wide range of APIs are then provided to allow modeling between LISA and SystemC models, to allow instruction accurate model to co-exist with cycle accurate model, and TLM with RTL models. MPARM [Loghi et al. 2004] is a similar cycle-accurate and SystemC co-exploration platform, used in exploration of AMBA, STBus and ×pipes NoC evaluation.

### 4.2 Design and Co-Exploration Methodology

The NoC components, as described in Section 2.1, lends itself to flexible NoC designs such as parameterizable singular IP core, or malleable building-blocks, customizable at the network-layer, for design and reuse into application-specific NoC. A SoC design methodology requiring a communication infrastructure, can exploit either characteristics to suit the application's needs. Keeping this in mind, different NoC researchers have uniquely tailored their NoC architectures. Figure 17 shows our assessment of instance-specific capability of these NoC architectures. The two axis are explained as follows.

—**Parametrizability at system-level:** By this, we mean the ease with which a system-level NoC characteristic can be changed at instantiation time. The NoC description may encompass a wide range of parameters, such as: number of slots in the switch, pipeline-stages in the links, number of ports of the network and others. This is very useful for co-exploration directly with IP cores of the SoC.

—**Granularity of NoC:** By granularity, we mean at what level the NoC or NoC components is described. At the coarser end, the NoC may described as a single core, while at

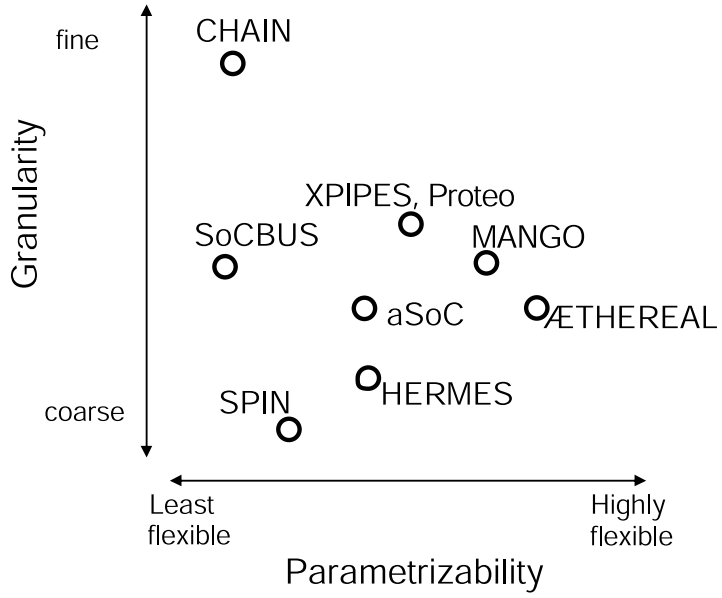



Fig. 17. NoC Instantiation Space

other end of the spectrum, the NoC may be assembled from lower-level blocks.

Consider the example of CHAIN [Bainbridge and Furber 2002]. It provides a library of fine-grained NoC components. Using these components, a NoC designer can use Lego-brick approach to build the desired NoC topology, though as system-level block such a NoC has low flexibility. Thus it may be disadvantageous, when trying to find the optimum SoC communication architecture in a recursive design space exploration process. The ÆTHEREAL [Goossens et al. 2002], SoCBUS [Sathe et al. 2003], and aSoC [Liang et al. 2000] networks describe the NoC as a relatively coarse grain system-level module but with widely different characteristics. The ÆTHEREAL is highly flexible in terms of adjusting the available slots, number of ports, etc which is useful for NoC exploration; where as aSoC and SoCBUS do not expose many parameters for change (though aSoC supports flexible programming of connections after instantiation). The SPIN NoC [Guerrier and Greiner 2000], designed as a single IP core, is least parameterizable with its fixed topology and protocol. Interestingly, the ×pipes [Osso et al. 2003] provides not merely a set of relatively fine-grain soft-macros of switches and pipelined links, which the ×pipesCompiler [Jalabert et al. 2004] uses to automatically instantiate an application specific network, but also enables optimization of system-level parameters such as removing redundant buffers from output ports of switches, sharing signals common to objects, etc. This lends itself to both flexibility for co-exploration and easy architectural changes when needed. Similarly, conclusions can be drawn of Proteo [Siguenza-Tortosa et al. 2004], HERMES [Moraes et al. 2004] and MANGO [Bjerregaard and Sparsø 2005a] NoCs. A detailed comparison of different features of most of the above listed NoCs is tabulated in [Moraes et al. 2004].

The impact on SoC design time and co-exploration, of different NoC design styles listed above is considerable. For example in [Jalabert et al. 2004], during design space exploration, to find an optimum NoC for three video applications: video object plane decoder, MPEG4 decoder and multi-window displayer; the ×pipesCompiler found that irregular networks with large switches may be more advantageous than regular networks. This is easier to realize in macro-block NoC such as CHAIN or ×pipes, than it is in NoC designed as a single (system-level) IP core such as SPIN. The basis for the compiler's decision is the pattern of traffic generated by the application. This is the focus of the next section. For further understanding of trade-offs in using a flexible instantiation-specific NoC can be found in [Pestana et al. 2004], where different NoC topologies and each topology with different router and NA configuration is explored.

### 4.3 Traffic Characterization

The communication types expected in a NoC range across virtual wires, memory access, audio/video stream, interrupts, and others. Many combinations of topology, protocol, packet sizes and flow control mechanisms exist for the efficient communication of one or more predominant traffic patterns. For example, in [Kumar et al. 2002] packet-switched NoC concepts have been applied to a 2D mesh network topology, whereas in [Guerrier and Greiner 2000] such concepts have been applied to a butterfly fat-tree topology. The design decisions were based on the traffic expected in the respective systems. Characterizing the expected traffic is an important first step towards making sound design decisions.

A NoC must accommodate different types of communication. We have realized that regardless of the system composition, clustering, topology and protocol, the traffic within a system will fall in one of three categories:

(1) **Latency-critical:** Latency-critical traffic is traffic with stringent latency demands such as for critical interrupts, memory access, etc. These often have low payload,

(2) **Data-streams:** Data streaming traffic have high payload and demand QoS in terms of bandwidth. Most often it is large, mostly fixed bandwidth which may be jitter critical. Examples are MPEG data, DMA access, etc.

(3) **Miscellaneous:** This is traffic with no specific requirements of commitment from the network.

The categorization above is a guideline, rather than a hard specification and is presented as a superset of possible traffic types. Bolotin et al. [Bolotin et al. 2004] provide a more refined traffic categorization, combining the transactions at the network boundary with service requirements, namely: signaling, real-time, read/write (RD/WR) and block-transfer. In relation to the above categorization; signaling is latency-critical, real-time is data-streaming, and RD/WR and block-transfer are both miscellaneous with distinguishing factor being the message size. Though one or more of the traffic patterns may be predominant in the SoC, it is important to understand that a realistic NoC design should be optimized for a mix of above traffic patterns. The conclusions of a case-study of NoC routing mechanism for three traffic conditions with fixed number of flits per packet as presented in [Ost et al. 2005], can thus be enriched by using non-uniform packet size and relating them to the above traffic categories.

It is important to understand the bandwidth requirements of the listed traffic types for a given application, and accordingly map the IP cores on the choosen NoC topology. Such

a study is done in [Murali and Micheli 2004a]. NMAP (now called SUNMAP [Murali and Micheli 2004b]), a fast mapping algorithm that minimizes the average communication delay with minimal-path and split-traffic routing in 2D mesh, is compared with greedy and partial branch-and-bound algorithms. It is shown to produce results of higher merit (reduced packet latency) for DSP benchmarks. Another dimension in the mapping task is that of allocating guaranteed communication resources. In [Goossens et al. 2005] and [Hansson et al. 2005], approaches to this task are explored for the ÆTHEREAL NoC.

Specific to data-stream type traffic described above, Rixner et al. [Rixner et al. 1998] have identified unique qualities relating to the inter-dependencies between the media streams and frequency of such streams in the system. It is called the streaming programming model. The basic premises of such programming is static analysis of the application to optimize the mapping effort, based on prior knowledge of the traffic pattern, so as to minimize communication. The communication architecture tuner (CAT) proposed by Lahiri et al. [Lahiri et al. 2000] is a hardware-based approach that does runtime analysis of traffic and manipulates the underlying NoC protocol. It does this by monitoring the internal state and communication transactions of each core, and then predicts the relative importance of each communication event in terms of its potential impact on different system-level performance metrics such as number of deadline misses, average processing time, etc.

The various blocks of NoC can be tuned for optimum performance with regard to a specific traffic characteristic, or the aim can be more general, towards a one-fits-all network, for greater flexibility and versatility.

## 5. NETWORK ANALYSIS

The most interesting and universally applicable parameters of NoC are *latency*, *bandwidth*, *jitter*, *power consumption* and *area usage*. Latency, bandwidth and jitter can be classified as performance parameters, while power consumption and area usage are the cost factors. In this section we will discuss the analysis and presentation of results in relation to these parameters.

### 5.1 Performance Parameters and Benchmarks

Specifying a single of the performance parameters introduced above is not sufficient to confer a properly constrained NoC behavior. The following example illustrates this:

Given a network during normal operation, it is assumed that the network is not overloaded. For such a network, all data is guaranteed to reach its destination, when employing a routing scheme in which no data is dropped (see Section 3.2.2, delay routing model). This means that as long as the capacity of the network is not exceeded, any transmission is guaranteed to succeed (any required bandwidth is guaranteed). However, nothing is stated concerning the transmission latency, which may well be very high in a network operated near full capacity. As seen in Figure 18, the exact meaning of which will be explained later, the latency of packets rise in an exponential manner, as the *network load* increases. The exact nature of the network load will be detailed later in this section. It is obvious that such guarantees are not practically usable. We observe that the bandwidth specification is worthless with out a bound on the latency as well. This might also be presented in terms of a maximum time window, within which the specified bandwidth would always be reached, i.e. the jitter of the data stream (the *spread* of the latencies). Jitter is often a more interesting parameter in relation to bandwidth, than latency, as it describes the temporal evenness of the data stream.

Likewise, a guaranteed bound on latency might be irrelevant, if the bandwidth supported at this latency is insufficient. Thus latency, bandwidth and jitter are closely related. Strictly speaking, one should not be specified without at least one of the others.

At a higher abstraction level, performance parameters used in evaluating multicomputer networks in general have been adopted by NoC researchers. These include *aggregated bandwidth*, *bisection bandwidth*, *link utilization*, *network load*, etc. The aggregate bandwidth is the accumulated bandwidth of all links, and the bisection bandwidth is the minimum collective bandwidth across links that when cut, separate the network into two equal set of nodes. Link utilization is the load on the link, compared with the total bandwidth available. The network load can be measured as a fraction of the *network capacity*, as *normalized bandwidth*. The network capacity is the maximum capacity of the network for a uniform traffic distribution, assuming that the most heavily loaded links are located in the network bisection. These and other aspects of network performance metrics are discussed in detail in Chapter 9 of [Duato et al. 2003].

For highly complex systems, such as full-fledged computer systems including processor(s), memory and peripherals, the individual parameters may say little about the overall functionality and performance of the system. In such cases, it is customary to make use of benchmarks. NoC-based systems represents such complexity, and benchmarks would be natural to use in its evaluating. Presenting performance in the form of benchmark results would help clarify the effect of implemented features, in terms of both performance benefits (latency, jitter and bandwidth) and implementation and operation costs (area usage and power consumption). Benchmarks would thus provide a uniform plane of reference from which to evaluate different NoC architectures. At present, no benchmark system exists explicitly for NoC, but its development is an exciting prospect. In [Vaidya et al. 2001] examples from the NAS benchmarks [Bailey et al. 1994] were used, in particular Class-A NAS-2. This is a set of benchmarks that has been developed for the performance evaluation of highly parallel supercomputers, which mimic the computation and data movement characteristics of large scale computational fluid dynamics applications. It is questionable however, how such parallel computer benchmarks can find use in NoC, as the applications in SoCs are very different. In particular, SoC applications are generally highly heterogeneous, and the traffic patterns therein likewise. Another set of benchmarks, used as basis of NoC evaluation in [Hu and Marculescu 2004a], are the embedded system synthesis suite (E3S) [Dick 2002].

## 5.2 Presenting Results

Generally it is necessary to simplify the multidimensional performance space. One common approach is to adjust a single aspect of the design, while tracking the effect on the performance parameters. An example is tracking the latency of packets, while adjusting the bandwidth capabilities of a certain link within the network, or the amount of background traffic generated by the test environment. In Section 5.2.1 we will give specific examples of simple yet informative ways of communicating results of NoC performance measurements.

Since the NoC is a shared, segmented communication structure, wherein many individual data transfer sessions can take place in parallel, the performance measurements depend not only on the traffic being measured upon, but also on the other traffic in the network, the *background traffic*. The degree of background traffic is often being indicated by the network load, as described above. Though very simple, this definition makes valuable sense

in considering a homogeneous, uniformly loaded network. One generally applicable practical method for performance evaluation is thus generating a uniform randomly distributed background traffic so that the network load reaches a specified point. Test packets can then be sent from one node to another, according to the situation that one desires to investigate, and the latencies of these packets can be recorded (see example (i) in Section 5.2.1 below).

Evenly distributed traffic however, may cloud important issues of the network performance. In [Dally and Aoki 1993] the degree of symmetry of the traffic distribution in the network was used to illustrate aspects of different types of routing protocols; adaptive and deterministic. The adaptive protocol resulted in a significant improvement of throughput over the deterministic one, for non-uniform traffic, but had little effect on performance with uniformly distributed traffic. The reason for this is that the effect of adaptive protocols is to even out the load, to avoid *hotspots*, thus making better use of the available network resources. If the bulk load is already evenly distributed, there is no advantage. Also traffic parameters like number of packets and packet size, can have a great influence on performance, e.g. in relation to queueing strategies in nodes.

There are many ways to approach the task of presenting test results. The performance space is a complex, multidimensional one, and there are many pitfalls to be avoided, in order to display intelligible and valuable information about the performance of a network. Often the presented results fail to show the interesting aspects of the network. It is easy to get lost in the multitude of possible combinations of test parameters. This may lead to clouding, or at worst failure to properly communicate, the relevant aspects of the research. Though the basis for performance evaluation may vary greatly, it is important for researchers to be clear about the evaluation conditions, allowing others to readily and intuitively grasp the potential of a newly developed idea, and the value of its usage in NoC.

5.2.1 *Examples.* Below we will give some specific examples that we find clearly communicate the performance of the networks being analyzed. What makes these examples good are their simplicity in providing a clear picture of some very fundamental properties of the involved designs.

**(i) Average latency vs. network load.** In [Dally and Aoki 1993] this is used to illustrate the effect of different routing schemes. Figure 18 is a figure from the article, showing how the average latency of the test data grows exponentially as the background traffic load of the network is increased. In the presented case, the *throughput saturation point*, the point at which the latency curve bends sharply upwards, is shifted right as more complex routing schemes are applied. This corresponds to a better utilization of available routing resources. The article does not address the question of cost factors of the implementation.

**(ii) Frequency of occurrence vs. latency of packet.** Displaying the average latency of packets in the network may work well for establishing a qualitative notion of network performance. Where more detail is needed, a histogram, or similar graph, showing the distribution of latencies, across the delay spectrum is often used with great effect. This form of presentation is used in [Dally 1992] to illustrate the effect of routing prioritization schemes on the latency distribution. Figure 19, taken from the article, shows the effect of *random scheduling* and *deadline scheduling*. Random scheduling schedules the packets for transmission in a random fashion, while deadline scheduling prioritize packets according to how long they have been waiting (oldest-packet-first). It is seen how the choice of scheduling affect the distribution of latencies of messages. In [Bjerregaard and Sparsø 2005b] such a latency distribution graph is also used, to display how a scheduling scheme

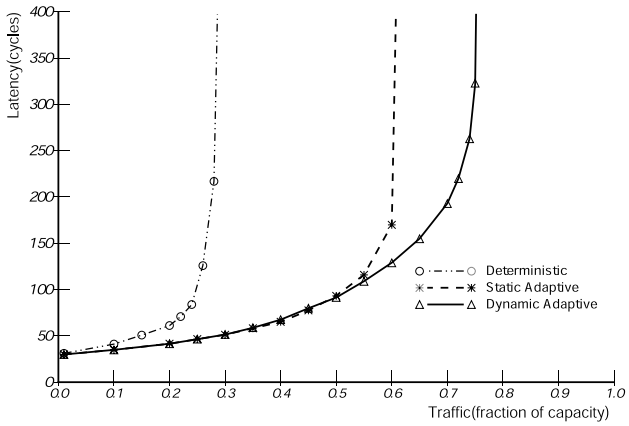



Fig. 18. Latency vs. network load for different routing schemes. The figure shows how the employment of more complex routing schemes move the point at which the network saturates (reprinted from [Dally and Aoki 1993] Fig. 5, ©1993, with permission from IEEE)).

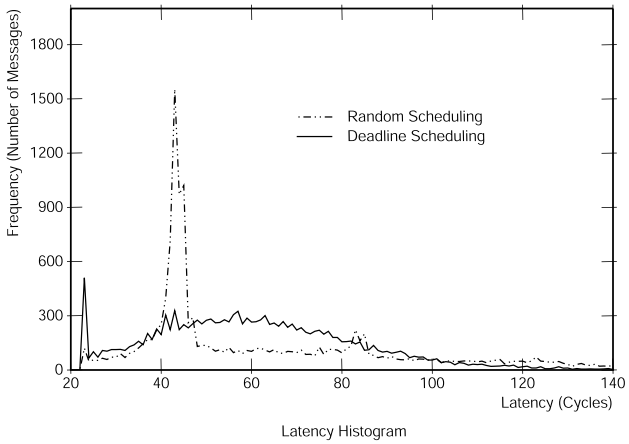



Fig. 19. Number of messages as a function of latency of message (latency distribution), for two scheduling schemes (reprinted from [Dally 1992] Fig. 17, ©1992, with permission from IEEE)).

provides hard latency bounds, in that the graph is completely empty beyond a certain latency.

**(iii) Jitter vs. network load.** The jitter of a sequence of packets is important when dimensioning buffers in the network nodes. High jitter (*bursty* traffic) needs large buffers to compensate, in order to avoid congestion resulting in suboptimal utilization of routing resources. This issue is especially relevant in multimedia application systems with large continuous streams of data, such as that presented in [Varatkar and Marculescu 2002]. In this work statistical mathematical methods are used to analyze the traffic distribution. Figure 20, taken from the article, explores the use of two different models based on stochastic

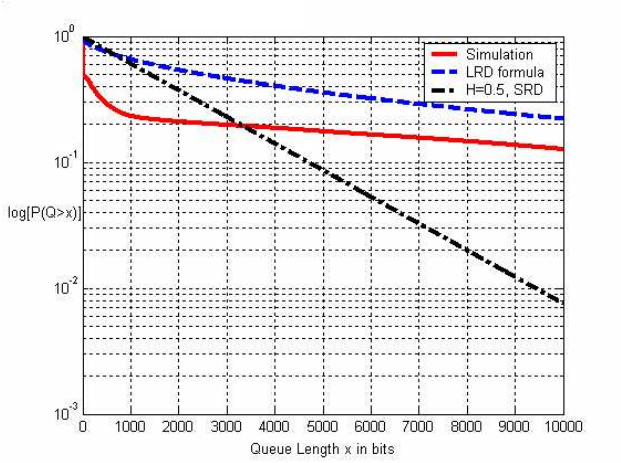



Fig. 20. The probability of queue length exceeding buffer size. The results for two models based on stochastic processes, LRD (Long Range Dependent) and SRD (Short Range Dependent), are plotted along with simulation results for comparison (reprinted from [Varatkar and Marculescu 2002] Fig. 6).

processes, for predicting the probability that the queue length needed to avoid congestion exceeds the actual buffer size, in the given situation. The models displayed in the figure are LRD (Long Range Dependent) or *self-similar*, and SRD (Short Range Dependent) or *Markovian* stochastic processes. In the figure, these models are compared with simulation results. The contributions of the paper include showing that LRD processes can be used effectively to model the bursty traffic behavior at chip-level, and the figure shows how indeed the predictions of the LRD model comes closer to the simulation results than those of the SRD model.

### 5.3 Cost Factors

The cost factors are basically power consumption and area usage. A comparative analysis of cost of NoC is difficult to make. As is the case for performance evaluation, no common ground for comparison exists. This would require different NoC being demonstrated for the same application, which is most often not the case. Hence a somewhat broad discussion of cost in terms of area and power cost is presented in this section.

The power consumption of the communication structure in large single-chip systems is a major concern, especially for mobile applications. As discussed earlier, the power used for global communication does not scale with technology scaling, leading to increased power use by communication relative to power use by processing. In calculating the power consumption of a system, there are two main terms: (i) power per communicated bit and (ii) idle power. Depending on the traffic characteristics in the network, different implementation styles will be more beneficial with regards to power usage. In [Nielsen and Sparsø 2001] a power analysis of different low-power implementations of on-chip communication structures was made. The effects on power consumption of scaling a design were seen and a bus design was compared with torus connected grid design (both synchronous and asynchronous implementations). Asynchronous implementation styles (discussed in

Section 3.3.1), are beneficial for low network usage, since they have very limited power consumption when idle, but use more power per communicated bit, due to local control overhead. Technology scaling however leads to increased leakage current, resulting in an increasing static power use in transistors. Thus the benefit of low idle power in asynchronous circuits may dwindle.

From a system-level perspective, knowledge of network traffic can be used to control the power use of the cores. Interest has been in investigating centralized versus distributed power management (DPM) schemes. Centralized power managers (PM) are a legacy in bus-based systems. Since NoC is most often characterized by distributed routing control, naturally distributed PMs such as those proposed in [Benini and Micheli 2001] and [Simunic and Boyd 2002], would be useful. In both of these studies, conceptually there is a node-centric and network-centric PM. The node-centric PM controls the powering up or down of the core. The network-centric PM is used to for overall load-balancing and to provide some estimations to the node-centric PM of incoming requests, thus masking the core's wake-up cost by precognition of traffic. This type of power management is expected to be favored to reduce power consumption in future NoCs. The results, presented in [Simunic and Boyd 2002] show that with only node PM, the power saving range from factor of 1.5 to 3 compare to no power managers. Combining dynamic voltage scaling with DPM gives overall saving of factor of 3.6. The combined implementation of node and network centric management approaches shows energy savings of a factor of 4.1 with performance penalty reduced by minimum 15% compared to node-only PM. Unlike these dynamic runtime energy monitors, in [Hu and Marculescu 2004b] a system-level energy-aware mapping and scheduling (EAS) algorithm is proposed, which statically schedules both communication transactions and computation tasks. For experiments done on 2D mesh with minimal-path routing, energy savings of 44% are reported, when executing complex multimedia benchmarks.

A design constraint of NoC less applicable to traditional multicomputer networks, lies in the area usage. A NoC is generally required to take up less than 5% of the total chip area. For a 0.13 $\mu$m SoC with one network node per core, and an average core size of 2x2 mm (app. 100 cores on a large chip), this corresponds to 0.2 mm$^2$ per node. One must also remember that the NA will use some area, depending of the complexity of the features that it provides. Trade-off decisions which are applicable to chip design in general and not particular to NoC are beyond the scope of this survey. At the network level, many researchers have concluded that buffering accounts for the major portion of the node area, hence wormhole routing has been a very popular choice in NoCs, see Section 3.2.2. As examples of an area issue related to global wires can be mentioned that introducing *fat wires*, i.e. the usage of wide and tall top level metal wires for global routing, the power figures may improve, at the expense of area [Sylvester and Keutzer 2000].

## 6. NOC EXAMPLES

In this section we briefly recapitulate on a handful of specific NoC examples, describing the design choices of actual implementations, and accompanying work by the research groups behind. This is by no means to be seen as a complete compilation of existing NoCs, there are many more, rather the purpose of this section is to address a representative set: ÆTHEREAL, NOSTRUM, SPIN, CHAIN, MANGO, and ×PIPES. In [Moraes et al. 2004] a list in tabular form is provided, which effectively characterizes many of the NoCs

not covered in the following.

i **ÆTHEREAL:** The ÆTHEREAL, developed at Philips, is a NoC that provides guaranteed throughput (GT) along side best-effort (BE) service [Rijpkema et al. 2001][Goossens et al. 2002][Wielage and Goossens 2002][Dielissen et al. 2003][Jantsch and Tenhunen 2003](pgs: 61-82)[Rijpkema et al. 2003][Radulescu et al. 2004][Goossens et al. 2005]. In the ÆTHEREAL the guaranteed services pervade as a requirement for hardware design and also as a foundation for software programming. The router provides both GT and BE services. All routers in the network have a common sense of time, and the routers forward traffic based on slot allocation. Thus a sequence of slots implement a virtual circuit. GT traffic is connection-oriented, and did in early router instantiations not have headers, as the next hop was determined by a local slot table. In recent versions the slot tables have been removed to save area, and the information is provided in a GT packet header. The allocation of slots can be setup statically, during an initialization phase, or dynamically during runtime. BE traffic makes use of non-reserved slots and of any slots reserved but not used. BE packets are used to program the GT slots of the routers. With regards to buffering, input queuing is implemented using custom-made hardware fifos, to keep the area costs down. The ÆTHEREAL connections support a number of different transaction types, such as read, write, acknowledged write, test and set, and flush, and as such it is similar to existing bus protocols. In addition, it offers a number of connection types: narrowcast, multicast, and simple.
In [Dielissen et al. 2003] an ÆTHEREAL router with 6 bidirectional ports of 32 bits was synthesized in 0.13 $\mu$m CMOS technology. The router had custom made BE input queues depth of 24 words per port. The total area was 0.175 $mm^2$, and the bandwidth was 500 MHz x 32 bits = 16 Gbit/s per port. A network adapter with 4 standard socket interfaces (either master or slave; OCP, DTL or AXI based) was also reported with an area of 0.172 $mm^2$ implemented in the same technology.
In [Goossens et al. 2005] and [Pestana et al. 2004] an automated design flow for instantiation of application specific ÆTHEREAL is described. The flow uses XML to input various parameters such as traffic characteristics, GT and BE requirements, and topology. A case study of MPEG codec SoC is used to validate and verify the optimizations undertaken during the automated flow.

ii **NOSTRUM:** The work of researchers at KTH in Stockholm has evolved from a system-level chip design approach [Kumar et al. 2002][Jantsch and Tenhunen 2003][Zimmer and Jantsch 2003][Millberg et al. 2004]. Their emphasis has been on architecture and platform-based design, targeted towards multiple application domains. They have recognized the increasing complexity of working with high density VLSI technologies and hence highlighted advantages of a grid-based, router-driven communication media for on-chip communication.
Also the implementation of guaranteed services has also been a focus point of this group. In the NOSTRUM NoC guaranteed services are provided by so called *looped containers*. These are implemented by virtual circuits, using an explicit time division multiplexing mechanism which they call *Temporally Disjoint Networks* (TDN) (refer to Sections 3.2.2 and 3.2.3 for more details).
In [Jantsch and Vitkowski 2005], the group addressed encoding issues and showed that lowering the voltage swing, then re-establishing reliability using error correction,

actually resulted in better power saving than a number of dedicated power saving algorithms used for comparison.

iii **SPIN:** The SPIN network (*Scalable Programmable Integrated Network*) [Guerrier and Greiner 2000][Andriahantenaina and Greiner 2003] implements a fat-tree topology with two one-way 32-bit datapaths at the link layer. The fat-tree is an interesting choice of irregular network, claimed in [Leiserson 1985] to be *nearly the best routing network* for a given amount of hardware. It is proven that for any given amount of hardware, a fat-tree can simulate any other network built from the same amount of hardware, with only a polylogarithmic slowdown in latency. This is in contrast to e.g. two-dimensional arrays or simple trees which exhibit polynomial slowdown when simulating other networks, and as such do not have any advantage over a sequential computer.

In SPIN, packets are sent via the network as a sequence of flits each of size 4 bytes. Wormhole routing is used, with no limit on packet size. The first flit contains the header, with one byte reserved for addressing, and the last byte of the packet contains the payload checksum. There are three types of flits; *first*, *data* and *last*. Link-level flow control is used to identify the flit type and act accordingly upon its content. The additional bytes in the header can be used for packet tagging for special services, and for special routing options. The performance of the network was evaluated primarily based on uniform randomly distributed load (see Section 5). It was noted that random hick-ups can be expected under high load. It was found that the protocol accounts for about 31% of the total throughput, a relatively large overhead. In 2003, a 32-port SPIN network was implemented in a 0.13 $\mu$m CMOS process, the total area was 4.6 mm$^2$ (0.144 mm$^2$ per port), for an accumulated bandwidth of about 100Gbits/s.

iv **CHAIN:** The CHAIN network (*CHip Area INterconnect*) [Bainbridge and Furber 2002], developed at the University of Manchester, is interesting in that it is implemented entirely using asynchronous, or *clockless*, circuit techniques. It makes use of delay insensitive 1-of-4 encoding, and source routes BE packets. An easy adaption along a path consisting of links of different bit widths is supported. CHAIN is targeted for heterogeneous low power systems, in which the network is system specific. It has been implemented in a smart card, which benefits from the low idle power capabilities of asynchronous circuits. Work from the group involved with CHAIN concerns prioritization in asynchronous networks. In [Felicijan et al. 2003] an asynchronous low latency arbiter was presented, and its use in providing differentiated communication services in SoC was discussed, and in [Felicijan and Furber 2004] a router implementing the scheme was described.

v **MANGO:** The MANGO network (*Message-passing Asynchronous Network-on-chip providing Guaranteed services over OCP interfaces*), developed at the Technical University of Denmark, is another clockless NoC, targeted for coarse-grained GALS-type SoC [Bjerregaard 2005]. MANGO provides connection-less BE routing as well as connection-oriented guaranteed services (GS) [Bjerregaard and Sparsø 2005a]. In order to make for a simple design, the routers implement virtual channels (VCs) as separate physical buffers. GS connections are established by allocating a sequence of VCs through the network. While the routers themselves are implemented using area efficient bundled-data circuits, the links implement delay insensitive signal encoding. This makes global timing robust, because no timing assumptions are necessary

between routers. A scheduling scheme called ALG (*Asynchronous Latency Guarantees*) [Bjerregaard and Sparsø 2005b], schedules access to the links, allowing latency guarantees to be made, which are not inversely dependent on the bandwidth guarantees, as is the case in TDM-based scheduling schemes. Network adapters provide OCP-based standard socket interfaces, based on the primitive routing services of the network [Bjerregaard et al. 2005]. This includes support for interrupts, based on virtual wires. The adapters also synchronize the clocked OCP interfaces to the clockless network.

vi **×PIPES:** ×pipes [Osso et al. 2003] and the accompanying NetChip compiler (a combination of ×pipesCompiler [Jalabert et al. 2004] and SUNMAP [Murali and Micheli 2004b]) are developed by University of Bologna and Stanford University. ×pipes consists of soft macros of switches and links that can be turned into instance-specific network components at instantiation time. It promotes the idea of pipelined links with a flexible number of stages to increase throughput. A go-back-N retransmission strategy is implemented as part of link-level error control, which reduces switch complexity, though at considerable delay since each flit is not acknowledged until it has been transmitted across the destination switch. The error is indicated by a CRC block running concurrently with switch operation. Thus the ×pipes architecture lends itself to be robust to interconnect errors. Overall, delay for a flit to traverse from across one link and node is 2N+M cycles where N is number of pipeline stages and M is switch stages. The ×pipesCompiler is a tool to automatically instantiate an application specific custom communication infrastructure using ×pipes components. It can tune flit size, degree of redundancy of the CRC error-detection, address space of cores, number of bits used for packet sequence count, maximum number of hops between any two network nodes, number of flit size, etc.

In a top-down design methodology, once the SoC floorplan is decided, the required network architecture is fed into the ×pipesCompiler. Examples of compiler optimization include removing redundant buffers from missing output ports of switches, sharing signals common to objects, etc. Via case studies presented in [Bertozzi et al. 2005], the NetChip compiler has been validated for mesh, torus, hypercube, Clos and butterfly NoC topologies for four video processing applications. Four routing algorithms: dimension-ordered, minimum-path, traffic splitting across minimum-path, and traffic splitting across all paths, is also part of the case study experiments. The floorplan of switches and links of NoC takes the IP block size into consideration. Results are available for average hop delay, area and power for mapping of each of the video application on the topologies. A light-weight implementation, named ×pipes-lite, presented in [Stergiou et al. 2005], is similar in to ×pipes in concept, but is however optimized for link latency, area and power, and provides direct synthesis path from SystemC description.

## 7. SUMMARY

NoC encompasses a wide spectrum of research, ranging from highly abstract software related issues, across system topology to physical level implementation. In this survey we have given an overview of activities in the field. We have first stated the motivation for NoC and given an introduction of the basic concepts. In order to avoid the wide range of topics relevant to large scale IC design in general, we have assumed a view of NoC as a

subset of SoC.

From a system level perspective, NoC is motivated by the demand for a well structured design approach in large scale SoCs. A modularized design methodology is needed, in order to make efficient use of the ever increasing availability of on-chip resources in terms of number of transistors and wiring layers. Likewise, programming these systems necessitates clear programming models and predictable behavior. NoC has the potential to provide modularity through the use of standard sockets such as OCP, and predictability through the implementation of guaranteed communication services. From a physical level perspective, with scaling of technologies into the DSM region, the increasing impact of wires on performance forces a differentiation between local and global communication. In order for global communication structures to exhibit scalability and high performance, segmentation, wire sharing and distributed control is required.

In structuring our work, we have adopted a layered approach similar to OSI, and divided NoC research into four areas: *System*, *Network Adapter*, *Network* and *Link* research. In accordance with the view of NoC as a subset of SoC, we have dealt first with the latter three areas of research, which relate directly to the NoC implementation. Thereafter, we have focussed on system level aspects.

The network adapter orthogonalizes communication and computation, enabling *communication-centric* design. It is thus the entity which enables a modularized design approach. Its main task is to decouple the core from the network, the purpose being to provide high-level network-agnostic communication services based on the low-level routing primitives provided by the network hardware. In implementing standard sockets, IP reuse becomes feasible, and the network adapter may thus hold the key to commercial success of NoC.

At the network level, issues such as network topology, routing protocols, flow control, and quality-of-service are dominant. With regards to topology, NoC is restricted by a 2D layout. This has made grid-type topologies a wide-spread choice. We have reviewed the most common routing schemes, *store-and-forward*, *wormhole* and *virtual cut-through* routing, and concluded that wormhole routing is by far the most common choice for NoC designs. The use of *virtual channels* in avoiding deadlocks and providing guaranteed services was illustrated and the motivation for guaranteed services was discussed. The predictability that such services incur facilitates easy system integration and analytical system verification, particularly relevant for real-time systems.

Unlike in macro networks, in NoC network adapter and network functionality is often implemented in hardware rather than in software. This is so, since NoC-based systems are more tightly bound, and simple, fast, power efficient solutions are required.

Link level research is much more hardware oriented. We have covered topics like synchronization, i.e. between clock domains, segmentation and pipelining of links in order to increase bandwidth and counteract physical limitations of DSM technologies, on-chip signaling such as low-swing drivers used to decrease the power usage in links, and future technologies such as on-chip wave guides and optical interconnects. Also we have discussed the reliability of long links, which are susceptible to a number of noise sources: *crosstalk*, *ground bounce*, *EMI* and *inter-symbol interference*. Segmentation helps keep the effect of these at bay, since the shorter a wire is the less influence they will have. Error detection and correction in on-chip interconnects was discussed, but this is not a dominating area of research. Different encoding schemes were discussed in relation to increasing

bandwidth as well as reducing power consumption.

NoC facilitates communication-centric design, as opposed to traditional computation-centric design. From a system level perspective, we have addressed topics relating to the role of NoC in SoC design flows. Key issues are modeling, design methodology and traffic characterization. The purpose of modeling is to evaluate trade-offs with regard to global traffic, in terms of power, area, design time, etc., while adhering to application requirements. With regard to design methodology, we identify two important characteristics of NoC, by which we classify a number of existing NoC solutions: (i) *parametrizability* of the NoC as a system level block and (ii) *granularity* of the NoC components by which the NoC is assembled. These characteristics greatly influence the nature of the design flow enabled by the particular NoC. As a tool towards identifying general requirements of a NoC, we have identified a set of traffic types, *latency-critical*, *data-streams* and *miscellaneous* traffic, which span the spectrum of possible traffic in a NoC-based system.

The basic performance parameters of NoC are *latency*, *bandwidth* and *jitter*. The basic cost factors are *power consumption* and *area usage*. At a higher level of abstraction, terms like *aggregate bandwidth*, *bisection bandwidth*, *link utilization* and *network load* can be used. These originate in multicomputer network theory and relate to data movement in general. Stepping up yet another abstraction level, benchmarks can be used for performance analysis. Currently no benchmarks exist specifically for NoC, but the use of benchmarks for parallel computers, as well as embedded systems benchmarks, has been reported.

Six case studies are conducted, explaining the design choices of the *ÆTHEREAL*, *NOSTRUM*, *SPIN*, *CHAIN*, *MANGO* and *×PIPES* NoC implementations. CHAIN and ×PIPES target a platform-based design methodology, in which a heterogeneous network can be instantiated for a particular application. ÆTHEREAL, NOSTRUM and MANGO implement more complex features such as guaranteed services, and target a methodology which draws closer to backbone-based design. SPIN differs from the others in that it implements a fat-tree, rather than a grid-type topology. CHAIN and MANGO also differ in that they are implemented entirely using clockless circuit techniques, and as such inherently support globally asynchronous and locally synchronous (GALS) systems.

Continued technology scaling enables large scale SoC. NoCs facilitate a modular, scalable design approach that overcomes both system and physical level issues. The main job of the NoC designer of the future will be to dimension and structure the network, according to the communication needs of the SoC. At present, an interesting challenge lies in specifying ways to define these needs.

## 8. ACKNOWLEDGEMENTS

We would like to thank professors Jens Sparsø and Jan Madsen of the Department for Informatics and Mathematical Modelling (IMM) at the Technical University of Denmark (DTU) for their tireless effort in helping us review, iterate and structure this survey. Also our grateful thanks to professor Axel Jantsch (KTH - Stockholm, Sweden) and Andrei Radulescu (Phillips - Eindhoven, Netherlands) for their valuable review of the survey as it was closing in on its final form, and to Mihai Budiu (Carnegie Mellon University - Pittsburgh, USA) for comments and suggestions. Finally, the extensive comments of the anonymous reviewers have helped in taking the survey to its final form.

REFERENCES

AGARWAL, A. 1999. The Oxygen project - Raw computation. *Scientific American*, 44–47.

AGGARWAL, A. AND FRANKLIN, M. 2002. Hierarchical Interconnects for On-chip Clustering. In *Proceedings of the 16th International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE Computer Society, 602–609.

AHONEN, T., SIGÜENZA-TORTOSA, D. A., BIN, H., AND NURMI, J. 2004. Topology optimization for application-specific networks-on-chip. In *International Workshop on System Level Interconnect Prediction (SLIP)*. ACM, 53–60.

AL-TAWIL, K. M., ABD-EL-BARR, M., AND ASHRAF, F. 1997. A survey and comparison of wormhole routing techniques in a mesh networks. *IEEE Network 11*, 38–45.

AMDE, M., FELICIJAN, T., EDWARDS, A. E. D., AND LAVAGNO, L. 2005. Asynchronous on-chip networks. *IEE Proceedings of Computers and Digital Techniques 152*, 273–283.

ANDREASSON, D. AND KUMAR, S. 2004. On improving best-effort throughput by better utilization of guaranteed-throughput channels in an on-chip communication system. In *Proceeding of 22th IEEE Norchip Conference*. IEEE.

ANDREASSON, D. AND KUMAR, S. 2005. Slack-time aware routing in NoC systems. In *International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2353–2356.

ANDRIAHANTENAINA, A. AND GREINER, A. 2003. Micro-network for SoC : Implementation of a 32-port spin network. In *Proceedings of Design, Automation and Test in Europe Conference and Exhibition*. IEEE, 1128–1129.

ARM. 2004. AMBA Advanced eXtensible Interface (AXI) Protocol Specification, Version 1.0. http://www.arm.com.

ARTERIS. 2005. A comparison of network-on-chip and busses. White paper downloadable from http://www.arteris.com/noc_whitepaper.pdf.

BAILEY, D., BARSZCZ, E., BARTON, J., BROWNING, D., CARTER, R., DAGUM, L., FATOOHI, R., FINEBERG, S., FREDERICKSON, P., LASINSKI, T., SCHREIBER, R., SIMON, H., VENKATAKRISHNAN, V., AND WEERATUNGA, S. 1994. RNR technical report RNR-94-007. Tech. rep., NASA Ames Research Center.

BAINBRIDGE, J. AND FURBER, S. 2002. CHAIN: A delay-insensitive chip area interconnect. *IEEE Micro 22*, 5 (October), 16–23.

BAINBRIDGE, W. AND FURBER, S. 2001. Delay insensitive system-on-chip interconnect using 1-of-4 data encoding. In *Proceedings of the 7th International Symposium on Asynchronous Circuits and Systems (ASYNC)*. 118 – 126.

BANERJEE, N., VELLANKI, P., AND CHATHA, K. S. 2004. A power and performance model for network-on-chip architectures. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE, 1250–1255.

BEIGNE, E., CLERMIDY, F., VIVET, P., CLOUARD, A., AND RENAUDIN, M. 2005. An asynchronous NOC architecture providing low latency service and its multi-level design framework. In *Proceedings of the 11th International Symposium on Asynchronous Circuits and Systems (ASYNC)*. IEEE, 54–63.

BENINI, L. AND MICHELI, G. D. 2001. Powering network-on-chips. In *The 14th International Symposium on System Synthesis (ISSS)*. IEEE, 33–38.

BENINI, L. AND MICHELI, G. D. 2002. Networks on chips: A new SoC paradigm. *IEEE Computer 35*, 1 (January), 70–78.

BERTOZZI, D., JALABERT, A., MURALI, S., TAMHANKAR, R., STERGIOU, S., BENINI, L., AND DE MICHELI, G. 2005. NoC synthesis flow for customized domain specific multiprocessor Systems-on-Chip. In *Transactions on Parallel and Distributed Systems*. IEEE, 113–129.

BHOJWANI, P. AND MAHAPATRA, R. 2003. Interfacing cores with on-chip packet-switched networks. In *Proceedings of the Sixteenth International Conference on VLSI Design*. 382–387.

BJERREGAARD, T. 2005. The MANGO clockless network-on-chip: Concepts and implementation. Ph.D. thesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby.

BJERREGAARD, T., MAHADEVAN, S., OLSEN, R. G., AND SPARSØ, J. 2005. An OCP compliant network adapter for gals-based soc design using the MANGO network-on-chip. In *Proceedings of International Symposium on System-on-Chip (ISSoC)*. IEEE.

BJERREGAARD, T., MAHADEVAN, S., AND SPARSØ, J. 2004. A channel library for asynchronous circuit design supporting mixed-mode modeling. In *Proceedings of the Fourteenth International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*. Springer, 301–310.

BJERREGAARD, T. AND SPARSØ, J. 2005a. A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE, 1226–1231.

BJERREGAARD, T. AND SPARSØ, J. 2005b. A scheduling discipline for latency and bandwidth guarantees in asynchronous network-on-chip. In *Proceedings of the 11th International Symposium on Advanced Research in Asynchronous Circuits and Systems*. IEEE, 34–43.

BOGLIOLO, A. 2001. Encodings for high-performance energy-efficient signaling. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED)*. 170–175.

BOLOTIN, E., CIDON, I., GINOSAUR, R., AND KOLODNY, A. 2004. QNoC: QoS architecture and design process for network-on-chip. *J. Syst. Archit. 50*, 2-3, 105–128.

CATTHOOR, F., CUOMO, A., MARTIN, G., GROENEVELD, P., RUDY, L., MAEX, K., DE STEEG, P. V., AND WILSON, R. 2004. How can system level design solve the interconnect technology scaling problem. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE, 332–337.

CHAPIRO, D. 1984. Globally-asynchronous locally-synchronous systems. Ph.D. thesis, Stanford University. Report No. STAN-CS-84-1026.

CHELCEA, T. AND NOWICK, S. M. 2001. Robust interfaces for mixed-timing systems with application to latency-insensitive protocols. In *Proceedings of the 38th Design Automation Conference (DAC)*. IEEE, 21–26.

CHIU, G.-M. 2000. The odd-even turn model for adaptive routing. *IEEE Transactions on Parallel and Distributed Systems 11*, 729 –738.

COLE, R. J., MAGGS, B. M., AND SITARAMAN, R. K. 2001. On the benefit of supporting virtual channels in wormhole routers. *Journal of Computer and System Sciences 62*, 152–177.

CULLER, D. E., SINGH, J. P., AND GUPTA, A. 1998. *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufmann. 1st Edition.

DALLY, W. J. 1990. Performance analysis of k-ary n-cube interconnection networks. *IEEE Transactions on Computer 39*, 6 (June), 775 – 785.

DALLY, W. J. 1992. Virtual-channel flow control. *IEEE Transactions on Parallel and Distributed Systems 3*, 2 (March), 194 – 205.

DALLY, W. J. AND AOKI, H. 1993. Deadlock-free adaptive routing in multicomputer networks using virtual channels. *IEEE Transactions on Parallel and Distributed Systems 4*, 4 (April), 466 – 475.

DALLY, W. J. AND SEITZ, C. L. 1987. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers C-36*, 5 (May), 547–553.

DALLY, W. J. AND TOWLES, B. 2001. Route packets, not wires: On-chip interconnection networks. In *Proceedings of the 38th Design Automation Conference (DAC)*. IEEE, 684–689.

DE MELLO, A. V., OST, L. C., MORAES, F. G., AND CALAZANS, N. L. V. 2004. Evaluation of routing algorithms on mesh based nocs. Tech. rep., Faculdade de Informatica PUCRS - Brazil. May.

DICK, R. 2002. Embedded system synthesis benchmarks suite. http://www.ece.northwestern.edu/ dickrp/e3s/.

DIELISSEN, J., RADULESCU, A., GOOSSENS, K., AND RIJPKEMA, E. 2003. Concepts and implementation of the phillips network-on-chip. In *Proceedings of the IP based SOC (IPSOC)*. IFIP.

DOBBELAERE, I., HOROWITZ, M., AND GAMAL, A. E. 1995. Regenerative feedback repeaters for programmable interconnections. *IEEE Journal of Solid-State Circuits 30*, 11 (November), 1246–1253.

DOBKIN, R., GINOSAUR, R., AND SOTIRIOU, C. P. 2004. Data synchronization issues in GALS SoCs. In *Proceedings of the 10th IEEE International Symposium on Asynchronous Circuits and Systems*. IEEE, 170–179.

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

DUATO, J. 1995. A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks. *IEEE Transactions on Parallel and Distributed Systems 6*, 10 (October), 1055–1067.

DUATO, J. 1996. A necessary and sufficient condition for deadlock-free routing in cut-through and store-and-forward networks. *IEEE Transactions on Parallel and Distributed Systems 7*, 8 (August), 841–854.

DUATO, J. AND PINKSTON, T. M. 2001. A general theory for deadlock-free adaptive routing using a mixed set of resources. *IEEE Transactions on Parallel and Distributed Systems 12,* 12 (December), 1219–1235.

DUATO, J., YALAMANCHILI, S., AND NI, L. 2003. *Interconnection Networks: An Engineering Approach*. Morgan Kaufmann.

FELICIJAN, T., BAINBRIDGE, J., AND FURBER, S. 2003. An asynchronous low latency arbiter for quality of service (QoS) applications. In *Proceedings of the 15th International Conference on Microelectronics (ICM)*. IEEE, 123–126.

FELICIJAN, T. AND FURBER, S. B. 2004. An asynchronous on-chip network router with quality-of-service (QoS) support. In *Proceedings IEEE International SOC Conference*. IEEE, 274–277.

FITZPATRICK, T. 2004. System verilog for VHDL users. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE Computer Society, 21334.

FORSELL, M. 2002. A scalable high-performance computing solution for networks on chips. *IEEE Micro 22,* 5, 46–55.

GAUGHAN, P. T., DAO, B. V., YALAMANCHILI, S., AND SCHIMMEL, D. E. 1996. Distributed, deadlock-free routing in faulty, pipelined, direct interconnection networks. *IEEE Transactions on Computers 45,* 6 (June), 651–665.

GENKO, N., ATIENZA, D., DE MICHELI, G., BENINI, L., MENDIAS, J., HERMIDA, R., AND CATTHOOR, F. 2005. A novel approach for network on chip emulation. In *International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2365–2368.

GERSTLAUER, A. 2003. Communication abstractions for system-level design and synthesis. Tech. Rep. TR-03-30, Center for Embedded Computer Systems, University of California, Irvine, CA 92697-3425, USA. October.

GINOSAUR, R. 2003. Fourteen ways to fool your synchrononizer. In *Proceedings of the 9th International Symposium on Asynchronous Circuits and Systems*. IEEE, 89–96.

GLASS, C. J. AND NI, L. M. 1994. The turn model for adaptive routing. *Journal of the Association for Computing Machinery 41*, 874–902.

GOOSSENS, K., DIELISSEN, J., GANGWAL, O. P., PESTANA, S. G., RADULESCU, A., AND RIJPKEMA, E. 2005. A design flow for application-specific networks on chip with guaranteed performance to accelerate SOC design and verification. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE, 1182–1187.

GOOSSENS, K., DIELISSEN, J., AND RADULESCU, A. 2005. Aethereal network on chip: Concepts, architectures and implementations. *IEEE Design & Test of Computers 22,* 5, 414–421.

GOOSSENS, K., MEERBERGEN, J. V., PEETERS, A., AND WIELAGE, P. 2002. Networks on silicon: Combining best-effort and guaranteed services. In *Proceedings of the Design, Automation and Test in Europe Conference (DATE)*. IEEE, 196–200.

GUERRIER, P. AND GREINER, A. 2000. A generic architecture for on-chip packet-switched interconnections. In *Proceedings of the Design Automation and Test in Europe (DATE)*. IEEE, 250–256.

GUO, M., NAKATA, I., AND YAMASHITA, Y. 2000. Contention-free communication scheduling for array redistribution. *Parallel Computing 26*, 1325–1343.

HANSSON, A., GOOSSENS, K., AND RADULESCU, A. 2005. A unified approach to constrained mapping and routing on networks-on-chip architectures. In *CODES/ISSS*. ACM/IEEE, 75–80.

HARMANCI, M., ESCUDERO, N., LEBLEBICI, Y., AND IENNE, P. 2005. Quantitative modelling and comparison of communication schemes to guarantee quality-of-service in networks-on-chip. In *International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1782–1785.

HAUCK, S. 1995. Asynchronous design methodologies: an overview. *Proceedings of the IEEE 83,* 1 (January), 69–93.

HAVEMANN, R. H. AND HUTCHBY, J. A. 2001. High-performance interconnects: An integration overview. *Proceedings of the IEEE 89,* 5 (May), 586 – 601.

HAVERINEN, A., LECLERCQ, M., WEYRICH, N., AND WINGARD, D. 2002. SystemC based SoC communication modeling for the OCP protocol. White paper downloadable from http://www.ocpip.org.

HEILIGER, H.-M., NAGEL, M., ROSKOS, H. G., AND KURZ, H. 1997. Thin-film microstrip lines for mm and sub-mm-wave on-chip interconnects. In *IEEE MTT-S International Microwave Symposium Digest*. Vol. 2. 421–424.

HO, R., MAI, K., AND HOROWITZ, M. 2003. Efficient on-chip global interconnects. In *Symposium on VLSI Circuits. Digest of Technical Papers*. IEEE, 271–274.

HO, R., MAI, K. W., AND HOROWITZ, M. A. 2001. The future of wires. *Proceedings of the IEEE 89*, 4 (April), 490 – 504.

HU, J. AND MARCULESCU, R. 2004a. Application-specific buffer space allocation for networks-on-chip router design. In *ICCAD*. IEEE Computer Society / ACM, 354–361.

HU, J. AND MARCULESCU, R. 2004b. Energy-aware communication and task scheduling for network-on-chip architectures under real-time constraints. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE, 10234–10240.

ITRS. 2001. International technology roadmap for semiconductors (ITRS) 2001. Tech. rep., International Technology Roadmap for Semiconductors.

ITRS. 2003. International technology roadmap for semiconductors (ITRS) 2003. Tech. rep., International Technology Roadmap for Semiconductors.

JALABERT, A., MURALI, S., BENINI, L., AND MICHELI, G. D. 2004. ×pipesCompiler: A tool for instantiating application specific networks-on-chip. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE, 884–889.

JANTSCH, A. 2003. Communication performance in networks-on-chip. Slides downloadable from http://www.ele.kth.se/ axel/presentations/2003/Stringent.pdf.

JANTSCH, A. AND TENHUNEN, H. 2003. *Networks on Chip*. Kluwer Academic Publishers.

JANTSCH, A. AND VITKOWSKI, R. L. A. 2005. Power analysis of link level and end-to-end data protection in networks-on-chip. In *International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1770–1773.

JUURLINK, B. H. H. AND WIJSHOFF, H. A. G. 1998. A quantitative comparison of parrallel computation models. *ACM Transactions on Computer Systems 16*, 3 (August), 271–318.

KAPUR, P. AND SARASWAT, K. C. 2003. Optical interconnects for future high performance intergrated circuits. *Physica E 16, Issue 3-4*, 620–627.

KARIM, F., NGUYEN, A., AND DEY, S. 2002. An interconnect architecture for networking systems on chips. *IEEE Micro 22*, 36–45.

KARIM, F., NGUYEN, A., DEY, S., AND RAO, R. 2001. On-chip communication architecture for OC-768 network processors. In *Proceedings of the 38th Design Automation Conference (DAC)*. ACM, 678–683.

KIM, D., LEE, K., JOONG LEE, S., AND YOO, H.-J. 2005. A reconfigurable crossbar switch with adaptive bandwidth control for networks-on-chip. In *International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2369–2372.

KIM, K., LEE, S.-J., LEE, K., AND YOO, H.-J. 2005. An arbitration look-ahead scheme for reducing end-to-end latency in networks-on-chip. In *International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2357–2360.

KUMAR, S., JANTSCH, A., SOININEN, J.-P., FORSELL, M., MILLBERG, M., OBERG, J., TIENSYRJÄ, K., AND HEMANI, A. 2002. A network-on-chip architecture and design methodology. In *Proceedings of the Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE Computer Society, 117–124.

KURD, N., BARKATULLAH, J., DIZON, R., FLETCHER, T., AND MADLAND, P. 2001. Multi-GHz clocking scheme for Intel pentium 4 microprocessor. In *Digest of Technical Papers. International Solid-State Circuits Conference (ISSCC)*. IEEE, 404–405.

LAHIRI, K., RAGHUNATHAN, A., AND DEY, S. 2001. Evaluation of the traffic-performance characteristics of system-on-chip communication architectures. In *Proceedings of the 14th International Conference on VLSI Design*. IEEE, 29–35.

LAHIRI, K., RAGHUNATHAN, A., LAKSHMINARAYANA, G., AND DEY, S. 2000. Communication architecture tuners: A methodology for the design of high-performance communication architectures for system-on-chips. In *Proceedings of the Design Automation Conference, DAC*. IEEE, 513–518.

LEE, K. 1998. On-chip interconnects - gigahertz and beyond. *Solid State Technology 41*, 9 (September), 85–89.

LEISERSON, C. E. 1985. Fat-trees: Universal networks for hardware-efficient supercomputing. *IEEE transactions on Computers c-34*, 10, 892–901.

LEROY, A., MARCHAL, P., SHICKOVA, A., CATTHOOR, F., ROBERT, F., AND VERKEST, D. 2005. Spatial division multiplexing: a novel approach for guaranteed throughput on nocs. In *CODES/ISSS*. ACM/IEEE, 81–86.

LIANG, J., LAFFELY, A., SRINIVASAN, S., AND TESSIER, R. 2004. An architecture and compiler for scalable on-chip communication. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems 12*, 7, 711–726.

LIANG, J., SWAMINATHAN, S., AND TESSIER, R. 2000. ASOC: A scalable, single-chip communications architecture. In *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques 2000*. 37–46.

LIU, J., ZHENG, L.-R., AND TENHUNEN, H. 2004. Interconnect intellectual property for network-on-chip (NoC). *Journal of Systems Architecture 50*, 65–79.

LOGHI, M., ANGIOLINI, F., BERTOZZI, D., BENINI, L., AND ZAFALON, R. 2004. Analyzing on-chip communication in a MPSoC environment. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE, 752–757.

MADSEN, J., MAHADEVAN, S., VIRK, K., AND GONZALEZ, M. 2003. Network-on-chip modeling for system-level multiprocessor simulation. In *Proceedings of the 24th IEEE International Real-Time Systems Symposium (RTSS)*. IEEE, 82–92.

MAHADEVAN, S., STORGAARD, M., MADSEN, J., AND VIRK, K. 2005. ARTS: A system-level framework for modeling MPSoC components and analysis of their causality. In *The 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE Computer Society.

MAI, K., PAASKE, T., JAYASENA, N., HO, R., DALLY, W. J., AND HOROWITZ, M. 2000. Smart memories: A modular reconfi gurable architecture. In *Proceedings of 27th International Symposium on Computer Architecture*. 161–171.

MEINCKE, T., HEMANI, A., KUMAR, S., ELLERVEE, P., OBERG, J., OLSSON, T., NILSSON, P., LINDQVIST, D., AND TENHUNEN, H. 1999. Globally asynchronous locally synchronous architecture for large high-performance ASICs. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*. Vol. 2. 512 –515.

MILLBERG, M., NILSSON, E., THID, R., AND JANTSCH, A. 2004. Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network-on-chip. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE, 890–895.

MIZUNO, M., DALLY, W. J., AND ONISHI, H. 2001. Elastic interconnects: Repeater-inserted long wiring capable of compressing and decompressign data. In *Proceedings of the International Solid-State Circuits Conference*. IEEE, 346–347, 464.

MORAES, F., CALAZANS, N., MELLO, A., MÖLLER, L., AND OST, L. 2004. HERMES: An infrastructure for low area overhead packet-switching networks on chip. *The VLSI Integration 38*, 69–93.

MULLINS, R. AND MOORE, A. W. S. 2004. Low-latency virtual-channel routers for on-chip networks. In *Proceedings of the 31st Annual International Symposium on Computer Architecture*. IEEE, 188–197.

MURALI, S. AND MICHELI, G. D. 2004a. Bandwidth-constrained mapping of cores onto noc architectures. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE, 20896–20902.

MURALI, S. AND MICHELI, G. D. 2004b. SUNMAP: A tool for automatic topology selection and generation for NoCs. In *In Proceedings of the 41st Design Automation Conference (DAC)*. IEEE, 914–919.

MUTTERSBACH, J., VILLIGER, T., AND FICHTNER, W. 2000. Practical design of globally-asynchronous locally-synchronous systems. In *Proceedings of the Sixth International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*. IEEE Computer society, 52–59.

NAKAMURA, K. AND HOROWITZ, M. A. 1996. A 50% noise reduction interface using low-weight coding. In *Symposium on VLSI Circuits Digest of Technical Papers*. IEEE, 144–145.

NEDOVIC, N., OKLOBDZIJA, V. G., AND WALKER, W. W. 2003. A clock skew absorbing flip-flop. In *Proceedings of the International Solid-State Circuits Conference*. IEEE, 342–497.

NEEB, C., THUL, M., WEHN, N., NEEB, C., THUL, M., AND WEHN, N. 2005. Network-on-chip-centric approach to interleaving in high throughput channel decoders. In *International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1766–1769.

NIELSEN, S. F. AND SPARSØ, J. 2001. Analysis of low-power SoC interconnection networks. In *Proceedings of Nordchip 2001*. 77–86.

OBERG, J. 2003. *Clocking Strategies for Networks-on-Chip*. Kluwer Academic Publishers, 153–172.

OCPIP. 2003a. The importance of sockets in SoC design. White paper downloadable from http://www.ocpip.org.

OCPIP. 2003b. Open Core Protocol (OCP) Specifi cation, Release 2.0. http://www.ocpip.org.

OKLOBDZIJA, V. G. AND SPARSØ, J. 2002. Future directions in clocking multi-GHz systems. In *Proceedings of the 2002 International Symposium on Low Power Electronics and Design, 2002 (ISLPED '02)*. ACM, 219.

OSSO, M. D., BICCARI, G., GIOVANNINI, L., BERTOZZI, D., AND BENINI, L. 2003. ×pipes: a latency insensitive parameterized network-on-chip architecture for multi-processor SoCs. In *Proceedings of 21st International Conference on Computer Design (ICCD)*. IEEE Computer Society, 536–539.

OST, L., MELLO, A., PALMA, J., MORAES, F., AND CALAZANS, N. 2005. MAIA - a framework for networks on chip generation and verification. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE.

PANDE, P., GRECU, C., JONES, M., IVANOV, A., AND SALEH, R. 2005. Effect of traffic localization on energy dissipation in NoC-based interconnect. In *International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1774–1777.

PANDE, P. P., GRECU, C., IVANOV, A., AND SALEH, R. 2003. Design of a switch for network-on-chip applications. *IEEE International Symposium on Circuits and Systems (ISCAS) 5*, 217–220.

PEH, L.-S. AND DALLY, W. J. 1999. Flit-reservation flow control. In *Proceedings of the 6th International Symposium on High-Performance Computer Architecutre (HPCA)*. IEEE Computer Society, 73–84.

PEH, L.-S. AND DALLY, W. J. 2001. A delay model for router microarchitectures. *IEEE Micro 21*, 26–34.

PESTANA, S., RIJPKEMA, E., RADULESCU, A., GOOSSENS, K., AND GANGWAL, O. 2004. Cost-performance trade-offs in networks on chip: a simulation-based approach. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE, 764–769.

PHILIPS SEMICONDUCTORS. 2002. Device Transaction Level (DTL) Protocol Specification, Version 2.2.

PIGUET, C., JACQUES, HEER, C., O'CONNOR, I., AND SCHLICHTMANN, U. 2004. Extremely low-power logic. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*, C. Piguet, Ed. IEEE, 1530–1591.

PIRRETTI, M., LINK, G., BROOKS, R. R., VIJAYKRISHNAN, N., KANDEMIR, M., AND IRWIN, M. 2004. Fault tolerant algorithms for network-on-chip interconnect. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*. IEEE, 46–51.

RADULESCU, A., DIELISSEN, J., GOOSSENS, K., RIJPKEMA, E., AND WIELAGE, P. 2004. An efficient on-chip network interface offering guaranteed services, shared-memory abstraction, and flexible network configuration. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE, 878–883.

RIJPKEMA, E., GOOSSENS, K., AND WIELAGE, P. 2001. A router architecture for networks on silicon. In *Proceeding of the 2nd Workshop on Embedded Systems*. 181–188.

RIJPKEMA, E., GOOSSENS, K. G. W., RADULESCU, A., DIELISSEN, J., MEERBERGEN, J. V., WIELAGE, P., AND WATERLANDER, E. 2003. Trade offs in the design of a router with both guaranteed and best-effort services for networks-on-chip. In *Proceedings of the Design, Automation and Test in Europe Conference (DATE)*. IEEE, 350–355.

RIXNER, S., DALLY, W. J., KAPASI, U. J., KHAILANY, B., LÓPEZ-LAGUNAS, A., MATTSON, P. R., AND OWENS, J. D. 1998. A bandwidth-efficient architecture for media processing. In *Proceedings of the 31st Annual ACM/IEEE International Symposium on Microarchitecture*. 3–13.

ROSTISLAV, D., VISHNYAKOV, V., FRIEDMAN, E., AND GINOSAUR, R. 2005. An asynchronous router for multiple service levels networks on chip. In *Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*. IEEE, 44–53.

SATHE, S., WIKLUND, D., AND LIU, D. 2003. Design of a switching node (router) for on-chip networks. In *Proceedings of the Fifth International Conference on ASIC*. IEEE, 75–78.

SIA. 1997. National technology roadmap for semiconductors 1997. Tech. rep., Semiconductor Industry Association.

SIGUENZA-TORTOSA, D., AHONEN, T., AND NURMI, J. 2004. Issues in the development of a practical NoC: the Proteo concept. In *Integration, the VLSI Journal*. Elsevier, 95–105.

SIMUNIC, T. AND BOYD, S. 2002. Managing power consumption in networks-on-chips. In *Proceedings of the Design, Automation and Test in Europe Conference (DATE)*. IEEE Computer Society, 110–116.

SINGH, M. AND NOWICK, S. 2000. High-throughput asynchronous pipelines for fine-grain dynamic datapaths. In *Proceedings of the Sixth International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*. IEEE Computer Society, 198–209.

SPARSØ, J. AND FURBER, S. 2001. *Principles of Asynchronous Circuit Design*. Kluwer Academic Publishers, Boston.

STERGIOU, S., ANGIOLINI, F., CARTA, S., RAFFO, L., BERTOZZI, D., AND MICHELI, G. D. 2005. ×pipes lite: A synthesis oriented design library for networks on chips. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE.

SVENSSON, C. 2001. Optimum voltage swing on on-chip and off-chip interconect. Manuscript available on authors web page at http://www.ek.isy.liu.se/ christer/ManuscriptSwing.pdf.

SYLVESTER, D. AND KEUTZER, K. 2000. A global wiring paradigm for deep submicron design. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems 19*, 242–252.

SYSTEMC. 2002. The SystemC Version 2.0.1. Web Forum (www.systemc.org).

TAMIR, Y. AND FRAZIER, G. L. 1988. High-performance multiqueue buffers for VLSI communication switches. In *Proceedings of the 15th Annual International Symposium on Computer Architecture*. IEEE Computer Society, 343–354.

TAYLOR, M. B., KIM, J., MILLER, J., WENTZLAFF, D., GHODRAT, F., GREENWALD, B., HOFFMAN, H., JOHNSON, P., LEE, J.-W., LEE, W., MA, A., SARAF, A., SENESKI, M., SHNIDMAN, N., STRUMPEN, V., FRANK, M., AMARASINGHE, S., AND AGARWAL, A. 2002. The RAW microprocessor: A computational fabric for software circuits and general-purpose programs. *IEEE xxx*.

TORTOSA, D. A. AND NURMI, J. 2004. Packet scheduling in proteo network-on-chip. In *Parallel and Distributed Computing and Networks*. IASTED/ACTA Press, 116–121.

VAIDYA, R. S., SIVASUBRAMANIAM, A., AND DAS, C. R. 2001. Impact of virtual channels and adaptive routing on application performance. *IEEE Transactions on Parallel and Distributed Systems 12*, 2 (February), 223 – 237.

VARATKAR, G. AND MARCULESCU, R. 2002. Traffi c analysis for on-chip networks design of multimedia applications. In *Proceedings of the 39th Design Automation Conference (DAC)*. ACM, 795–800.

VSI ALLIANCE. 2000. Virtual component interface standard Version 2. Available from VSI Alliance (www.vsi.org).

WANG, H.-S., ZHU, X., PEH, L.-S., AND MALIK, S. 2002. Orion: a power-performance simulator for interconnection networks. In *Proceedings of the 35th Annual ACM/IEEE International Symposium on Microarchitecture*. IEEE Computer Society Press, 294–305.

WEBER, W.-D., CHOU, J., SWARBRICK, I., AND WINGARD, D. 2005. A quality-of-service mechanism for interconnection networks in system-on-chips. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE, 1232–1237.

WIEFERINK, A., KOGEL, T., LEUPERS, R., ASCHEID, G., MEYR, H., BRAUN, G., AND NOHL, A. 2004. A system level processor/communication co-exploration methodology for multi-processor system-on-chip platforms. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE Computer Society, 1256–1261.

WIELAGE, P. AND GOOSSENS, K. 2002. Networks on silicon: Blessing or nightmare? In *Proceedings of the Euromicro Symposium on Digital System Design (DSD)*. IEEE, 196–200.

WORM, F., THIRAN, P., MICHELI, G. D., AND IENNE, P. 2005. Self-calibrating networks-on-chip. In *International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2361–2364.

XANTHOPOULOS, T., BAILEY, D., GANGWAR, A., GOWAN, M., JAIN, A., AND PREWITT, B. 2001. The design and analysis of the clock distribution network for a 1.2 GHz alpha microprocessor. In *Digest of Technical Papers, IEEE International Solid-State Circuits Conference, 2001 ISSCC. 2001.* IEEE, 402 –403.

XU, J., WOLF, W., HENKEL, J., AND CHAKRADHAR, S. 2005. A methodology for design, modeling, and analysis of networks-on-chip. In *International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1778–1781.

XU, J., WOLF, W., HENKEL, J., CHAKRADHAR, S., AND LV, T. 2004. A case study in networks-on-chip design for embedded video. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE, 770–775.

ZHANG, H., GEORGE, V., AND RABAEY, J. M. 1999. Low-swing on-chip signaling techniques: Effectiveness and robustness. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems 8*, 3 (August), 264 – 272.

ZHANG, H., PRABHU, V., GEORGE, V., WAN, M., BENES, M., ABNOUS, A., AND RABAEY, J. M. 2000. A 1 V heterogeneous reconfi gurable processor IC for baseband wireless applications. In *International Solid-State Circuits Conference. Digest of Technical Papers (ISSCC)*. IEEE, 68–69.

ZIMMER, H. AND JANTSCH, A. 2003. A fault tolerant notation and error-control scheme for switch-to-switch busses in a network-on-chip. In *Proceedings of Conference on Hardware/Software Codesign and System Synthesis Conference CODES ISSS*. ACM, 188–193.

Paper B

# A Router Architecture for Connection-Oriented Service Guarantees in the MANGO Clockless Network-on-Chip

Tobias Bjerregaard and Jens Sparsø

Published in *Proceedings of the Design, Automation and Test in Europe conference*, Munich, IEEE 2005.

# A Router Architecture for Connection-Oriented Service Guarantees in the MANGO Clockless Network-on-Chip

Tobias Bjerregaard and Jens Sparsø
Informatics and Mathematical Modelling
Technical University of Denmark (DTU), 2800 Lyngby, Denmark
{tob, jsp}@imm.dtu.dk

## Abstract

*On-chip networks for future system-on-chip designs need simple, high performance implementations. In order to promote system-level integrity, guaranteed services (GS) need to be provided. We propose a network-on-chip (NoC) router architecture to support this, and demonstrate with a CMOS standard cell design. Our implementation is based on clockless circuit techniques, and thus inherently supports a modular, GALS-oriented design flow. Our router exploits virtual channels to provide connection-oriented GS, as well as connection-less best-effort (BE) routing. The architecture is highly flexible, in that support for different types of BE routing and GS arbitration can be easily plugged into the router.*

## 1 Introduction

Due to the effects of scaling micro-chip technologies, it has become unavoidably necessary to differentiate between local and global on-chip communication [18]. The concept of communication-centric design derives from the fact that global communication is becomming more and more costly, in relation to processing power. Physical issues of deep submicron technologies as well as design complexity issues make current, poorly scalable solutions for global on-chip communication, such as busses, unsuited for future large scale system-on-chip (SoC) designs. There is a general consensus that shared, segmented interconnection networks, so called networks-on-chip (NoC), constitute a viable solution space to the problems faced today [4][7][10]. NoC facilitates a truely modular and scalable design approach in which IP reuse is an integral part. This is called upon in order to make efficient use of the exploding amount of resources available to chip designers, and so within the constraints of ever shortening design cycles.

Chip-wide synchronous operation is becoming prohibitively difficult to achieve in large, high performance chips [1]. Possible solutions are in the direction of the *globally asynchronous locally synchronous* (GALS) concept [13], by which a chip consists of synchronous islands which communicate asynchronously. Since a NoC spans the entire chip, it seems natural not to attempt a globally synchronous implementation but instead use a GALS approach or even a fully asynchronous, i.e. clockless, realization. This is illustrated in Figure 1. The very nature of a clockless NoC also makes the integration of cores with different timing characteristics an integral part of the design flow, promoting the concept of modularity. Clockless circuits have a number of advantages over clocked ones. Since they make use of self-timed, data-driven control they operate at the maximum speed possible, and they have zero

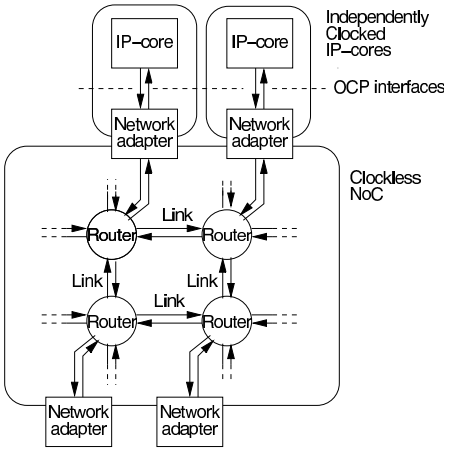



**Figure 1. A shared clockless network connecting independently clocked IP cores facilitates modularity in large scale SoC designs.**

Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'05)
1530-1591/05 $ 20.00 IEEE

dynamic power consumption when idle [17]. These advantages make them ideal for implementing NoC [14].

In this paper we present a worm-hole NoC router to be used in MANGO (*Message-passing Asynchronous Network-on-Chip providing Guaranteed services through OCP interfaces*) the NoC being developed at DTU. Our implementation is based on clockless circuit techniques. To our knowledge, MANGO is the first clockless NoC to provide connection-oriented guaranteed services (GS) as well as connection-less best-effort (BE) routing. It has features, area and performance comparable to a similar class of clocked routers. Additional benefits are an inherent support for GALS systems and zero dynamic idle power. The novelty lies in the concepts of establishing the GS paths and the application of methods we have developed in [5] to implement non-blocking behaviour amongst several independent streams in the shared part of the link/router. We demonstrate with a 0.12 $\mu$m CMOS standard cell implementation.

The paper focuses more on the router architecture and less on circuit realizations. In Section 2 we provide argumentation for the implementation of GS in NoC, and in Section 3 we provide an overview of MANGO. In Sections 4 and 5 we explain the details of the router architecture, and explain how it is possible to provide GS via connections. Finally, in Section 6 we describe our implementation and give a conclusion in Section 7.

## 2 Background

When designing NoCs, much research done in the area of multicomputer networks can be readily build upon. The trade-offs differ however, and while pincount and global wiring have been the main limiting factors in multicomputer networks, NoC designers face constraints on area and power. Also latency requirements are more stringent in tightly coupled single-chip systems. As a result, NoC demands simple solutions to ensure high speed and low area.

Traditionally multicomputer networks support best-effort (BE) routing. Much NoC research builds on this by improving on BE routing efficiency under the constraints of single-chip system design [15][12]. In particular the use of virtual channels (VCs) – logically independent channels sharing a physical link – is employed. We propose to use the VC routing resources somewhat differently, i.e. for establishing GS connections. While an improvement of BE routing simply makes better use of existing wires, connection-oriented GS routing adds explicit new features: packets travel along virtual circuits, avoiding the mutual influence that BE packets routed on the same logical network may experience. From an application-level perspective, the predictability that GS incurs promotes system integrity, and simplifies functional verification since there is less call for taking into account the complex dynamics of

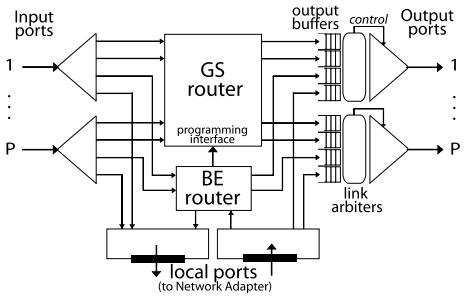



**Figure 2. The MANGO router conceptually.**

a shared communication structure. At the circuit level, the circuits needed to implement GS also turn out to be simpler than those needed for BE, since less arbitration is called for during routing control.

Other NoCs which implement GS are ÆTHEREAL [8] and NOSTRUM [11]. Both employ variants of time division multiplexing (TDM) for allocating bandwidth. TDM is not possible in a clockless NoC which has no notion of time, thus other approaches must be taken. In [9], a clockless NoC which provides differentiated services by prioritizing VCs is presented. Though this approach delivers improved latency for certain connections, no *hard* guarantees are provided. In [5] and [6] we have explored the use of VCs in clockless NoC for providing hard per connection bandwidth and latency guarantees respectively. In this paper we describe a supporting router architecture in which the proposed access schemes can be applied, providing end-to-end GS in a segmented network structure.

## 3 The MANGO Router

As shown in Figure 1, a MANGO NoC consists of network adapters (NA), routers and links. Each IP core is connected to the network through an NA, providing high level communication services, i.e. OCP transactions [2], on the basis of primitive services implemented by the network. Each NA, which also performs the synchronization between the clocked IP core and the clockless network, is connected to a router. The routers are connected by links in a grid-type structure, either homogeneous or heterogeneous. To keep speed up, long links can be implemented as pipelines.

Figure 2 shows a conceptual picture of the MANGO router. The router implements a number of uni-directional ports. Two of these are local ports which connect to the NA. The local ports consist of a number of physical interfaces. The remaining ports are network ports which, via point-to-point links, connect the router to neighboring routers. Each of these ports implement a number of independently

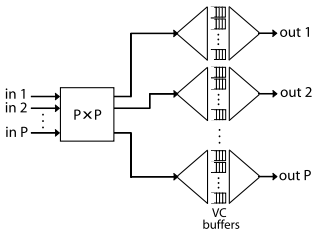



**Figure 3. Generic output buffered virtual channel router architecture.**

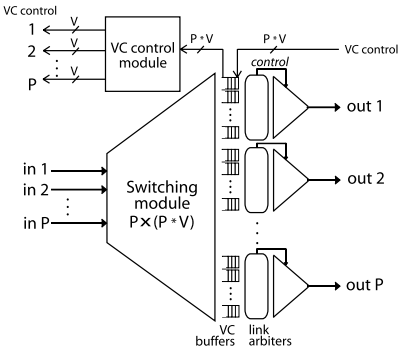



**Figure 4. The MANGO GS router implements non-blocking input to output switching.**

buffered VCs. Network ports and local ports implement the same type of interface. It is the function of the NA to translate communication to and from network packet format.

Internally, the router consists of a BE router, a GS router, output buffers, and link arbiters. The BE and the GS router are separately implemented. This is done in order to make the router modular. A similar approach is seen in the ÆTHEREAL router [16]. The BE router dynamically source-routes connection-less data packets, according to the routing path defined in the packet header. A subset of the VCs are allocated for BE routing. The GS router uses the remaining VCs to route header-less data streams on statically programmable connections. In order to make hard service guarantees, GS connections must be logically independent of other network traffic. In MANGO, a connection implements a logical point-to-point circuit between two different local ports in the network, by reserving a sequence of independently buffered VCs. The GS router provides non-blocking switching between the input ports and the output buffers, therefore *GS can be realized purely on the basis of link access arbitration*. This will become clear in the following sections. The GS connections are set up by programming these into the GS router via the BE router.

## 4 The GS Router

### 4.1 Overview

Figure 3 shows a generic output buffered VC router with $P$ input and output ports; a $P \times P$ switch is followed by a split module, which splits the incoming traffic to individual VC buffers at the output ports. Since several input ports may attempt to access the same output port simultaneously, congestion may occur. This makes the architecture unsuitable for providing service guarantees.

Figure 4 illustrates our connection-oriented GS router architecture. Again $P$ is the number of ports on each routing node, while $V$ is the number of VCs on each port. As touched upon in Section 3, a connection is a reserved sequence of VCs through the network. In the router, incomming flits (flowcontrol units) are guided through the switching module (Section 4.2) – by way of *steering bits* which are appended to the flits in the previous router – to the VC buffer that has been reserved for the given connection.

To prevent flits of different VCs from stalling and blocking each other on the links, VC control must be employed. This ensures that flits will only be transmitted on a link, if there is free buffer space in the target VC buffer. The VC control module (Section 4.3) establishes a VC control channel from the VC buffers back to the VC buffers in the previous router; a step back on a given connection. For each connection, a router thus stores the steering bits needed to guide flits to the VC buffer reserved for the connection in the *next* router, as well as *control channel bits* used to establish a VC control channel back to the VC buffer in the *previous* router. This information is programmed into the router using BE packets (via the programming interface, Figure 2). As seen, the setup information for each hop on a connection is stored in two places: one for the flit forward path, and one for the VC control reverse path. This overhead was accepted because it facilitates some very simple circuits.

The switching module is non-blocking. This means that the latency from the time that a flit is granted access to a link, until it arrives at the designated VC buffer in the following router, is constant. In order to provide hard GS for a connection there is thus only call for link access arbitration. The link arbiter (Section 4.4) is the key element in providing GS. It arbitrates amongst the VCs contending for access to the link, implementing the type of GS that is provided.

### 4.2 The Switching Module

The switching module makes it possible for all input ports to route flits to any combination of VC buffers. It

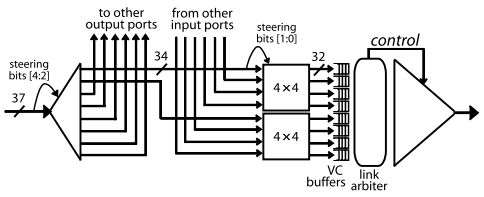



**Figure 5. Input/output port pair of the switching fabric of a 5x5 GS router with 8 VCs/port.**

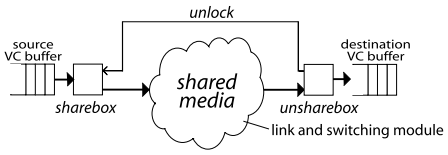



**Figure 6. Share-based VC control.**

is made entirely without any form of arbitration necessary, making its performance highly predictable and simplifying its design, saving both area and routing latency. Since a VC buffer is part of only one connection, no congestion will occur as only one input will attempt to route to one VC buffer at any given time. Figure 5 shows an input/output port pair in a $5 \times 5$ port router with eight VCs on each port. This configuration will form the basis for the implementation being used as an example of the architecture. The diagram indicates how the remaining ports are connected. An input port needs only connect to four output ports as it is not useful to route flits back in the direction they came from. The diagram also indicates how a total of five steering bits are used. The first three bits are used by a split module, which directs the flit to one of two $4 \times 4$ switches at each output port. At each switch two more steering bits are used to direct the flit to one of four VC buffers. At each stage, the steering bits used are stripped, as they are no longer needed.

The switching module, which consitutes a considerable part of the total router area, scales linearly with the number of VCs, and thus with the number of connections supported.

### 4.3 The VC Control Module

In this work we employ *share-based* VC control, which we have described in [5]. The scheme, illustrated in Figure 6, uses a single wire per VC to implement non-blocking access to a shared media, e.g. a link. When admitting a flit to the media the *sharebox* locks, not allowing further flits to pass. The flit passes across the media, to the *unsharebox* at the far side. The unsharebox implements a latch, into which the flit is accepted. When the flit in turn leaves the unsharebox, the *unlock* control wire toggles. This unlocks the sharebox, admitting another flit to the media. As long as the media is deadlock free, no flit will stall within it.

In the MANGO router, the link and the switching module constitute the shared media. Even if the link cycle for each flit transmitted on a VC is long, the full link bandwidth is exploited by the unlock handshake of different VCs overlapping. A single VC cannot utilize the full link bandwidth. However with an appropriate link access scheme, our goal to provide well defined worst case guarantees, can be met.

The cycle time of the VC link is sensitive to the forward latency of the flits. In this regard, clockless circuits are advantageous, since they make possible a very short forward latency per pipeline stage. An advantage of the share-based VC control scheme is that it is much cheaper, both area and power wise, than the commonly used credit-based VC control scheme. As shown in [5], the two schemes can be used together, controlling access to the same link. This is useful when the same link is used for both GS and BE routing. During BE routing it is clearly an advantage using credit-based VC control to improve the average case performance.

As indicated in Figure 4, each input port of the router implements an output unlock wire for each VC, likewise each output port implements an input unlock wire per VC. Under the share-based VC control scheme, the VC control module establishes control channels between arbitrary VC buffers and the inputs, simply by circuit switching the unlock wires, originating in the VC buffers, unto the VC control outputs at the input ports. Thus the VC control module implements a non-blocking $(P * V) \times (P * V)$ switch. The control channel bits stored in the router map the VC buffers to the appropriate input VC wires, according to the connection path. In our $5 \times 5$, 8 VC router we simply use $5 * 8$ instantiations of a $(5 - 1) * 8$-input multiplexer. For larger number of VCs, it might prove worthwhile to implement a more complex switch structure, e.g. a Clos network, which scales better. The mapping between input and output VCs can be considered static during connection usage.

### 4.4 Link Access

The MANGO router implements output buffers. Since a connection is a reserved sequence of VCs, the target VC buffer of a flit entering the router is deterministic. There is no need for arbitrating amongst VCs on different output ports. Placing the VC buffers at the output simplifies the link access arbitration scheme. Thus simple, fast and low area circuits can be used to implement access schemes. Since the path across the link, through the switching module, to the target VC buffer is fully deterministic and congestion free, by way of the non-blocking switching module described in Section 4.2, the link access is the only critical point on a connection, with regard to congestion. If guaran-

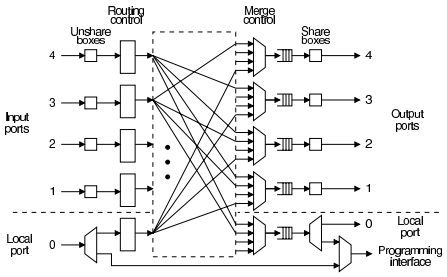



**Figure 7. The BE router.**

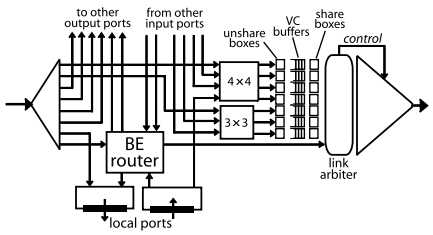



**Figure 8. The BE router is integrated into the GS router, using a subset of the VCs.**

tees can be made for link access, guarantees can be made for the entire connection. Thus link access arbitration is what facilitates GS in MANGO.

The *link arbiter* shown in Figure 2, 4, 5 and 8 implement the arbitration scheme. According to the arbitration amongst contending VCs, the flits are merged unto the shared link, which connects to the neighboring router. The steering bits which indicate the destination of the flit are appended before the flit is transmitted. In the router implementation presented herein, as a demonstration of the general architecture, we have implemented a fair-share access scheme [5], by which each of 8 VCs are guaranteed a minimum of one 8th of the total link bandwidth. To keep the area down, our output buffers are a single flit deep plus one flit in the unsharebox. This is enough to ensure the fair-share scheme to function over a sequence of links, providing a hard lower bound on the throughput of a connection. If a VC does not use its allocated bandwidth, the link is automatically used by another contending VC.

It is seen how the GS routing functionality is decoupled from the switching functionality in the router. This makes it an easy and modular task to instantiate new GS schemes.

## 5 The BE Router

The BE router, shown in Figure 7, implements a simple source routing scheme. The first flit of a packet is the header flit. The two MSBs of the header indicate one of four output ports. Choosing a direction back to where it came from, the packet is routed to the local port. The header is then rotated two bits, positioning the header bits for the next hop. With 32-bit flits, a packet can make a total of 15 hops. The packets are variable length; a control bit is used to indicate the last flit. The outputs arbitrate fairly between input ports contending for access. Once an input port has gained access, it will retain this until the last flit has been routed, thus keeping packet coherency. The interface used to program GS connections into the GS router, is implemented as an extention on port 0, the local port. Figure 8 shows a complete picture of the router, indicating how the BE router is integrated into the GS router architecture. To avoid deadlocks XY-routing is employed.

When a flit enters the BE router, three steering bits have been stripped, and a total of 34 bits remain (see Figure 5), 32 of which are flit data. Of the other two, one is the control bit used to indicate the last flit. The remaining bit can be used to indicate one of two BE VCs. This is not used in the present implementation, but can be used to extend the BE router to provide more complex deadlock free routing, adaptive VC allocation, etc. The VC control of the BE channels is handled separately from the VC control module and can be implemented using a credit-based scheme [5].

The focus of this paper is means of providing GS by way of the non-blocking GS router. The BE router has been included for the sake of completeness and holds lots of potential for improvement. First of all it should implement credit-based VC control, in order to facilitate improved average case performance. The modular architecture of the MANGO router makes it easy to plug in a new BE router.

## 6 Implementation and Results

We have implemented a 32-bit MANGO router using 0.12 $\mu$m CMOS standard cells. The router is designed with 4-phase bundled data clockless control circuits. It is a relatively small entity, thus internal timing is quite predictable. The links between neighboring routers are much longer, and thus more sensitive to timing variations. In order to make assembling a NoC-based SoC a modular and timing safe exercise, and in order to save power, we advocate delay insensitive signaling between routers, e.g. 1-of-4 signaling [3]. This will be realized in future MANGO versions.

The router has local input and output ports providing 5 interfaces each (4 for GS and 1 for BE) and 4 input and 4 output network ports (each implementing 8 VCs). The router simultaneously supports connection-less BE routing plus a total of 32 independently buffered GS connections. Each VC provides one 8th of the bandwidth on a link.

**Table 1. Area usage in the MANGO router.**

| Module | Area |
|---|---|
| Connection table | 0.005 $mm^2$ |
| Switching module | 0.065 $mm^2$ |
| VC buffers | 0.047 $mm^2$ |
| Link access | 0.022 $mm^2$ |
| VC control | 0.016 $mm^2$ |
| BE router | 0.033 $mm^2$ |
| **Total** | **0.188 $mm^2$** |

The performance in netlist simulations using worst-case timing parameters (1.08 V/125 C) was 515 MHz per port (795 MHz under typical timing conditions). The pre-layout area was 0.188 $mm^2$. The area usage is detailed in Table 1. The switching module and the VC buffers together account for more than half of the total area. Much area could be saved by using custom-designed buffers. As a quick comparison, a 0.13 $\mu$m instantiation of the globally synchronous ÆTHEREAL NoC, which also provides per connection bandwidth guarantees, had a port speed of 500 MHz and a laidout area of 0.175 $mm^2$ [8], using custom hardware FIFOs. The ÆTHEREAL router supports up to 256 connections. These are however not independently buffered – a key feature of the MANGO router connections. Thus end-to-end flowcontrol is needed, e.g. using credits. In the MANGO architecture end-to-end flowcontrol is inherent. Also, in order to save area, routing infomation is not stored locally in ÆTHEREAL. Connections thus need to support the routing overhead of a packet header. This is unwarranted in the architecture presented herein.

## 7 Conclusion

In this paper, we have presented MANGO (*Message-passing Asynchronous Network-on-Chip providing Guaranteed services through OCP interfaces*), the NoC being developed at DTU. The MANGO router is implemented using clockless circuit techniques and thus inherently supports a modular, GALS oriented design flow. It exploits virtual channels to provide connection-oriented service guarantees (GS), as well as connection-less best-effort routing. We view the predictability of GS as a way to promote system-level integrity. A 32-bit $5\times5$ MANGO router using 0.12 $\mu$m CMOS standard cells was implemented, which supports 32 independently buffered connections, has a worst-case port speed of 515 MHz and a pre-layout area of 0.188 $mm^2$.

## References


[1] International technology roadmap for semiconductors (ITRS) 2001. Technical report, International Technology Roadmap for Semiconductors, 2001.

[2] Open Core Protocol Specifi cation, Release 2.0, 2003.

[3] W. Bainbridge and S. Furber. Delay insensitive system-on-chip interconnect using 1-of-4 data encoding. In *Proceedings of the 7th International Symposium on Asynchronous Circuits and Systems*, pages 118 – 126, March 2001.

[4] L. Benini and G. D. Micheli. Networks on chips: A new SoC paradigm. *IEEE Computer*, 35(1):70 – 78, January 2002.

[5] T. Bjerregaard and J. Sparsø. Virtual channel designs for guaranteeing bandwidth in asynchronous network-on-chip. In *Proceedings of the IEEE Norchip Conference*, 2004.

[6] T. Bjerregaard and J. Sparsø. A scheduling discipline for latency and bandwidth guarantees in asynchronous network-on-chip. In *Proceedings of the 11th International Symposium on Advanced Research in Asynchronous Circuits and Systems*. IEEE, 2005.

[7] W. J. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *Proceedings of the 38th Design Automation Conference*, pages 684–689, June 2001.

[8] J. Dielissen, A. Radulescu, K. Goossens, and E. Rijpkema. Concepts and implementation of the phillips network-on-chip. In *Proceedings of the IPSOC'03*, 2003.

[9] T. Felicijan and S. B. Furber. An asynchronous on-chip network router with quality-of-service (QoS) support. In *Proceedings IEEE International SOC Conference*, pages 274–277. IEEE, 2004.

[10] A. Jantsch and H. Tenhunen. *Networks on Chip*. Kluwer Academic Publishers, 2003.

[11] M. MillBerg, E. Nilsson, R. Thid, and A. Jantsch. Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip. In *Proceedings of the Design, Automation and Testing in Europe Conference (DATE'04)*. IEEE, 2004.

[12] R. Mullins, A. West, and S. Moore. Low-latency virtual-channel routers for on-chip networks. In *Proceedings of the International Symposium on Computer Architecture (ISCA) 2004*. IEEE, 2004.

[13] J. Muttersbach, T. Villiger, K. Kaeslin, N. Felber, and W. Fichtner. Globally-Asynchronous Locally-Synchronous Architectures to Simplify the Design of On-CHIP Systems. In *Proc. 12th International ASIC/SOC Conference*, pages 317–321, Sept. 1999.

[14] S. F. Nielsen and J. Sparsø. Analysis of low-power SoC interconnection networks. In *Proceedings of Nordchip 2001*, pages 77–86, 2001.

[15] L.-S. Peh and W. J. Dally. A delay model for router microarchitectures. *IEEE Micro*, 21:26 – 34, 2001.

[16] E. Rijpkema, K. G. W. Goossens, A. Radulescu, J. Dielissen, J. V. Meerbergen, P. Wielage, and E. Waterlander. Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip. In *Proceedings of the Design, Automation and Test in Europe Conference (DATE'03)*, pages 350–355. IEEE, 2003.

[17] J. Sparsø and S. Furber. *Principles of Asynchronous Circuit Design - a Systems Perspective*. Kluwer Academic Publishers, Boston, 2001.

[18] D. Sylvester and K. Keutzer. A global wiring paradigm for deep submicron design. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, 19:242–252, 2000.

Paper C

# Implementation of Guaranteed Services in the MANGO Clockless Network-on-Chip

Tobias Bjerregaard and Jens Sparsø

Submitted to *IEE Proceedings: Computers and Digital Techniques*, IEE 2005.

# Implementation of Guaranteed Services in the MANGO Clockless Network-on-Chip

Tobias Bjerregaard and Jens Sparsø
Technical University of Denmark (DTU)
Informatics and Mathematical Modelling
2800 Lyngby, Denmark
web: www.imm.dtu.dk/~ {tob, jsp}

**Abstract**

Shared, segmented, on-chip interconnection networks, known as networks-on-chip (NoC), may become the preferred way of interconnecting IP cores in future giga-scale system-on-chip (SoC) designs. A NoC can provide the required communication bandwidth while accommodating the effects of scaling microchip technologies. Equally important, a NoC facilitates a truly modular and scalable design flow.

In this paper we present the MANGO NoC, and we explain how its key characteristics (clockless implementation, standard socket access points and guaranteed communication services) make MANGO suitable for a modular SoC design flow. Among the advantages of using clockless circuit techniques are inherent global timing closure, low forward latency in pipelines, and zero dynamic idle power consumption. Time division multiplexing, generally used to provide bandwidth guarantees in clocked NoCs, however is not possible in a clockless environment. MANGO provides an alternative, high-performance solution to providing hard, connection-oriented service guarantees, using clockless circuit techniques. We present in-depth circuit details and describe the 0.13 $\mu$m standard cell implementation of a $5 \times 5$ routing node, for use in a mesh type NoC.

## 1 Introduction

Developers of giga-scale SoC designs encounter a range of challenges. Each new generation of microchip fabrication technologies makes available an increased amount of on-chip resources in terms of transistors and wiring layers, and facing reduced design cycles engineers must increase their design productivity at a dramatic rate. Meanwhile the effects of technology scaling degrade the performance of on-chip wires, relative to that of transistors. These effects, together with the increased routing congestion in large complex systems, make it unavoidable to differentiate between local and global on-chip communication [1, 2, 3]. Current solutions for global on-chip communication – such as busses – scale

poorly and are unsuited for the requirements set by future giga-scale SoC design. The demand for IP reuse and system level scalability is growing, and the design methodologies in use today are inadequately geared for dealing with the problems at hand. There is a general consensus that shared, segmented interconnection structures, so called networks-on-chip (NoC) [4, 5, 6] constitute a viable solution space to emerging SoC design challenges. NoCs hold the potential to accommodate both design flow issues as well as physical level issues of giga-scale SoC design.

There is a wide spectrum of advantages of NoCs as a replacement for global wiring in single-chip systems. Dynamic allocation of shared data routing resources facilitates modularity and reconfigurability, better wire utilization through sharing reduces wiring congestion and power consumption, and a segmented architecture enables parallelism through pipelining and counteracts physical level issues of scaling process technologies.

In this paper we introduce MANGO (*Message-passing Asynchronous Network-on-chip providing Guaranteed services over OCP interfaces*). Key features of MANGO include an asynchronous, or *clockless*, implementation, memory mapped read/write style standard socket access points, and connection-oriented end-to-end communication service guarantees. The basic router architecture was first presented in [7], and network adapters which implement the access points were presented in [8]. In [9] we presented the implementation of virtual channels (VCs) – independently buffered channels sharing a single physical link [10] – and showed how these can be used to provide bandwidth sharing of the links. Contributions of the work presented herein are improved, high-performance clockless circuits for the implementation of MANGO links, and a detailed description of how MANGO uses VCs to implement hard, connection-oriented, end-to-end bandwidth guarantees, by way of virtual circuit-switching. The novelty of MANGO is contained particularly in its combination of a clockless implementation and its provision of hard service guarantees. The architecture is demonstrated with a 0.13 $\mu$m CMOS standard cell implementation of a $5 \times 5$ routing node.

The paper is organized as follows: in Section 2 we provide motivation for the implementation of service guarantees in NoCs, and present some related work. In Section 3 we give an overview of the key characteristics of MANGO. In Section 4 we detail the circuit implementation of VC control used in the MANGO links. Section 5 details circuits of the link access circuits (which quantify the service guarantees provided) and the 2-phase delay insensitive link pipeline. Section 6 describes the router design, Section 7 presents the demonstration router implementation, and Section 8 concludes the paper.

## 2 Background

In order to limit the need for area costly data-buffering registers, NoCs generally employ wormhole routing rather than e.g. store-and-forward routing. In wormhole routers for parallel computers [11, 12] VCs have traditionally been used to

improve the performance of best-effort (BE) traffic [13, 14]. Much NoC research builds on this by implementing VC routers under the constraints of single-chip system design [15, 16]. The trade-offs differ from those of parallel computer systems. While chip pin count and inter-chip wiring have been the main limiting factors in parallel computer networks, NoC designers face constraints on area and power.

BE networks route packets on a shared set of VCs. Adaptive routing mechanisms can be employed, to always select the most optimal of the free VCs on a given set of links on the route. Ultimately however, a BE packet may be blocked by other packets contending for access to the shared routing resources (buffers or VCs). Hence the performance of BE routing networks is very difficult – if not impossible – to predict. Designers of parallel computer networks have focussed mainly on improving average performance. While this performance aspect is naturally also important, NoC design additionally encompasses general aspects of SoC design, e.g. issues related to modularity, IP reuse and timing closure. Recent NoC research [17, 18, 8] has proposed the use of standard sockets such as OCP [19] and AXI [20] at the network access points. In doing so, the first step towards a higher degree of modularity in the SoC design flow is taken. IP cores can be plugged into the network, and the top-level data flow can in turn be specified dynamically, using the shared network. However, for true modularity a high level of predictability is also required. This is all the more true the larger the system is, as dynamic dependencies between traffic streams, quickly become very complex. In our view, hard performance guarantees are essential in performing system-level verification of large heterogeneous systems. A subsystem composed of communicating IP cores should behave in a predictable manner, independently of the interaction between other IP cores in the system. A NoC providing guaranteed services (GS) facilitates this. GS is thus not only a means of meeting real-time constraints. It also has large implications on the design flow by enhancing modularity and by enabling *analytical* rather than *statistical* system verification. This view has also been argued for in [21] and [22].

It is important to distinguish between *hard* and *soft* service guarantees. Soft guarantees, usually based on priority scheduling in connection-less routing protocols, is a variant of BE routing and may offer improved performance. This could be low latency for interrupts, or high bandwidth for data streams. But since the routing is connection-less and packets are routed on a shared set of VCs, unless a global view of the traffic in the network is known, only statistical guarantees can be provided. By hard guarantees, we refer to hard bounds on latency and bandwidth, irrespective of other data traffic. In order to provide such guarantees, connection-oriented routing is absolutely essential. This is so because a data stream must be logically independent from all other traffic in the network. This can be ensured by circuit-switching (virtual, time multiplexed or otherwise). Instead of using VCs for adaptively routing packets, in order to improve average performance, we propose to use VCs for establishing dedicated, logically circuit-switched end-to-end connections. In [9] and [23] we have explored the use of VCs in NoCs for providing hard per connection bandwidth and latency guarantees.

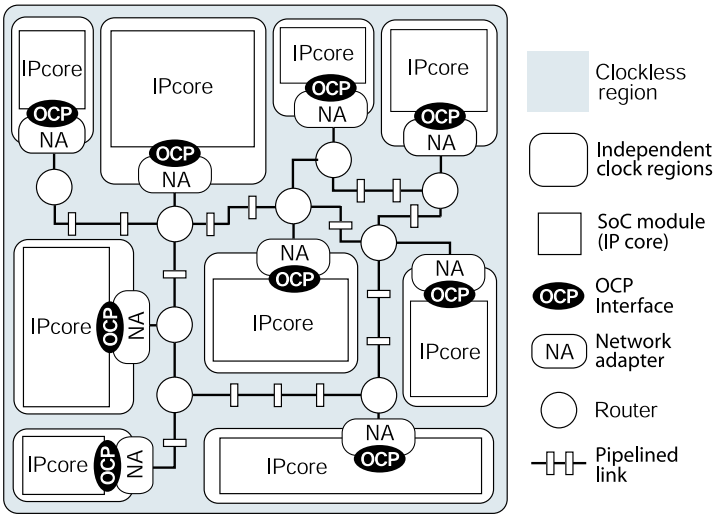



Figure 1: A MANGO-based SoC.

Another issue in the design of a NoC is that of clocking and synchronization. A SoC typically includes a number of clock domains and problems related to clock distribution and global timing closure are becoming increasingly difficult to handle in large scale chip designs. Spanning an entire chip, a NoC design must also address these issues. In MANGO, we adopt a *Globally Asynchronous Locally Synchronous* system view (GALS) [24, 25]. A GALS-based system allows IP cores to run asynchronously, i.e. at different clock speeds. This can be handled by inserting clock domain crossing FIFOs between different clock regions on the chip, an approach adapted by the commercial Arteris NoC [26]. Another approach is to run the NoC at its own clock speed, then synchronizing with the cores at the network access points, as done in the ÆTHEREAL NoC [27]. We have chosen the approach of implementing the network entirely using clockless circuits. This approach eases a modular design flow by inherently ensuring global timing closure. Clockless circuits have a number of additional advantages which will be detailed in Section 3.1.1.

Previously published NoCs which provide hard connection-oriented service guarantees are all based on synchronous circuit techniques, employing variants of time division multiplexing. These include ÆTHEREAL [17, 22], NOSTRUM [28] and aSOC [29]. Clockless NoCs include CHAIN [30], QNoC [31] and ANoC [32]. While QNoC and ANoC allow routing with different priority levels, neither supports dedicated connections with hard performance guarantees.

# 3 MANGO overview

Figure 1 illustrates a MANGO-based SoC. The network consists of network adapters (NAs) implementing the network access points, routers and pipelined links. In this section, we will first describe characterising traits of MANGO, then explain the basic GS connection functionality.

## 3.1 Characterising Traits

Three characterising traits of MANGO are (i) clockless implementation of the core of the network, i.e. the routers and links, (ii) network access points compatible with standard (clocked) sockets and (iii) connection-oriented communication services providing hard performance guarantees. These three traits are all key features in facilitating a modular design flow.

### 3.1.1 Clockless Implementation

Increasingly, large SoCs are being implemented using a GALS organization. Most published NoCs assume a synchronous or mesochronous implementation with all routers and links clocked at the same frequency. This implies chip-wide distribution of a common, high-speed clocking signal.

MANGO is different in that the entire NoC is implemented using clockless circuit techniques. The very nature of a clockless NoC makes the integration of cores with different timing characteristics an integral part of the design flow, promoting the concept of modularity. Also, in a globally asynchronous system, timing closure becomes less problematic since it is reduced to a local problem. Furthermore clockless circuits have a number of characteristics which can be exploited to advantage when implementing a NoC [33]: (1) Since they make use of self-timed, data-driven control based on local handshaking they always operate at the maximum speed possible, and they can be pipelined more aggressively. (2) For a NoC, high speed or high performance means not only a high throughput but also a low end-to-end latency. A clockless NoC provides also the latter, and there are two reasons for this. First a communication transaction involves fewer clock domain crossings and hence fewer synchronization events, and secondly an end-to-end connection behaves like a fall-through FIFO whereas a clocked NoC behaves like a clocked shift register. (3) A clockless NoC has zero dynamic power consumption in those parts of the NoC which are idle. In real life applications where communication requirements fluctuate a lot this leads to a lower power consumption. (4) In some contexts the handshaking in clockless circuits constitutes an overhead. In a NoC however, it inherently provides local flow control, a mechanism which would have to be added to a clocked design. (5) A clockless NoC avoids the need for a (power hungry) global clock tree and it leads to a more evenly spread power surge due to the globally asynchronous operation. This results in less strain on power wires and reduced switching noise [34, 35].

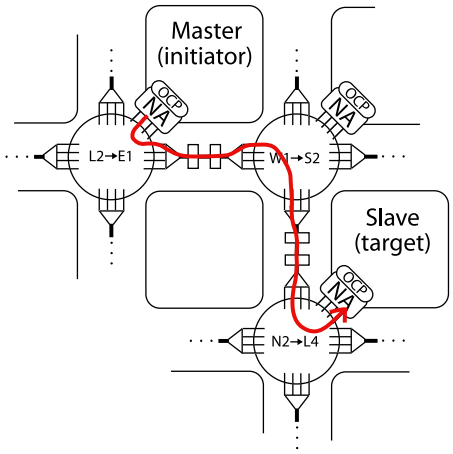



Figure 2: A virtual circuit implements a path through the network by reserving a sequence of virtual channels from source to destination. At each routing node, the path is set up by mapping from input VC to output VC: first local 2 to east 1, then west 1 to south 2, and finally north 2 to local 4.

The routers in MANGO are implemented using the 4-phase bundled data handshake protocol, in order to minimize the area. The links are implemented using the 2-phase dual-rail delay insensitive protocol, in order to obtain global timing robustness.

### 3.1.2 OCP Compliant Network Adapters (NAs)

In the NAs the clockless network is synchronized to the local clock regions. The network access points are implemented as standard OCP sockets [19]. Thus any core which is OCP compliant can be plugged into the system. This allows cumberless IP reuse. OCP is a family of synchronous point-to-point interfaces which implement a memory mapped, read/write-style interface. It supports a range of different transaction modes such as simple read/write, burst read/write, threading, etc. The MANGO NAs support all the required OCP v2.0 basic signals and a consistent subset of burst, thread and interrupt extensions, allowing support for single reads and writes, single-request burst reads and writes, threads, connections and interrupts [8].

The basic datagram of network-level flow control is called a *flit* (**fl**ow control un**it**) [10]. In the NA, the OCP transactions are packetized and transmitted across the network as a stream of flits. At the receiving end, packets are reassembled and unwrapped. Responses are also packetized and returned across the network.

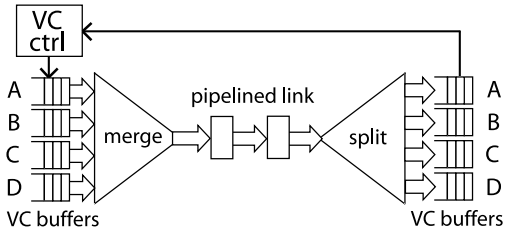



Figure 3: Basic structure of a link in which several virtual channels (VCs) A to D contend for access to a shared, pipelined link.

#### 3.1.3 Guaranteed Services

MANGO allows the reservation of end-to-end connections, between two access points of the network. Service guarantees, in terms of bandwidth and latency, can be made on these connections. In addition to connection-oriented GS circuit switching, MANGO supports connection-less BE routing. GS and BE routing share the same physical links, thus improving the link utilization.

In order to provide hard performance guarantees, it is absolutely essential that a connection is logically independent of other connections. In synchronous networks, this can be ensured by temporally separating data streams, using time division multiplexing. We have chosen to implement MANGO using clockless circuit techniques because of a number of advantages mentioned above. Since an absolute sense of time does not exist in clockless circuits however, we must separate the virtual circuits in a different manner. In this work we present a solution to this in a clockless environment.

### 3.2 VC Links and Connections

GS connections are provided on virtual circuits. A virtual circuit is implemented by an exclusively reserved sequence of VCs, and is thus in effect logically separated from other traffic in the network. Figure 2 shows how a virtual circuit is established as a path through the network, the path being determined by a mapping between input VCs and output VCs in each routing node in the path. The circuit is independently buffered, and thus implements a distributed FIFO between source and destination. End-to-end flow control is thereby inherently handled by a series of local handshakes between neighboring nodes. By arbitrating between VCs contending for access to a link, it is possible to make bandwidth and latency guarantees for the traffic on the these [9, 23]. If for instance a bandwidth of B units is guaranteed on each VC in a virtual circuit, the end-to-end guarantee is also B units, assuming that the links – not the routers – are the bottlenecks on the circuit. In Section 6.1 we shall see how this is the case in the MANGO architecture.

Figure 3 illustrates the basic MANGO link structure, indicating how a number of VCs – A to D – share a physical link; the goal is to save on global wiring

resources. Also power can be saved; first, a high utilization of the large link drivers and other active circuitry make for a lower proportion of leakage power, and secondly a heavily used link justifies the overhead of implementing power saving data encoding schemes. The VC control makes sure that when a VC stalls, other VCs are not blocked. A flit is allowed to enter the pipeline only if it can be consumed by its VC buffer at the receiving end of the link.

There are two main tasks in designing such a network: link design and switch design. The VC link is the basic building block of GS connections in MANGO. Implementing this conceptually simple link using clockless circuit techniques, there are a number of interesting aspects and trade-offs to consider. The links are the performance bottleneck, hence a considerable effort has been put into optimizing them. For a complete picture, we also provide an overview of the switch design, in Section 6.

# 4 Virtual Channel Design

In this and the next section we present circuit details of the MANGO links. First we describe how we have implemented two types of VC control used in MANGO. In Section 5 we then address the topic of link access and link pipeline design.

In future technologies, global wires are projected to become more costly relative to processing logic, both with regards to power consumption and area usage [1]. Thus the sharing of long global wires will become increasingly advantageous. In [36] it is concluded that by far the major part power usage in the Nostrum NoC is dissipated in the links, only a few percent being dissipated in the routers and network adapters. This can be seen as a general trend for NoC. Hence the optimal implementation of these links is of utmost importance.

Referring to Figure 3; flits arising from the contending VCs A to D must be merged and transmitted across the shared physical link, and finally split out to their respective channel buffers. If a flit stalls on the link however, because the data buffers at the receiving end are full, other channels making use of the link could be blocked. Non-blocking behavior is an essential characteristic of VCs, thus controlling the access of flits to the shared link is necessary. In this section we will present two different clockless VC control methods, and their circuit implementation.

## 4.1 Clockless VC Design

Requirements to the VC control circuits are that they should (i) be modular, in order to facilitate easy integration into a variety of systems, (ii) have high performance, in order not to be the bottleneck and (iii) have low cost, in terms of power, circuit area and wiring.

Figure 4 illustrates how the handshakes of the individual VCs overlap, as the flits arising from each are interleaved on the link. If each VC implements only a single flit buffer at the receiving end, this buffer must be empty before a flit

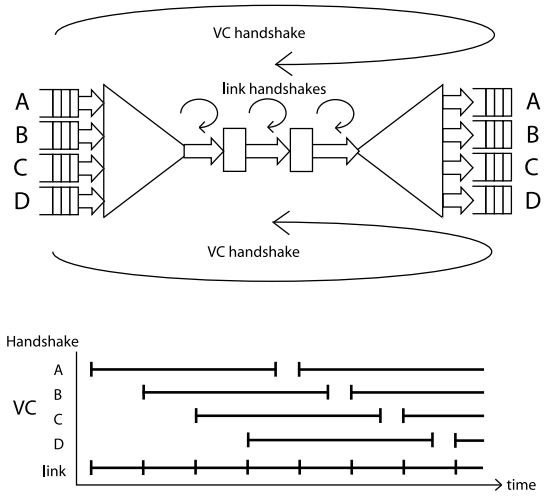



Figure 4: Overlapping VC handshakes in order to maximize link utilization.

can be transmitted. This flit must then arrive at the buffer – and leave it again – before a new flit can be transmitted. The long VC handshake in the figure illustrates this requirement. While the handshakes on the pipelined link are fast, several VC handshakes may overlap, utilizing the full link bandwidth. If deeper buffers are implemented, several flits can be in-flight from each VC. This however has an area cost. It may improve best-case performance, as the link can be utilized better by a single active VC. However, if we assume e.g. a fair-share scheduling of the VCs contending for access to the link, each VC may never need more than a single buffer, if all VCs are active and the VC handshake is faster than the link handshake times the number of VCs. In this case, the worst-case performance for each VC thus does not improve by increasing the buffer depth. Hence, implementing deeper buffers does not change the performance *guarantee* that can be made for a VC.

In a clocked implementation of the VC link illustrated in Figure 4 the pipeline depth must be smaller than the number of VCs, in order to fully utilize the link. In a clockless circuit, the timing is not this simple [37]. The forward latency through a pipeline stage is only a fraction of the handshake cycle time. Hence the link can be pipelined much deeper than a clocked implementation with an equal number of VCs.

In the following we will present two methods of VC access control, which we refer to as *lock-based*[1] and *credit-based* [9]. A lock-based VC is characterized by a low area and power overhead. A credit-based VC allows several in-flight flits per VC, and hence a better link utilization if only a single VC is contending for access to the link. The methods are modular, wrapping around the shared parts

[1]Our earlier works [9], [23] and [7] refer to lock-based as *share*-based VCs

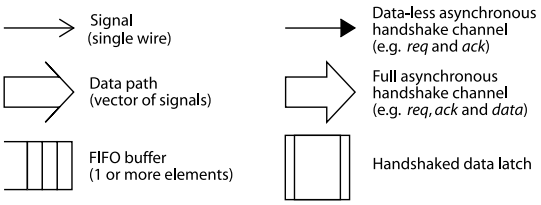



Figure 5: Symbols for signal, handshake channels and basic elements used in the figures of this paper.

of the link, and lock- and credit-based VCs can be used together on the same link. In MANGO, low overhead lock-based VCs are used for GS connections, the aim being to provide hard *lower* bounds on performance. Credit-based VCs are used for connection-less BE routing, which benefits from a better average performance.

While any asynchronous handshake scheme can be chosen, the circuits presented herein are mainly based on the 4-phase bundled-data push protocol [37]: (1) The sender asserts a *request* (req) signal indicating valid data. (2) The receiver acknowledges data reception by raising an *acknowledge* (ack) signal. After this follows a return-to-zero sequence, during which (3) the req, then (4) the ack is lowered. These handshake control signals, the data and the protocol by which they abide, constitute an asynchronous *handshake channel* (not to be confused with a network *communication channel*, e.g. a VC). Figure 5 shows the symbols used in the figures throughout this paper.

Before going into details of our VC implementations, we will describe the *output decoupler*, a simple flow control element which is used extensively throughout MANGO. The element decouples the handshake on its input from that on its output.

## 4.2 Output Decoupler

The output decoupler, shown in Figure 6, is a latch-less flow control element which is inserted in the control path of a handshake channel. It can be useful, in the case where a consumer has pending tasks that cannot commence until its input channel has completed its handshake cycle. When decoupling the producer (e.g. a VC input) from the consumer (e.g. a shared link), the consumer does not need to wait for a potentially slow producer to finish its handshake. Rather, it can quickly begin servicing another producer. In MANGO, the element is used for decoupling paths in circuits implementing nested synchronization, e.g. in the overlapping VC handshakes of Figure 4 and in the link access circuits to be described in Section 5.1. More details will be presented in the sections describing circuits making use of the element.

Figure 7 shows the output decoupler signal transition graph specification and its circuit schematic, as synthesized using the Petrify asynchronous synthesis

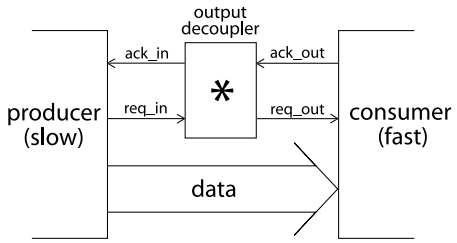



Figure 6: A latch-less output decoupler, placed in the handshake path between a slow producer and a fast consumer, allows the consumer to complete its handshake independently of the producer.

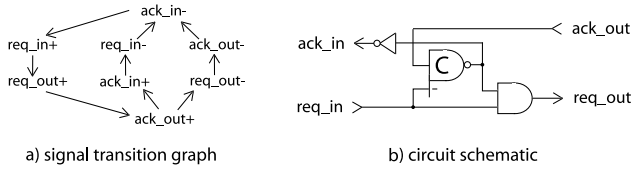



Figure 7: Output decoupler a) signal transition graph, b) circuit schematic.

tool [38, 39]. When a producer module at its input asserts *req_in*, the output decoupler immediately asserts *req_out*. When the consumer acknowledges, by asserting *ack_out*, the remaining handshake signaling is performed independently on the input and output channels. Hence the output can complete quickly, even when the input is slow.

The output decoupler can be used with or without a data path. If a data path is inserted, as shown in Figure 6, the data-validity of the handshake channel, i.e. the time interval during which the data is valid with respect to the control signals [37], can only be improved – never degraded – after passing the element.

## 4.3 Lock-based VCs

Figure 8 shows a lock-based VC link with 4 VCs, A through D. The figure also indicates some control signal names in the link (italicized). These refer to signal names in simulation wave diagrams to be presented in Section 4.5. At each VC input, a *lockbox* controls access to the merge, and hence to the shared link. After letting a flit through the box locks, blocking any further flits from passing on that particular VC. The *unlockbox* at the link output implements a buffer, and the flit will be able to leave the shared part of the link once it has crossed it, being accepted into this buffer. Thus, a flit which has been granted access to the link will never stall, blocking other flits. When the flit in turn leaves the unlockbox, freeing its buffer, the lockbox is unlocked by the unlockbox toggling the *unlock* wire.

Figure 9 shows the internals of a lock-unlockbox pair. When a transfer starts

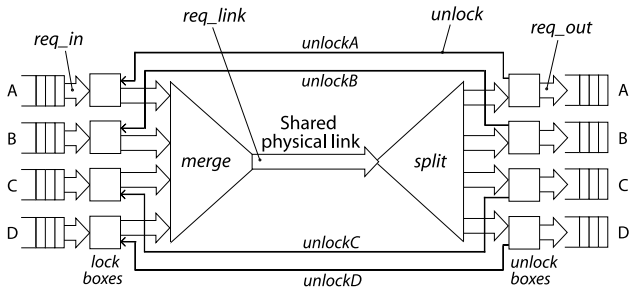



Figure 8: Virtual channel link based on the lock-unlock method.

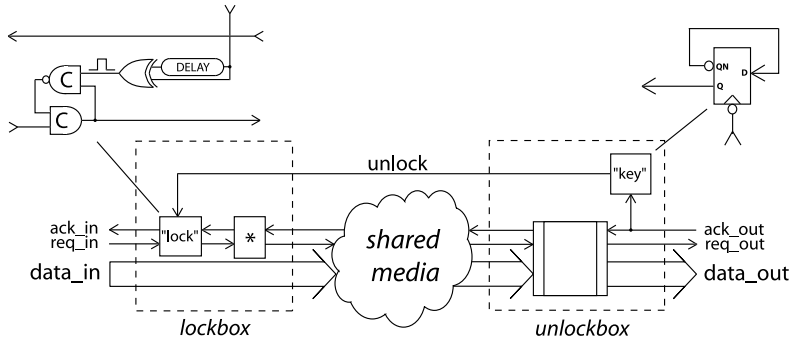



Figure 9: Lock- and unlockbox internals. In applying the method to a VC link, the shared media constitutes the merge, the shared physical link and the split, as illustrated by Figure 8.

in the lockbox, its handshake control locks. The output decoupler at the output of the lockbox decouples the locked handshake from the shared part of the link, allowing the link to service other VCs, even when the individual VC handshakes are locked. This also makes the aggregate link performance independent of the speed of the VC inputs, since the *flit-time*, defined as the time it takes the link to consume one flit, is independent of the handshake of the VC inputs. The flit is transmitted across the shared media, i.e. the link, and accepted into the handshake-latch implemented in the unlockbox. Here, a flip flop toggles the unlock signal when the output handshake cycle completes (falling edge of the output's acknowledge), indicating that the latch is free to receive another flit. The XOR gate in the lockbox in turn generates a pulse upon the unlock signal toggling, resetting the C-element thus unlocking the input handshake.

The lock-unlock method is very simple, leading to high performance and a very low area overhead. In terms of routing, it needs only one extra inter-router wire per VC, and this wire is toggled only once per flit, making the power overhead minimal. There are also some drawbacks. First, the bandwidth

utilization of the link is poor, when only a single VC is actively contending for link access while the remaining VCs are idle. Each active VC faces a long lock-unlock cycle spanning the lockbox, the shared part of the link, the unlockbox, and back through the unlock wire. Since only one flit can be transmitted per lock-unlock cycle, this causes the bandwidth available to a single active VC to be only part of the total bandwidth available on the physical link. Note however that in the case that there are enough VCs to saturate the link (if all are active), this does not affect the hard lower bound bandwidth guarantees of a VC. Hence it is not a drawback when targeting hard service guarantees, i.e. lower bounds on performance. As the flits of several active VCs merge unto the link, the link utilization is increased. The lock-unlock cycle time of a VC is sensitive to the forward latency of the flits. In this regard, clockless circuits are advantageous, since they make possible a very short forward latency per pipeline stage. In our implementation, which features a pipelined merge and a three stage link pipeline (a total of seven pipeline stages), four active channels are enough to saturate the link.

## 4.4 Credit-based VCs

The credit-based VC solution shown in Figure 10 overcomes the limitations of lock-based VCs, by allowing multiple in-flight flits per VC (the italicized control signal names here also refer to signal names in simulation wave diagrams to be presented in Section 4.5). The *creditbox* controls access to the link, by pairing incoming flits with credits from a credit FIFO. Credits are dataless *tokens* which grant access to the link. A credit at the input indicates available buffer space at the receiving end of the link. Figure 11 shows the internals of a credit-uncreditbox pair. Note that the solid arrows symbolize a full handshake channel (see Figure 5). the boxes are made entirely of standard clockless handshake components: a dataless credit FIFO, a join module, a data FIFO, a fork module [37] and an output decoupler (Section 4.2). The depth of the credit FIFO is equal to the depth of the data FIFO, which buffers flits at the receiving end of the link. At reset the credit FIFO is full of credits, while the data FIFO is empty. When there are no more credits, this indicates that the data FIFO may be full, and the receiving end cannot guarantee to accept more flits. The uncreditbox in turn sends credits back through the *credit link* as flits leave through its output. Thus, if the credit/data FIFOs are deep enough a continuous flow of flits can be maintained, even by a single active VC, while still ensuring non-blocking behavior between VCs.

In controlling the flow of credits back across the link, it is important to note that *there will never be sent more credits back than the credit FIFO can accept*. Each flit passing the creditbox consumes one credit, each flit leaving the uncreditbox generates one credit. A credit can only be *generated* by a flit which has *consumed* a credit earlier. Therefore there will always be buffer space available for a credit arriving at a creditbox, and credits will never stall on the credit link.

An advantage of a credit-based VC link is that the number of global wires

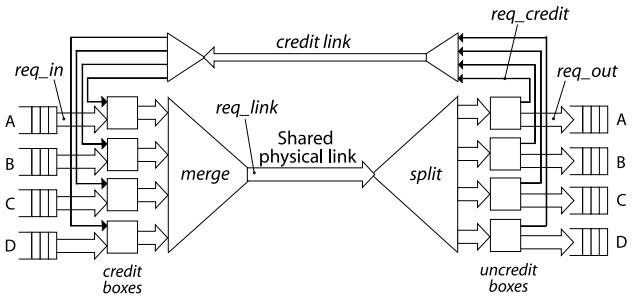



Figure 10: Virtual channel link based on the credit-uncredit method.

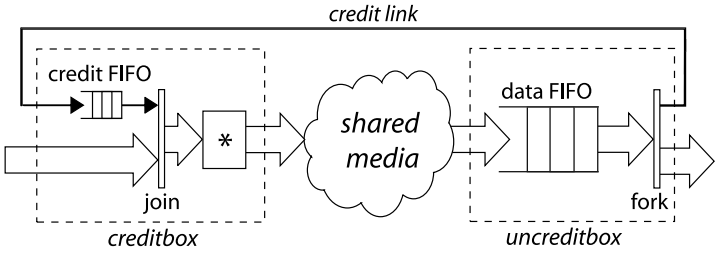



Figure 11: Credit- and uncreditbox internals. As in Figure 9, the shared media is the part of the link between merge and split.

scales well with regard to number of VCs. N being the number of VCs, the credit link will have $\log_2(N)$ wires, indicating to which VC the credit belongs, plus handshake signal wires. The cost of the credit-based solution is in area, mainly used by the FIFOs, and in power consumption. The credit link needs a full handshake per credit. Even if using a 2-phase protocol a minimum of 2 global wire transitions are needed per credit, twice as much as for the lock-based method. Large data/credit FIFOs may also use more power than smaller ones, e.g. if implemented as ripple FIFOs, and care must be taken on this account.

## 4.5 Simulations

Figure 12 shows simulations of two 8-channel VC links, a lock-based and a credit-based. Transactions are indicated by the request ticks. The positions in the system of the signals shown are indicated in Figures 8 and 10. It is seen how VC[0] is given a share of the total BW, when all VCs are actively contending for link access. This share depends on the link access arbitration mechanism implemented by the merge module (see Section 5.1). When the other VCs are idle VC[0] speeds up, limited in the lock-based link by the lock-unlock cycle, and in the credit-based link by the number of credits. In this case there are 4 credits, as seen by the bursts of 4 VC[0] transmissions. For full link utilization,

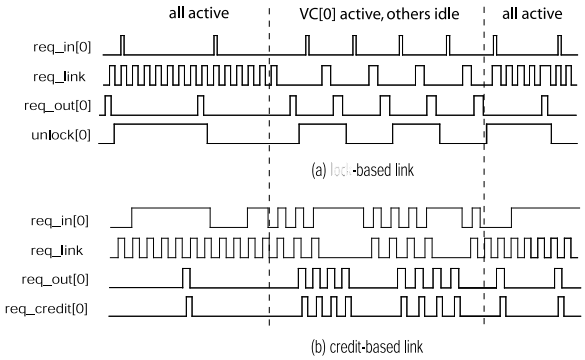



Figure 12: Wave diagram showing simulations of the sharing of a physical link by 8 VCs in (a) lock-based and (b) credit-based links. Signal names are indicated in Figures 8 and 10.

larger credit and data FIFOs are needed.

Clockless circuits, being ruled by local handshaking mechanisms, do not analyze as easily as clocked ones. This can be seen by the difference in width of pulses in the wave diagram. In Figure 12(b) for example, during the phase when only VC[0] is active the second burst of request pulses is seen to be more separated than those in the first burst. This is because the rate of the first burst of flits is dictated only by the link speed, since the credit tokens are already available in the credit FIFO of VC[0]. The rate of the second burst is dictated by the speed at which credits return from the uncreditbox, obviously slightly slower when credits all belonging to the same VC are grouped. A thorough treatment of the dynamics in clockless circuits is outside the scope of this paper.

# 5 Link Design

In this section, we will go into the details of the link access part of the link. This task is handled by the merge module in Figure 3. Link access arbitration quantifies the service guarantee that is provided on a connection. We will also discuss the implementation of the physical link pipeline itself, with regards to issues important for long on-chip interconnects such as power consumption and timing robustness.

## 5.1 Link Access

As explained in Section 3.2, GS connections are provided on virtual circuits. When a connection is in use, it is statically known by the router, to which output VC a given input VC maps. The GS router in MANGO implements no input buffers, routing arriving flits to the appropriate output VC buffers immediately.

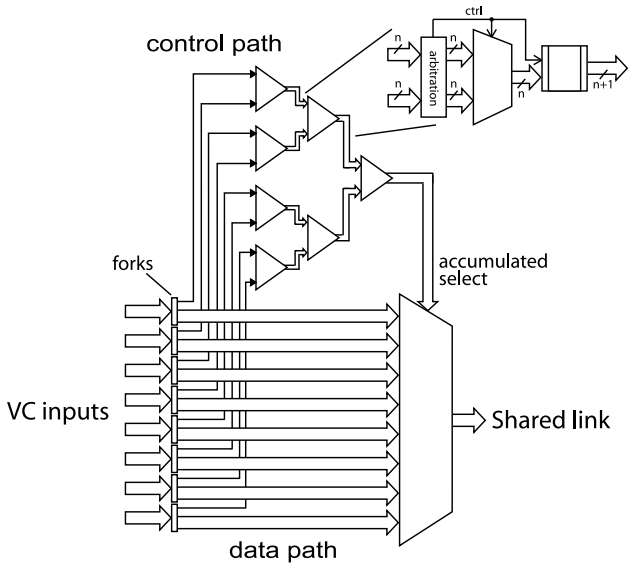



Figure 13: The link access circuit (merge) providing a fair bandwidth share for eight virtual channels.

Placing the VC buffers at the output simplifies the link access arbitration scheme as it is only necessary to arbitrate between VCs on that particular output. Thus simple, fast and low area circuits can be used to implement access schemes. The path across the link, through the GS router, to the target VC buffer is congestion free, by way of the non-blocking switching module to be described in Section 6.3. Hence performance on this path is fully bounded, and link access is the only critical point on a virtual circuit, with regards to congestion. If guarantees can be made for link access, guarantees can be made for the entire virtual circuit. Thus link access arbitration is what facilitates GS connections in MANGO, quantifying the service.

In the router implemented for this work – as a demonstration of the MANGO architecture – we use a simple bandwidth sharing mechanism based on a binary tree structure. Each node in the tree is implemented by a two-input arbitrating merge. The arbiters are fair, in the sense that if an input is being served and the other input is waiting, the waiting input will be served next. This way a given input at a leaf of the tree is guaranteed a portion of the total available link bandwidth. Its is important to note that the flit-time on the link, i.e. the time it takes to complete one flit handshake on the link, is bounded and predictable. This is so for two reasons: (1) because of the VC control mechanisms described in Sections 4.3 and 4.4, flits never stall on the link, and (2) the VC inputs are decoupled from the merge (using the output decouplers), as shown in Figures 9 and 11 (also see Section 4.3). Hence, in e.g. a balanced 8-input binary-tree

merge, each VC is guaranteed a minimum bandwidth of:

$$BW_{VCmin} = \frac{1}{8 \times flit-time}$$

The bandwidth available to a given VC may be higher than indicated above, if not all VCs are actively contending for access. However, the equation expresses the worst-case bandwidth available, i.e. the *hard* bandwidth guarantee. This guarantee holds no matter what the traffic on other VCs is.

When addressing hard lower bounds on performance on individual VCs, it is the aggregate throughput on a link that is important. VC switching inside the router can be slower, without impairing the aggregate link performance. This is the crux of targeting hard guarantees rather than improved average performance.

A pipelined tree-based merge has the drawback of a high area overhead of implementing data buffers at each node in the tree. As shown in Figure 13 we have separated the control and the data path, thus we can omit these area costly data buffers. At each node in the control path tree, two inputs are served fairly and the arbitration decision is appended as an extra bit of output data. This way a select value is accumulated for the 8-input multiplexer. The aggregate throughput is still high, since the control path is pipelined. Now however, a single active channel cannot utilize the full link bandwidth, since each flit will need to wait at its input, for the control decision to complete. This is not to be considered a drawback when employing lock-based VCs, since each VC must in any case complete the transmission of a flit across the link before the lockbox admits the next flit.

In a heterogeneous SoC, different connections may require different measures of bandwidth. To accommodate such heterogeneous service requirements, an asymmetric tree can be instantiated instead of a balanced one. For a given connection, an appropriate VC can then be chosen, which provides the needed bandwidth. On a connection, the end-to-end bandwidth guarantee is dictated by the bottleneck in the path, i.e. the VC with the smallest bandwidth guarantee.

Figure 14(a) shows the path of one VC through the merge. In order to fully utilize the throughput of the pipelined control path, OD1 decouples it from the input. The other output decoupler, OD2, decouples the shared link from the VC input. This is not strictly necessary, but prevents a fast link from being stalled by a slow VC input. Also, it makes the flit-time on the link deterministic and independent of the individual VC inputs.

Figure 14(b) shows an expanded illustration, displaying some of the control circuits with an indication of submodules *fork*, *OD2* and *join*. Join is the part of the output multiplexer which joins the control handshake channel with the data handshake channel. By making some minor timing assumptions, we can achieve major performance improvements, allowing us to utilize much better the throughput potential of a simple pipelined link. Note that the figures display a simplified view of the circuit. The output multiplexer naturally generates a separate acknowledge for each VC input (explained below and in Figure 15).

The multiplexer joins a request from the control path, *req_ctrl*, with a *req_data*. First we assume that *req_in*, indicating that the flit data is valid, is

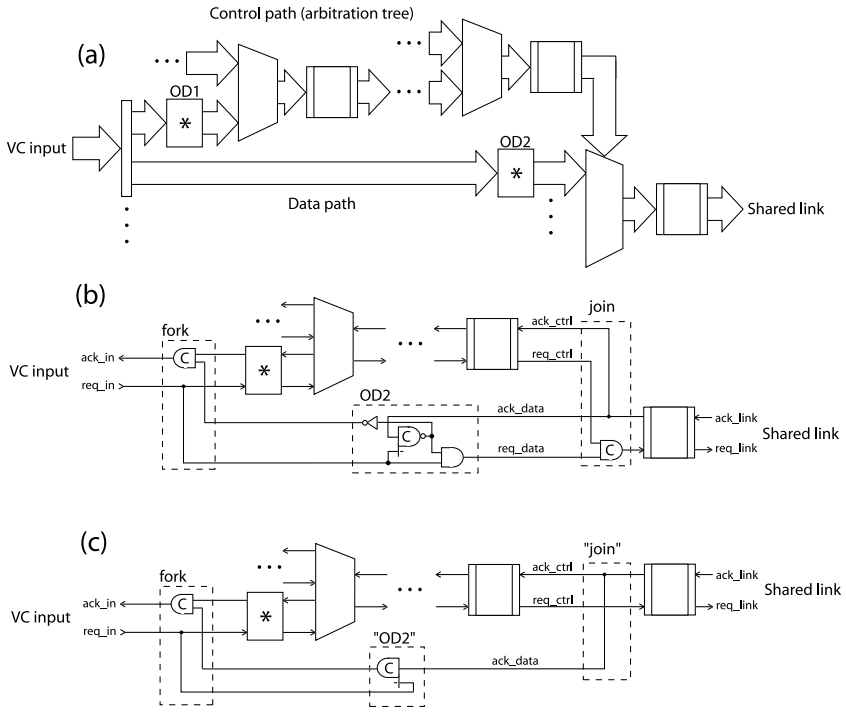



Figure 14: Merge path circuitry: (a) path of one VC through the merge module, (b) control circuits exposed and (c) optimized control circuits with the timing assumption that the C-element in "OD2" is faster than the pulse width of ack_ctrl.

high before *req_ctrl* goes high. This is trivial, since *req_in* leads to *req_ctrl*. The acknowledge from the multiplexer, *ack_data*, which is merely a copy of *ack_ctrl*, is caught by the C-element in OD2. Assuming the C-element is fast enough to capture the *ack_data* pulse, we can ignore the remainder of the return-to-zero phase of the data path handshake. Thus *req_data* can be entirely omitted, reducing the join submodule to simple wires. This improves the cycle time of the output considerably. Figure 14(c) shows the optimized circuit, illustrating how the output cycle is now entirely contained by the link and the control path tree. OD2, now reduced to a single gate, merely snoops *ack_data*, detecting a pulse. The only timing assumption we have made is that the C-element in OD2 is faster than the width of this pulse. This is quite reasonable, since the response loop in the handshake control of the last stage of the tree is more than one C-element.

As indicated above, the output multiplexer generates independent acknowledge signals for each VC input. This is needed for the appropriate OD2 to

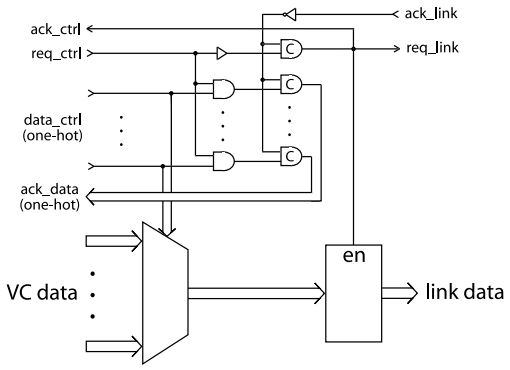



Figure 15: Output multiplexer details.

catch and keep the link acknowledge event, letting the given VC input complete its handshake independently of further handshaking occurring on the shared link. Figure 15, illustrating the output multiplexer and output latch of the merge, shows how we generate *req_link* (and thus also *ack_ctrl*) in parallel with *ack_data*. Here, we are making the timing assumption that the delay of the path from *req_ctrl* to *ack_ctrl* matches that of the path from *req_ctrl* to *ack_data*. The delay matching does not need to be precise, since the data validity at the input of the latch, from the point at which *ack_data* goes high, is extended first by the delay of *ack_data* back through the input fork and OD2, secondly by the delay of the data through the multiplexer. Since only the pulse of *ack_data* is used, the circuit may be made arbitrarily timing safe, by delaying *ack_data*, further extending the data validity. This doesn't compromise the aggregate performance of the link, as the return-to-zero phase of the VC input is not in any critical loop, since it is completely decoupled from the control path.

The decoupling of control- and data-paths and the timing assumptions used to simplify the control circuits help increase the throughput of the merge module significantly. We were able to obtain speeds up to 1 Gflits/s using 0.13$\mu$m CMOS standard cells in the link access circuit described above (back-annotated with typical timing parameters). When making use of delay insensitive (DI) link signaling, the throughput however is limited by the DI decoding (see Section 5.2). If DI signaling is not required, the full performance potential can be exploited.

Similarly to time division multiplexing access schemes, the tree-based link access scheme described herein has an inverse dependency between bandwidth allocation and worst-case link access latency. If low latency is required, a large portion of bandwidth must also be reserved. This makes it unsuitable for implementing low bandwidth connections with stringent latency requirements, e.g. interrupts. But since the MANGO router architecture allows a modular replacement of the arbitration scheme, the binary tree suffices as an example in showing the concept of implementing GS connections on virtual circuits. Please refer to

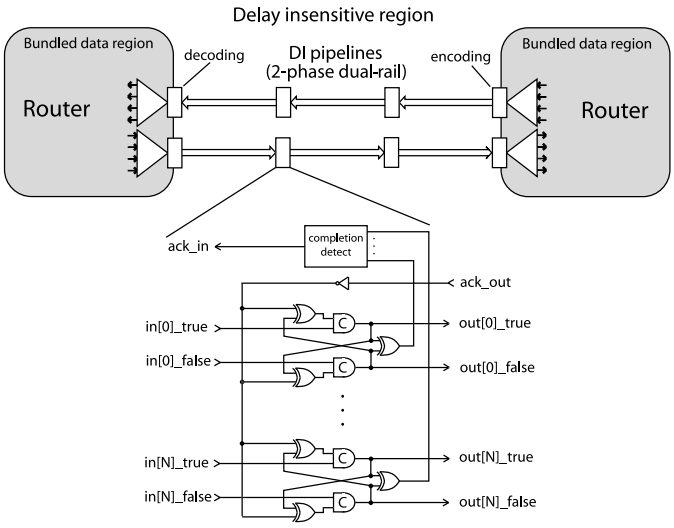



Figure 16: Delay insensitive links.

our work in [23] for an access scheme which is more elaborate and flexible with regards to latency guarantees, yet still area efficient.

## 5.2 Inter-Router Pipelines

As projected in [1] and [2], global SoC communication will incur an increasing part of the total power budget in future microchip technologies. Also the delay uncertainty of globally propagated signals force circuit designers to allocate progressively safe timing margins for these, at the expense of performance. For these reasons we have implemented our 33-bit pipelined inter-router links using a dual-rail encoded delay insensitive (DI) 2-phase asynchronous signaling protocol.

Figure 16 illustrates how the routers themselves are implemented using area efficient bundled data handshaking. A bundled data handshake channel is characterized by a data path and separate handshake signals (request and acknowledge) in parallel with this. In implementing such handshaking, timing matching is required, with regards to the propagation delay of the data signals and the request signal (which indicates data validity). Within the relatively limited physical extent of a router, wire delays can be considered small compared with gate delays, and timing issues are thus more easily handled. Wire forks can mostly be considered isochronic. On the long inter-router links on the other hand, DI signaling is employed, in order to enhance timing robustness. The figure also shows how the DI pipeline stages are implemented. In a DI handshake channel data validity is encoded into the data itself. Hence DI signaling eliminates timing problems caused by delay variations in data and handshake control signal propagation. This is a highly desirable property on long on-chip

links, and helps in obtaining a modularized system design flow. Timing integrity between routers is inherently ensured. See [37] for more details on asynchronous handshaking protocols.

In the current version of MANGO, we implement 2-phase dual-rail signaling on the pipeline. For each flit transaction we thus need 1 data wire toggle for each bit communicated, plus 1 toggle of the acknowledge wire. As the power cost of toggling global wires increases in future technologies, relative to that of processing logic, it will prove worthwhile to implement 1-of-4 encoding. This would reduce the amount of toggling on global wires to 1 toggle for each 2 bits plus 1 toggle for the acknowledge. This encoding was also tried, but was discarded because it impaired performance too much.

The decoding of DI signals at the router inputs has turned out to be the performance bottleneck. Although our link access circuits operate at up to 1 Gflits/s, our DI-decoders limit the per port performance to 650 Mflits/s (back annotated, typical timing parameters). In our simulations, the pipeline segments are loaded with 0.12 pF corresponding to 0.7 mm in the used 0.13 $\mu$m technology. Hence each link is just over 2 mm long. The forward latency in the pipeline is merely 300 ps per pipeline stage, under worst-case timing conditions. This clearly illustrates an advantage of a clockless implementation.

One way to increase the throughput of the DI decoders would be to partition the links into bundles, each with an independent acknowledge. This way, the decoding can be performed in smaller steps. Alternatively the DI encoders/decoders may also be omitted, in order to obtain the full operational speed. This would then require careful delay matching of handshake control and data signals on the links. Such matching is becoming increasingly difficult in scaling technologies. Process variations are worsening, and signal integrity is harder to ensure on long wires, as wire pitches decrease and signal transition times (of neighboring aggressor wires) become faster.

# 6 Router Design

In the following we will detail the implementation of the router. The router consists of two separate routers; the BE router and the GS router. The BE router dynamically routes packets based on the information in their header. The GS router switches data streams on GS connections. Because of the modular architecture, a MANGO router can easily be instantiated to fit specific needs of the system in which it is used.

## 6.1 The MANGO Router

Figure 17 shows a conceptual picture of the MANGO router, illustrating how the VC links are integrated into the router architecture. The router implements a number of uni-directional input and output ports, and a local port. The input/output ports are *network* ports which, via point-to-point VC links, connect the router to neighboring routers. The local port consists of a number of

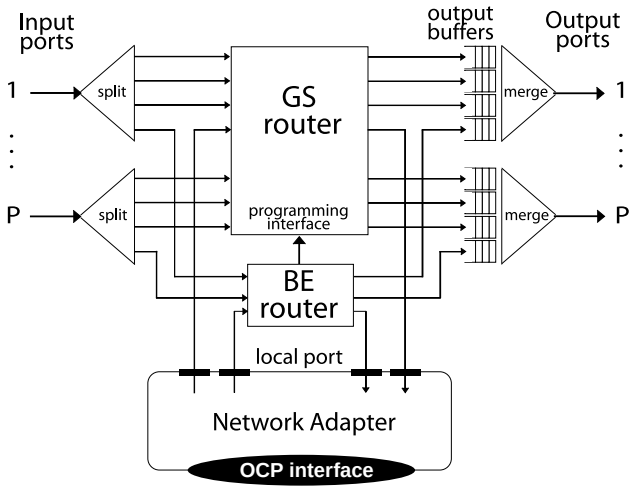



Figure 17: The MANGO router conceptually.

unidirectional physical interfaces, which connect to the local NA. The network ports and the local port implement the same type of interface with regards to bit width and flow control. The only difference between the two is that the local port implements one physical interface per channel while a network port implements VCs, merging all logical channels unto a single link. Each interface on the local port provides access to, or from, a network service. This service could be BE routing, or a GS connection.

Internally, the router consists of a BE router, a GS router, output buffers, splits and arbitrating merges (link arbiters). The BE and the GS router are implemented as separate submodules. A conceptually similar approach is seen in the ÆTHEREAL router [40]. In MANGO, this is done in order to make for a simple implementation and a modular instantiation. A subset of the VCs are allocated for BE routing (in Figure 17, a single VC is allocated for BE). The BE router dynamically source-routes connection-less data packets, according to the routing path specified in the packet header (see Section 6.2). The GS router uses the remaining VCs to implement virtual circuits through the network. The GS router provides non-blocking switching between the input ports and the output buffers, therefore *GS can be quantified purely on the basis of link access arbitration.* This is the key concept of enabling end-to-end service guarantees in MANGO. It is explained further in Section 6.3.

GS connections are established by programming input-to-output VC mappings into the GS router. This is done by routing BE packets to the GS programming interface via the BE router. The GS router simply allows its internal mapping table to be written to. The BE header flit, which contains routing information, is discarded, the second flit contains the mapping table address and the third flit contains the data to be written. Connection allocation and

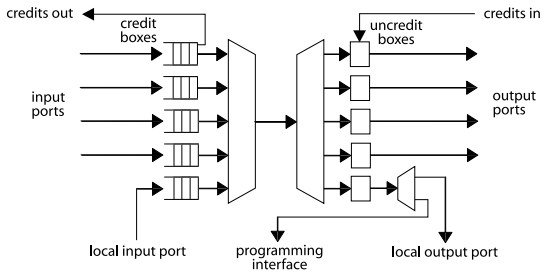



Figure 18: The BE router.

handling tasks need to be managed centrally, by a system programming unit (SPU). This central programming model is adapted to keep router complexity low. Since GS connections are set up using BE routing, the time it takes to program these into the network is unknown. We envision a run time scenario as a temporally separated set of relatively long-lived setups, each separated by a network programming phase. During the programming phase, only the SPU will access the network. Thus no congestion will occur, and the time to program the GS connections is bounded and predictable. At the end of the programming phase, the SPU releases its lock on the system, allowing the connections to be used. Detailing the programming model further is outside the scope of this paper.

## 6.2 The BE Router

The BE router employs credit-based VCs. This allows multiple in-flight BE flits on the links. Investigating means of GS routing is the focus of this work, and the BE router is mostly included for sake of completion, as a way to program GS connections into the network. Several other papers describe BE router implementations for NoCs [30, 15, 16, 31, 32]. The modular architecture of the MANGO router makes it easy to plug in a different, more optimal BE router.

The simple, area efficient BE router, shown in Figure 18, implements a source routing scheme. The first flit of a packet is the header flit. The three MSBs of the header indicate one of five output ports. After passing the router, the header is rotated three bits, positioning the header bits for the next hop. The router arbitrates fairly between input ports contending for access. The packets are variable length; a control bit is used to indicate the last flit (end-of-packet bit). Once an input port has gained access, the routing decision will *stick* until the last flit has been detected, thus keeping packet coherency. With a flit size of 33 bits (of which one is the end-of-packet bit) it is thus possible to make 10 routing hops. As systems scale, it will also prove viable to increase the flit size, allowing for more hops. Alternatively a BE router which allows multi-flit headers could be designed. The interface used to program GS connections into the GS router, is implemented as an extension on the local port. Deadlocks can

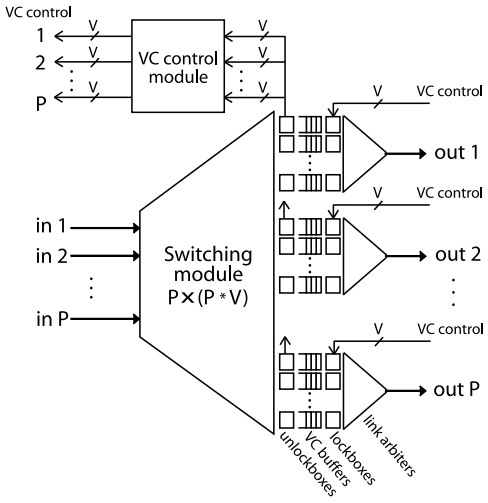



Figure 19: The GS router implements non-blocking input to output switching.

be avoided by scheduling the paths of BE packets with an appropriate routing strategy, according to the chosen network topology.

## 6.3 The GS Router

Figure 19 illustrates the GS router architecture. Here $P$ is the number of ports on each routing node and $V$ is the number of VCs on each port. It is seen how the VC split modules (see Figure 3) are integrated into the switch itself, resulting in its area being linearly scalable with the number of VCs. It is also seen how $V$ VC control signals (unlock wires, as per Section 4.3) enter the router on each output port, and exit the router on each input port.

In the previous router the flits are extended, by appending a number of *steering bits* to the 33 data bits. These bits guide the flits through the pipelined *switching module* to the VC buffer that has been reserved for the given connection. The reason for appending these steering bits in the previous router is that this allows for the flits to be routed to their VC buffer directly. Alternatively, the ID of the VC from which the flit originated would need to be appended (uniquely identifying the source of the flit) and a lookup based on this would be performed. This however results in a power overhead and performance degradation. In the $5 \times 5$ demonstration router presented herein, there are 4 possible destination ports for an incoming flit. Hence, irrespective of the number of VCs, appending the destination VC ID instead of the source VC ID merely results in a 2-bit overhead in the data path.

The switching module is non-blocking, meaning that the latency of a flit across the link, through the router, to its designated VC buffer, is predictably

bounded. Accessing the link is the only uncertainty involved in transmitting data from a VC buffer to a VC buffer in the next router. In order to provide hard GS - of any type - there is only call for appropriate link access arbitration. The link arbiter (Section 5.1) is thus the key element in providing GS on virtual circuits. It arbitrates between the VCs contending for access to the link, determining the type and quantity of GS that is provided.

It is thus seen how the GS quantification is decoupled from the switching functionality in the router. This makes it an easy and modular task to instantiate new types of GS. Accordingly, GS schemes such as the one presented in this paper, the one presented in [23], or other schemes, can easily be applied directly to the link and router architectures presented herein.

The GS router employs area efficient lock-based VC control, its aim being to deliver hard lower bounds on performance (see Section 4.3). According to Figure 9, the link as well as the switching module constitute the shared media around which the VC control is wrapped. The VC control module in Figure 19 establishes a VC control channel from an unlockbox back to a lockbox in the previous router; a step back on a given connection. Thus a given VC will only transmit a flit across the shared media, if the buffer in the forward path of the virtual circuit of which it is part, has free buffer space.

The lock-based VC control scheme uses a single wire per VC. Establishing a VC control channel is simply a matter of multiplexing an unlock output signal wire from an unlockbox onto the unlock input signal wire of the appropriate lockbox in a neighboring router, one step back on the connection. As explained in Section 4.3, the lock-unlock cycle determines the highest throughput on a VC. The full link bandwidth however is exploited by interleaving flits from different VCs, i.e. the overlapping of lock-unlock cycles of these. A single VC cannot utilize the full link bandwidth, but with an appropriate link access scheme, our goal to provide hard lower bounds on performance – performance guarantees – can be met, while benefitting from the low area overhead of lock-based VC control. Note that a GS connection needs only a single-flit buffer in each router. Flow control is maintained on the virtual circuit, from buffer to buffer, as a distributed FIFO through the network. The BE router described in Section 6.2 on the other hand targets an improved average performance. Here it is clearly an advantage to use credit-based VC control instead.

For each virtual circuit, the router stores the steering bits needed to guide flits to the VC buffer reserved for the circuit in the *next* router, as well as control bits used to establish a VC control channel back to the lockbox in the *previous* router. As seen, the setup information for each hop on a connection is thus stored in two places: one for the flit forward path, and one for the VC control reverse path. This overhead is accepted because it facilitates some very simple circuits. In any case, it constitutes a small fraction of the total router area. For further details concerning the GS router, please refer to [7].

Table 1: Area usage in the MANGO router.

| Module | Area | |
|---|---|---|
| Connection table | 0.005 $mm^2$ | 1.8% |
| Switching module | 0.096 $mm^2$ | 34.7% |
| VC buffers | 0.054 $mm^2$ | 19.5% |
| GS link access | 0.025 $mm^2$ | 9.0% |
| VC control | 0.018 $mm^2$ | 6.5% |
| BE router | 0.028 $mm^2$ | 10.1% |
| DI encode/decode | 0.051 $mm^2$ | 18.4% |
| **Total** | **0.277 $mm^2$** | **100%** |

# 7 Implementation

As a demonstration of the MANGO architecture we have implemented a $5 \times 5$ 33-bit MANGO router using 0.13 $\mu$m CMOS standard cells from STMicroelectronics. The router supports 7 independently buffered GS connections on each of the four network ports in addition to connection-less BE source-routing, with 4-flit deep BE buffers on each input port. The local port implements 4 GS ports and 1 BE port. When routing data using the BE router, one bit is reserved to indicate end-of-packet. The network ports implement 2-phase dual-rail DI encoding/decoding. The performance in netlist simulations using worst-case timing conditions (125 C/1.08 V/slow process corner) was 420 Mflits/s per port (650 Mflits/s under typical timing conditions). The performance was limited by the DI decoding stage. Without DI signaling, the per port performance was 646 Mflits/s (1 Gflits/s under typical timing conditions).

The pre-layout area was 0.277 $mm^2$. The area usage, detailed in Table 1, may seem a bit high. One must keep in mind however, that this demonstration router is to be considered a deluxe version, targeted for coarse-grained SoC with complex communication requirements, i.e. multiple connections per core and a need for per connection GS. For an average core size of 5 $mm^2$, and one router per core, a NoC with such routers would constitute approximately 5.5% of the total chip area. For a SoC with less complex routing needs, a router with a smaller number of VCs can easily be instantiated, with the same high per port speed but with a reduced area.

The switching module and the VC buffers together account for more than half of the total area. Much area could be saved by using custom-designed buffers. As a quick comparison, a 0.13 $\mu$m instantiation of an ÆTHEREAL router, which also provides per connection bandwidth guarantees, had a data path speed of 500 MHz (worst case timing parameters) and a laid out area of 0.175 $mm^2$ [17], using custom hardware FIFOs. The router supports any number of connections, as the data path can be time sliced at an arbitrary granularity. The connections are however not independently buffered – a key feature of the MANGO router. Thus end-to-end flow control is needed, e.g. using credits. This makes for more complex and area consuming network adapters. In the

Table 2: Latency on GS connections (worst-case timing).

| Module | Latency |
|---|---|
| VC buffers + share/unshareboxes | 1.2 ns |
| GS link access | 2.3 ns |
| DI encoding | 0.7 ns |
| Link pipeline | 0.9 ns |
| DI decoding | 1.5 ns |
| GS switching module | 1.6 ns |
| **Total** | **8.2 ns** |

MANGO architecture end-to-end flow control is inherent. Moreover, the type of GS in ÆTHEREAL is an integral part of the router design (bandwidth sharing by time division multiplexing). The MANGO router adapts a more modular GS approach, supporting a range of potential GS types with the same router architecture.

Pipelining of the switching module is necessary in order to keep performance up. However this also takes a considerable part of the total router area. Since the VCs used for GS connections employ lock-based VC control, only a single flit can be transmitted on each VC at one time. The lock-unlock cycle time, which constitutes the forward latency of the pipelined link and the GS router plus the reverse latency of the unlock signal, is 9.0 ns. The forward latency is detailed in Table 2. The lock-unlock cycle determines the maximum throughput of a single VC, which accordingly is 111 Mflits/s. In this respect, clockless circuits are at a major advantage over synchronous ones, in that the forward latency is much less than the number of pipeline stages times the cycle time of one stage. Note that too much pipelining degrades the best-case performance of a connection, increasing the length of the lock-unlock cycle time. A balance is needed, on one hand to maximize the aggregate throughput of the router, and on the other hand to minimize the amount of pipelining in order to keep the area down and the per VC best-case performance up.

# 8 Conclusion

During recent years, research into on-chip communication structures has shown NoC-based SoC design to be a promising concept for SoC communication. The greatest advantage of NoCs is that they facilitate modularity in the SoC design flow. By the use of standard sockets and guaranteed communication services, building and verifying a SoC could well be a matter of piecing together off-the-shelf IP cores.

In this paper we have detailed the implementation of guaranteed services in MANGO (*Message-passing Asynchronous Network-on-Chip providing Guaranteed services over OCP interfaces*). Three key features of MANGO which help enable a modular SoC design flow are (i) clockless implementation, (ii) stan-

dard OCP socket access points and (iii) guaranteed communication services. In addition to this, the router architecture is highly modular itself. This will assist in automatic generation of application specific networks. As a demonstration, a $5 \times 5$ router has been designed using commercially available 0.13 $\mu$m CMOS standard cells. The design shows that it is feasible to implement the concepts introduced, and the use of clockless circuit techniques proves to be a viable solution, to the challenge of implementing global communication infrastructures in large SoCs.

## References

[1] D. Sylvester and K. Keutzer, "A global wiring paradigm for deep submicron design," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 19, no. 2, pp. 242–252, 2000.

[2] R. Ho, K. W. Mai, and M. A. Horowitz, "The future of wires," *Proceedings of the IEEE*, vol. 89, no. 4, pp. 490–504, April 2001.

[3] Semiconductor Industry Associations. International technology roadmap for semiconductors (ITRS) 2003. [Online]. Available: http://public.itrs.net/Files/2003ITRS/Home2003.htm

[4] L. Benini and G. D. Micheli, "Networks on chips: A new SoC paradigm," *IEEE Computer*, vol. 35, no. 1, pp. 70–78, January 2002.

[5] A. Jantsch and H. Tenhunen, *Networks on Chip*. Kluwer Academic Publishers, 2003.

[6] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Proceedings of the 38th Design Automation Conference (DAC 2001)*, June 2001, pp. 684–689.

[7] T. Bjerregaard and J. Sparsø, "A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip," in *Proceedings of Design, Automation and Testing in Europe Conference (DATE 2005)*. IEEE Computer Society, 2005, pp. 1226–1231.

[8] T. Bjerregaard, S. Mahadevan, R. G. Olsen, and J. Sparsø, "An OCP compliant network adapter for GALS-based SoC design using the MANGO network-on-chip." in *Proceedings of International Symposium on System-on-Chip (SOC 2005)*. IEEE, 2005, (To appear).

[9] T. Bjerregaard and J. Sparsø, "Virtual channel designs for guaranteeing bandwidth in asynchronous network-on-chip," in *Proceedings of the IEEE Norchip Conference (NORCHIP 2004)*. IEEE, 2004, pp. 269–272.

[10] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks - an Engineering Approach*. Morgan Kaufmann, 2003, ch. 9, pp. 475–558.

[11] W. J. Dally and C. L. Seitz, "The torus routing chip," *Distributed Computing*, vol. 1, no. 4, pp. 187–196, 1986.

[12] C. L. Seitz and W.-K. Su, "A family of routing and communication chips based on the mosaic," in *Proc. of 1993 Symposium on Research on Integrated Systems*. MIT Press, Jan. 1993, pp. 320–337.

[13] W. J. Dally, "Virtual-channel flow control," *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 2, pp. 194–205, March 1992.

[14] R. J. Cole, B. M. Maggs, and R. K. Sitaraman, "On the benefit of supporting virtual channels in wormhole routers," *Journal of Computer and System Sciences*, vol. 62, no. 1, pp. 152–177, 2001.

[15] L.-S. Peh and W. J. Dally, "A delay model for router microarchitectures," *IEEE Micro*, vol. 21, no. 1, pp. 26 –34, 2001.

[16] R. Mullins, A. West, and S. Moore, "Low-latency virtual-channel routers for on-chip networks," in *Proceedings of the International Symposium on Computer Architecture (ISCA 2004)*. IEEE Computer Society, 2004, pp. 188–197.

[17] J. Dielissen, A. Rădulescu, K. Goossens, and E. Rijpkema, "Concepts and implementation of the Philips network-on-chip," in *Proceedings of the International Workshop on IP-Based SOC Design (IPSOC 2003)*, Nov. 2003.

[18] M. D. Osso, G. Biccari, L. Giovannini, D. Bertozzi, and L. Benini, "Xpipes: A latency insensitive parameterized network-on-chip architecture for multiprocessor SoCs," in *Proceedings of the 21st International Conference on Computer Design (ICCD 2003)*. IEEE Computer Society, 2003, pp. 536–539.

[19] OCP International Partnership. (2003) Open Core Protocol Specification, Release 2.0. [Online]. Available: http://www.ocpip.org

[20] ARM. (2004, March) AMBA AXI Protocol Specification, version 1.0. [Online]. Available: http://www.arm.com/products/solutions/axi_spec.html

[21] K. Goossens, J. Dielissen, O. P. Gangwal, S. G. Pestana, A. Radulescu, and E. Rijpkema, "A design flow for application-specific networks on chip with guaranteed performance to accelerate SoC design and verification," in *Proceedings of the Design, Automation and Test in Europe Conference (DATE 2005)*. IEEE, 2005, pp. 1182–1187.

[22] K. Goossens, J. Dielissen, and A. Radulescu, "Æthereal network on chip: Concepts, architectures and implementations," *IEEE Design & Test of Computers*, vol. 22, no. 5, pp. 414–421, 2005.

[23] T. Bjerregaard and J. Sparsø, "A scheduling discipline for latency and bandwidth guarantees in asynchronous network-on-chip," in *Proceedings of the 11th IEEE International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC 2005)*. IEEE, 2005, pp. 34–43.

[24] D. Chapiro, "Globally-asynchronous locally-synchronous systems," Ph.D. dissertation, Stanford University, 1984, Report No. STAN-CS-84-1026.

[25] J. Muttersbach, T. Villiger, K. Kaeslin, N. Felber, and W. Fichtner, "Globally-Asynchronous Locally-Synchronous Architectures to Simplify the Design of On-Chip Systems," in *Proc. 12th International ASIC/SOC Conference*. IEEE, Sept. 1999, pp. 317–321.

[26] Arteris - the network-on-chip company. [Online]. Available: http://www.arteris.com

[27] A. Radulescu, J. Dielissen, K. Goossens, E. Rijpkema, and P. Wielage, "An efficient on-chip network interface offering guaranteed services, shared-memory abstraction, and flexible network configuration," in *Proceedings of the 2004 Design, Automation and Test in Europe Conference (DATE 2004)*. IEEE, 2004, pp. 4–17.

[28] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch, "Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip," in *Proceedings of the Design, Automation and Testing in Europe Conference (DATE 2004)*. IEEE, 2004, pp. 890–895.

[29] J. Liang, A. Laffely, S. Srinivasan, and R. Tessier, "An architecture and compiler for scalable on-chip communication," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 7, pp. 711–726, 2004.

[30] J. Bainbridge and S. Furber, "Chain: A delay-insensitive chip area interconnect," *IEEE Micro*, vol. 22, no. 5, pp. 16–23, October 2002.

[31] D. Rostislav, V. Vishnyakov, E. Friedman, and R. Ginosar, "An asynchronous router for multiple service levels networks on chip," in *Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC 2005)*. IEEE, 2005, pp. 44–53.

[32] E. Beigne, F. Clermidy, P. Vivet, A. Clouard, and M. Renaudin, "An asynchronous NOC architecture providing low latency service and its multi-level design framework," in *Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems, 2005 (ASYNC 2005)*. IEEE, 2005, pp. 54–63.

[33] S. F. Nielsen and J. Sparsø, "Analysis of low-power SoC interconnection networks," in *Proceedings of the 19th Norchip Conference (NORCHIP 2001)*, 2001, pp. 77–86.

[34] H. van Gageldonk, D. Baumann, K. van Berkel, D. Gloor, A. Peeters, and G. Stegmann, "An asynchronous low-power 80C51 microcontroller," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC 1998)*, 1998, pp. 96–107.

[35] S. B. Furber, J. D. Garside, P. Riocreux, S. Temple, P. Day, J. Liu, and N. C. Paver, "AMULET2e: An asynchronous embedded controller," *Proceedings of the IEEE*, vol. 87, no. 2, pp. 243–256, Feb. 1999.

[36] A. Jantsch and R. L. A. Vitkowski, "Power analysis of link level and end-to-end data protection in networks on chip," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS 2005)*. IEEE, 2005, pp. 1770–1773.

[37] J. Sparsø and S. Furber, *Principles of Asynchronous Circuit Design - a Systems Perspective*. Kluwer Academic Publishers, Boston, 2001.

[38] Petrify, a tool for synthesis of Petri nets and asynchronous controllers. [Online]. Available: http://www.lsi.upc.edu/petrify/

[39] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev, "Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers," *IEICE Transactions on Information and Systems*, pp. 315–325, 1997.

[40] E. Rijpkema, K. G. W. Goossens, A. Radulescu, J. Dielissen, J. V. Meerbergen, P. Wielage, and E. Waterlander, "Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip," in *Proceedings of the Design, Automation and Test in Europe Conference (DATE 2003)*. IEEE, 2003, pp. 350–355.

Paper D

# A Scheduling Discipline for Latency and Bandwidth Guarantees in Asynchronous Network-on-Chip

Tobias Bjerregaard and Jens Sparsø

Published in *Proceedings of the 11th IEEE International Symposium on Advanced Research in Asynchronous Circuits and Systems*, New York, IEEE 2005.

# A Scheduling Discipline for Latency and Bandwidth Guarantees in Asynchronous Network-on-Chip

Tobias Bjerregaard and Jens Sparsø
Technical University of Denmark (DTU)
Informatics and Mathematical Modelling
2800 Lyngby, Denmark
Email: {tob, jsp}@imm.dtu.dk

## Abstract

*Guaranteed services (GS) are important in that they provide predictability in the complex dynamics of shared communication structures. This paper discusses the implementation of GS in asynchronous Network-on-Chip. We present a novel scheduling discipline called Asynchronous Latency Guarantee (ALG) scheduling, which provides latency and bandwidth guarantees in accessing a shared media, e.g. a physical link shared between a number of virtual channels. ALG overcomes the drawbacks of existing scheduling disciplines, in particular the coupling between latency and bandwidth guarantees. A 0.12 μm CMOS standard cell implementation of an ALG link has been simulated. The operation speed of the design was 702 MDI/s.*

## 1. Introduction

Physical issues of deep submicron technologies, as well as design complexity issues of large scale chip designs, make current poorly scalable solutions for global on-chip communication such as busses, unsuited for future system-on-chip (SoC). There is a general consensus that the communication requirements, as well as the design flow, of billion transistor SoC are best accommodated by shared, segmented interconnection networks [3][13][8]. Recent years have seen the development of such dedicated SoC communication structures, known as Network-on-Chip (NoC). NoC facilitates a truly modular and scalable design approach, allowing the easy integration of a variety of cores in a SoC.

Chip-wide synchrony is becoming prohibitively difficult to achieve in large chips [1]. Possible solutions lie in the concept of global asynchronous locally synchronous systems (GALS) [7][16]. With GALS, the use of fully asynchronous circuits for implementing NoC seem an obvious possibility. The problem of distributing a global clock can be entirely avoided, and the integration of cores with different timing specifications becomes an integral part of the design flow. Only the network has a global span, and benefits from advantages of asynchronous circuits such as distributed control and zero dynamic idle power.

Traditionally multicomputer networks support best-effort (BE) routing, for which no performance guarantees are given. Much NoC research builds on this by improving BE routing efficiency under the constraints of single-chip system design [17][15]. In [18] the authors argue for the necessity of a combination of BE routing as well as guaranteed service (GS) routing in NoC. Basically BE improves the average resource utilization while GS incur predictability, a quality which is often desirable, in particular in real-time systems.

Previously published NoCs which provide GS are ÆTHEREAL [18][9] and NOSTRUM [14]. Both are synchronous and employ variants of time division multiplexing (TDM) for providing per connection bandwidth (BW) guarantees. TDM has the drawback of the connection latency being inversely proportional to the BW, thus connections with low BW *and* low latency requirements, e.g. interrupts, are not supported. Also, TDM is not possible in asynchronous systems, which have no explicit notion of time. In [12], an asynchronous NoC providing differentiated services by prioritizing VCs was presented. Though this approach delivers improved latency on prioritized connections, no *hard* guarantees are given.

This work presents a novel scheduling discipline called *Asynchronous Latency Guarantee* (ALG) scheduling, that provides hard per connection latency and BW guarantees. The guarantees are not inversely dependent on each other, thus ALG overcomes the limitations of BW allocation schemes based on TDM, and supports a wide range of traffic types characterized by different GS requirements. At opposite ends of the GS spectrum; ALG supports both latency critical, low BW traffic such as interrupts, but also streaming data, which does not have strin-

Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'05)
1522-8681/05 $20.00 © 2005 IEEE

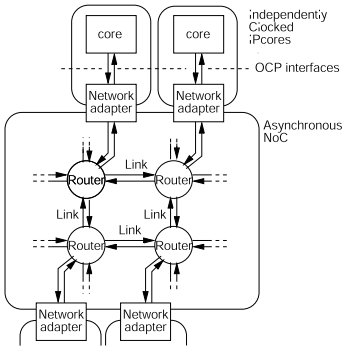



**Figure 1. An asynchronous network connecting independently clocked cores facilitates modularity in large scale SoC designs.**

gent latency requirements but requires GS in terms of BW. In addition, ALG operates in a completely asynchronous environment. We demonstrate with a low area 0.12 $\mu$m CMOS standard cell implementation.

The rest of the paper is organized as follows. In Section 2 we introduce our asynchronous NoC MANGO (*Message-passing Asynchronous Network-on-chip providing Guaranteed services over OCP interfaces*) for which ALG will constitute the scheduling discipline. Section 3 looks at the background of GS, addressing current solutions, and defining the requirements for an optimal solution in NoC. Section 4 explains the concepts of ALG scheduling and provides proof of its functionality, and Section 5 extends the proof given in Section 4 to account for buffer limitations as well. In Section 6 we describe our implementation of an on-chip ALG link, and in Section 7 we present some simulation results. Finally Section 8 provides a conclusion.

## 2. MANGO Overview

This section provides a brief overview of MANGO. As shown in Figure 1, a MANGO NoC consists of network adapters (NA), routers and links. Each core is connected to the network through an NA, which provides high level communication services, in the form of OCP (Open Core Protocol) transactions [2], on the basis of primitive packet routing services implemented by the network. Each NA is connected to a router, and also performs the synchronization between the clocked core and the asynchronous network. The routers are connected by links in a grid-type structure, either homogeneous or heterogeneous.

A link in MANGO, as illustrated in Figure 2, implements a number of independently buffered, logically separated virtual channels (VCs). These VCs contend for access to the shared physical link. VC control ensures that no flit (flow control unit) stalls while making use of the shared media, causing flits from other VCs to be blocked. The implementation of VCs in asynchronous systems was the topic of our work in [4] and will be further discussed in Section 6.

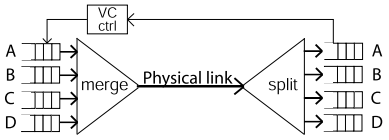



**Figure 2. Basic link in which virtual channels A to D share a physical link.**

In [5] details of the MANGO router architecture were presented. The MANGO router provides connectionless BE routing as well as connection-oriented GS routing. A GS connection is a logical point-to-point circuit between two different NAs in the network. Such a *virtual circuit* is established by reserving a sequence of VCs. A connection is thus in effect a logical FIFO, distributed across the network. This is a key feature of MANGO, and helps simplify the use of connections, in particular with regard to end-to-end flow control. Connections are established using BE packets.

To enable modularity in instantiating the MANGO router, BE and GS routing are implemented by two separate modules. The GS router is implemented by a switching fabric which provides non-blocking switching between the input ports and the output buffers; any flit arriving at any input port will be routed to the appropriate VC output buffer, immediately and without congestion. Since the router is non-blocking, the only point at which congestion between different connections can occur is during link access. Therefore *GS can be realized purely on the basis of link access arbitration*. This is a key concept of MANGO. The fact that MANGO implements output buffers furthermore makes the link access arbitration circuits simple, and facilitates a modular approach to the implementation of GS; new GS schemes can be instantiated simply by plugging a new link arbiter module into the router. Please refer to [5] for further details.

## 3. Guaranteed Services

In the following we first discuss network performance parameters and establish a taxonomy. We then argue for the need for connection-oriented routing and discuss GS schemes used in current NoC and in macro networks, and finally we propose a set of requirements for GS in NoC.

### 3.1. Performance Parameters

Service guarantees are quantified in terms of one or more performance parameters. Aspects of network performance analysis are discussed in detail in [10], the basic parameters being BW and latency. In order to appropriately specify service bounds, both BW and latency must be indicated. A latency guarantee is useless, if the throughput which can be sustained is too small, likewise a BW guarantee is useless without bounds on the latency incurred.

While the BW bound of a stream of flits is determined by the bottleneck in its path, the total latency of a flit in a network is characterized by the sum of latencies encountered. These include the network admission latency $t_{admit}$, during which the required network connection is accessed, and a number of hop latencies, a hop being the flit movement from the buffer in one routing node, across a link and into the buffer in the neighboring routing node. The hop latency consists of an access latency $t_{access}$, the time it takes for the flit to be granted access to the shared routing resources, e.g. the link, plus a transmission latency $t_{link}$, the time it takes to transmit the flit to the buffer in the next routing node, once access has been granted. The total latency, of a flit traversing a path which is $X$ hops long, is thus $t_{total} = t_{admit} + t_{access1} + t_{link1} + \ldots + t_{accessX} + t_{linkX}$.

### 3.2. Connection-Oriented GS

In order to provide hard service guarantees, connection-oriented routing is absolutely essential. In connection-less routing, all data travels on the same logical network, and any transmission can potentially stall another. GS traffic must be logically independent of other traffic in the network. In MANGO (see Section 2) we use VCs to establish connections on logically independent virtual circuits.

Hard bounds on service guarantees are beneficial from a system-level point of view in that they promote a modular design flow. Without such guarantees, a change in the system may require extensive top-level re-verification. Thus GS in NoC holds the potential to reduce turn-around-time of large SoC designs. Also, while formal verification of the performance of BE routing networks is often not possible – as desirable in critical real-time systems – GS makes it so.

### 3.3. GS Schemes

An overview of scheduling disciplines for GS in packet-switched networks is given in [19]. The basic solution to providing BW guarantees is based on *fair fluid queuing* (FFQ). FFQ is a general form of *head-of-line processor sharing* (HOL-PS), which implements separate queues for each connection. The heads of the queues are serviced in a manner such as to provide fair-share access to the shared media, e.g. a link.

In an asynchronous NoC, FFQ-type access schemes have the unpleasant drawback of a very high worst case latency. A tree-arbiter can approximate FFQ, however since nothing can be said concerning the timing of the inputs with respect to each other, any packet arriving on a given channel, potentially has to wait for all other inputs before being serviced. Thus the worst case latency accumulated in an asynchronous NoC implementing this link access scheme is very high. Also, the access time is inversely proportional to the BW reservation. To get low latency, a large portion of the BW must be reserved. The TDM-based GS solutions of ÆTHEREAL and NOSTRUM are also examples of FFQ-type schemes. Since these NoCs are globally synchronous, the latency through the network can be guaranteed to one clock cycle per hop, however the latency in accessing a connection at the source is still inversely proportional to the reserved BW. Also, in order to realize such a low per-hop latency, explicit end-to-end flow control mechanisms are required, as the connections are not independently buffered. To provide better bounds on the latency, decoupled from the BW guarantees, a different scheme is needed.

Macro networks are of a globally asynchronous nature, since it is obviously not possible to implement clock level synchronization among network nodes in a wide area network. This makes them somewhat similar to asynchronous NoC. In employing FFQ-type solutions to GS, latency problems as described above for asynchronous NoC are a well known drawback. In [20] a service discipline called rate-controlled static-priority (RCSP) queueing was introduced to overcome these drawbacks. An admission controller assigns an eligible transmission time to all incoming packets. When this point of time arrives, the packets are queued in a static priority queue (SPQ). This way not only BW guarantees but also latency guarantees can be provided, independently of each other. The admission control however requires that the node has a local notion of time. This makes it unsuitable for implementation in an asynchronous NoC. Another drawback of the method is that it is non-work-conserving, meaning that the router may be idle, even though there are packets in the channel queues, waiting for their eligible transmission time. This reduces the efficiency of using the available network resources, and even if the latency bounds are respected, the average connection latency and the link utilization are reduced. A work-conserving version was also proposed in [20], but this introduces the overhead of an extra *stand-by* queue, which is a BE queue working in parallel with the RCSP queues.

### 3.4. Requirements for GS in NoC

Our proposal to the requirements of a solution for GS in a NoC is that it (i) is simple in order to facilitate high operation speed and low hardware overhead, (ii) is work-conserving in order to make efficient use of network resources, and (iii) can provide bounds on latency and BW which are decoupled, or at least not inversely dependent on

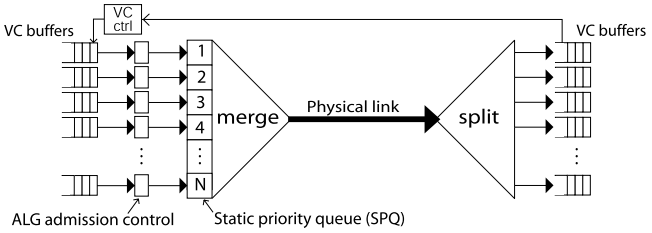



**Figure 3. A complete ALG link. The static priority queue (SPQ) prioritizes access to the link, providing latency guarantees, the admission control makes sure that the flow of flits adheres to the conditions required by the SPQ, and the VC control ensures non-blocking behaviour.**

each other. Additionally a requirement to a solution for GS in an *asynchronous* NoC is that it (iv) does not require a notion of time, neither local nor global. ALG conforms to all of these requirements, and is thus a valid solution to providing GS in both synchronous and asynchronous NoC.

In Section 4 we explain the ALG scheduling discipline, demonstrating its use in providing latency and BW guarantees on a shared link. As described in Section 2, in the MANGO router architecture, link access guarantees are sufficient to provide end-to-end guarantees for a connection. Note however that ALG-based access can be applied to any shared media. Also, though our implementation is based on asynchronous circuits, ALG is not restricted to such. However, the fact that ALG does not require a notion of time makes it particularly suitable for asynchronous systems.

## 4. ALG Scheduling

In this section we explain the ALG scheduling discipline. We first provide an intuitive understanding of its workings, and thereafter prove formally that it works. All indications of time in the following are quantized using the time unit *flit-time*. A flit-time is defined as the time it takes to complete one handshake cycle on the physical link. VC control measures ensure that no flits will stall on the link, thus the duration of such a handshake is well defined. The circuits being asynchronous, naturally the flit-time is not constant throughout the entire network. However we assume the flit-time to be fairly uniform.

Figure 3 shows the complete ALG link. The ALG admission control and the static priority queue (SPQ) implement the ALG scheduler. The VC control wraps around these. How these three sub-systems work together to provide latency and BW guarantees across multiple VCs sharing a physical link will become clear in the following. The principles of ALG scheduling are best understood from the inside out. The SPQ prioritizes VCs, providing latency guarantees accordingly, but only under certain conditions. The admission control makes sure that these conditions are met. The VC control mechanism ensures that flits are transmitted on the shared link only if there is free buffer space at the receiving end, thus preventing flits from stalling on the link and invalidating the latency and BW guarantees.

### 4.1. Prioritized Channels

In order to provide a latency guarantee, it is necessary to provide bounds on the link access time. Looking at Figure 2, envision flits arriving on channels A to D at random but large intervals. Now consider the channels being serviced by priority, A having the highest priority. Flits arriving on A will always be serviced immediately, thus it is guaranteed that the maximum link access time is one flit-time, i.e. the time it would take to finish a potentially on-going transmission. Since we – at this point – make the simplifying assumption that there is a large interval between flits arriving on A, flits arriving on B will wait for A no more than once before they are serviced. Thus flits on B will be delayed a maximum of two flit-times, since they will maximally wait for an on-going transmission to finish *and* for a transmission on A. Likewise, C will wait a maximum of three flit-times, etc. As a result, the maximum link access time is proportional to the priority of the channel. This – the basic foundation of ALG – is the functionality that is implemented by the SPQ in Figure 3.

### 4.2. Admission Control

The discipline explained above requires a large flit interval. This is not always possible to guarantee, in particular in an asynchronous network with distributed routing control. Even if a specific flit interval is provided at the source, the network may introduce jitter to the data stream, causing the interval requirement to be invalidated somewhere inside the network [19]. This necessitates an *admission control* stage, which regulates admission to the SPQ. In Figure 3, the ALG admission control is illustrated as boxes in

Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'05)
1522-8681/05 $20.00 © 2005 IEEE

front of the SPQ. This is similar to RCSP used in macro networks, in that it also implements an admission control stage and an SPQ. In RCSP however, admission is based on the local timing of the channels. This is not possible in a fully asynchronous system, which has no notion of time at all.

The condition, to be implemented by the ALG admission control for the latency bounds of the SPQ not to be invalidated, is that a flit on a given (higher priority) VC can stall a flit in another (lower priority) VC only once. This can be achieved by looking at the flits waiting in the SPQ when a flit is contending for access on a given VC. In order not to invalidate the latency guarantee of flits on lower priority VCs, all flits that have been waiting while the preceding flit on the given VC was being prioritized in the SPQ, must be serviced before a new flit is admitted. This is ensured by sampling the occupation of the SPQ when a flit is being transmitted on the link. Once all the flits waiting in the SPQ at this point of time have been serviced, a new flit on the same VC can be admitted. Thus flits on VCs of lower priority will be stalled a maximum of one flit-time by flits on each higher priority VC. Note that when a given flit is granted access to the link there will only be flits waiting on lower priority VCs, since by definition of the SPQ functionality, all flits on higher priority VCs will have already been serviced.

Figure 4 illustrates ALG by example. It is seen how the latency guarantee of the B and C queues are being met. The A queue has too many flits coming in, and is thus being inhibited by the admission control. The reason for the burst on A might be found at an earlier point in the network, due to A flits being transmitted very quickly (faster than the guaranteed latency bound) on previous links.

### 4.3. Latency and Bandwidth Guarantees of ALG

In this section we will state the latency and BW guarantees provided by an ALG connection. The results will be deducted formally in Section 4.4.

The service guarantees of ALG are characterized by the priority level of each VC reserved for the connection, as well as the total number of VCs on each link. Consider a connection for which the VCs with priority levels $Q_1, Q_2, \ldots, Q_X$ have been reserved on a sequence of ALG links $1, 2, .., X$. Each link implements $N$ VCs. The links provide a bound of $Q_1, Q_2, \ldots, Q_X$ flit-times on the link access time. This is so, under the condition that a flit interval of $t_{interval} \geq N + Q_{max} - 1$ flit-times is respected at the source, $Q_{max}$ being the maximum $Q$ value on the sequence of VCs. This is the so called *interval condition*, which will be derived in Section 4.4. The interval condition is also an access rate guarantee for the connection, and as such characterizes the BW guarantee of the connection, in terms of a fraction of the full link capacity: $BW_{min} = BW_{min}[Q_{max}] = BW_{link}/(N + Q_{max} - 1)$.

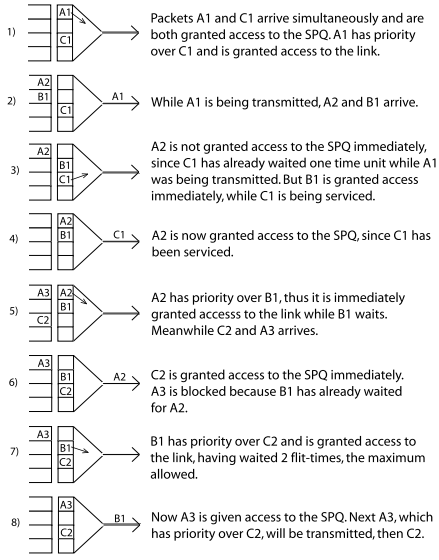



**Figure 4. ALG operational example.**

Note that the sum of the BW guarantees, on each of the $N$ VCs on a given link, results in less than 100% of the total link capacity. For a link implementing 8 VCs, a maximum of $BW_{min}[1] + BW_{min}[2] + .. + BW_{min}[8] = 73\%$ of the total link BW can be reserved. This is acceptable since most networks will need to allocate BW for the support of BE traffic, alongside the GS connections. The important point is to note the fact that the latency guarantee provided by ALG is decoupled from the BW guarantee. Increasing $N$, the number of VCs on a link, the BW guarantee can be made arbitrarily small while still maintaining a link access time of down to one flit-time. Thus latency critical connections with low BW needs, e.g. interrupts, are supported without the need to over-allocate BW.

Although existing (synchronous) GS disciplines for NoC based on TDM-type BW allocation realize a one flit-time per hop latency, the initial connection access latency still causes the total end-to-end latency to be inversely proportional to the BW guarantee. ALG provides instant access to the GS connection, as long as the interval condition is met. Also one may note that the forward latency per stage in an asynchronous network can be made very small, much less than a clock cycle of a comparable synchronous circuit. Thus while ALG guarantees a bound on the latency, an asynchronous NoC also potentially has a much lower minimum latency. In this lies a major advantage of implementing NoC using asynchronous versus synchronous circuits.

### 4.4. Proof

In the following we will prove that if the interval condition is respected at the source, a bound on the end-to-end connection latency can be made. The admission control might hold a flit, but only if the flit is ahead of its global schedule, causing the flit intervals observed locally to be shortened. The proof consists of two parts. In the first part we prove that the ALG discipline works for a single link. We first show that the first flit transmitted on a connection meets its latency requirements, or *makes its deadline*. Then we show that any flit following a flit that made its deadline, and which adheres to the interval condition, will also make its deadline, and from this we reach the value of the interval. By induction, all flits adhering to the interval condition make their deadlines. In the second part of the proof, we prove that for a sequence of ALG links a flit will make its deadline at each link, if the interval condition is respected at the source. Thus the end-to-end latency for the connection is bounded by the sum of the latency guarantees on each link, regardless of the interval condition being invalidated inside the network.

***The Single Link Theorem:*** *On an ALG link implementing $N$ VCs, all flits on VC Q will be guaranteed a maximum link access time of $Q$ flit-times under the flit interval condition of $t_{interval} \geq N + Q - 1$ flit-times.*

***Proof:*** Take a given link implementing $N$ VCs each corresponding to a priority level $1, 2, 3, \ldots, N$ in the SPQ. The first flit arriving on a given VC Q $\in$ {1, …,N} will be granted access to the SPQ immediately. In the SPQ it will wait for a maximum of $Q$ flit-times before being granted access to the link. Thus it makes its deadline, which is bounded by a maximum link access time of $Q$ flit-times.

Now consider on Q, a flit A which was granted access to the SPQ immediately thus making its deadline, and a flit B following flit A, also on Q. Flit B arrives $t_{interval}$ flit-times after flit A. Flit A was waiting 0 flit-times for access to the SPQ, and a maximum of $Q$ flit-times in the SPQ. A maximum of $N - Q$ flits, the number of VCs of lower priority than Q, were waiting in the SPQ when flit A was granted access to the link. According to the ALG discipline, these must all be transmitted before the next flit on Q is granted access to the SPQ. In a worst case scenario, a maximum of $Q-1$ flits, the number of VCs of higher priority than Q, can take priority over the $N-Q$ flits that must be transmitted before the admission control of Q admits flit B to the SPQ. The sum of these partial delays indicate the maximum time that can pass between one flit on Q and the following being admitted to the SPQ, $Q + (N - Q) + (Q - 1) = N + Q - 1$. This means that if $t_{interval} \geq N + Q - 1$ flit-times, then the flit B will be sure to be granted access to the SPQ immediately, and waiting a maximum of $Q$ flit-times in the SPQ, it too will make its deadline.

Thus, under the interval condition, since any flit following a flit which made its deadline will itself make its deadline, and since the first flit makes its deadline, by induction all flits will make their deadlines. □

We now show that, for a sequence of ALG links, even if the interval condition is invalidated locally, due to jitter being introduced in the network, ALG ensures that all flits make their deadlines at each link. Thus the end-to-end latency bound is the sum of the latency bounds at each link. At this point, we are still assuming that there is always enough buffer space in the nodes. In Section 5 we strengthen the proof, calculating the buffer requirement.

***The Sequence of Links Corollary:*** *Under the assumption that there is always enough buffer space, for a given connection having reserved VCs $Q_1, Q_2, \ldots, Q_X$ on a sequence of $X$ ALG links each implementing $N$ VCs, the latency bound is the sum of the latency bounds at each link, under the condition that a flit interval of $t_{interval} \geq N + Q_{max} - 1$ flit-times is respected at the source. Here, $Q_{max}$ is the maximum of $\{Q_1, Q_2, \ldots, Q_X\}$.*

***Proof:*** Consider a link on the connection in question, on which VC Q $\in \{Q_1, Q_2, \ldots, Q_X\}$ has been reserved, and a flit A on the connection, which has made its deadline on that link. Since the flit made its deadline, according to the proof of the Single Link Theorem above, the admission control will open for admission to the SPQ of a proceeding flit B, on the same VC, a maximum of $N + Q - 1$ flit-times after flit A was granted access to the SPQ. Since flit A made its deadline, it was on or ahead of its schedule. If flit B is further ahead of its schedule than flit A, it will arrive less than $N + Q - 1$ flit-times after flit A was granted access to the SPQ, and the admission control might not grant it access to the SPQ immediately. At the latest $N + Q - 1$ flit-times after flit A was granted access, flit B will be sure to be granted access. Their separation at the source was $N + Q_{max} - 1$ flit-times. It will be this or less now, so flit B will be at least as far ahead of its schedule as flit A. Thus it will also make its deadline. If flit B on the other hand is less ahead of its schedule than flit A – due to congestion at an earlier stage in the network – it will arrive more than $N + Q_{max} - 1$ flit-times after flit A was granted access to the SPQ. It will thus be granted access immediately, and make its deadline.

The first flit transmitted on a connection makes its deadline, since it is not stalled in the admission control of any link. Since any flit, following a flit making its deadline, will itself make its deadline, by induction all flits on the connection will make their deadlines, at all links. □

The minimum sustainable BW follows from this:

***The Minimum Bandwidth Corollary:*** *On a given connection which has reserved VCs $Q_1, Q_2, \ldots, Q_X$ on a sequence of $X$ ALG links each providing a total bandwidth of $BW_{link}$, and each implementing $N$ VCs, the minimum bandwidth sustained will be $BW_{min} = BW_{link}/(N + Q_{max} - 1)$. Here, $Q_{max}$ is the*

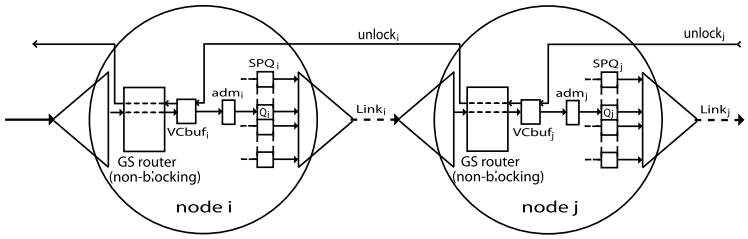



**Figure 5. Model of a section of a sequence of VCs reserved by a connection.**

*maximum of* $\{Q_1, Q_2, \ldots, Q_X\}$.

***Proof:*** According to the Sequence of Links Corollary, all flits on an ALG connection, adhering to the interval condition of $t_{interval} \geq N + Q_{max} - 1$ flit-times, have a bounded latency. Thus a stream of flits can be transmitted at a flit rate of at least $1/(N + Q_{max} - 1)$ of the total flit rate supported by a link, without causing congestion. From this follows directly that the sustainable bandwidth is at least: $BW_{min} = BW_{link}/(N + Q_{max} - 1)$. □

## 5. Buffers

In the previous section, we have assumed that flits flow freely in the network, constrained only by the ALG link access scheduling discipline. Since this work targets lossless networks, in which flits are never dropped, each link must also implement back pressure flow control, ensuring that a flit can only be transmitted on a VC if the receiving end has free buffer space. This introduces an extra layer of admission control, the VC control shown in Figure 3. The VC control wraps around the ALG admission control and the SPQ, only letting flits through if the receiving VC buffer indicates that it has free space. A flit must only be presented to the ALG admission control if it can move freely to the receiving end of the link. Otherwise the latency guarantees provided by the ALG discipline may be invalidated, by flits stalling on the link. On the other hand, a flit must not be unduly delayed by the VC control, so that it is caused to miss its deadline, again invalidating the ALG latency guarantees.

In this work we employ *share-based* VC control, which we have described in [4]. The scheme, illustrated in Figure 6, uses a single wire per VC to implement non-blocking access to a shared media, e.g. a link. After admitting a flit the *sharebox* locks, not allowing further flits to pass. The flit passes across the media, to the *unsharebox* at the far side. The unsharebox implements a latch, into which the flit is accepted. When the flit in turn leaves the unsharebox, the *unlock* control wire toggles. This unlocks the sharebox, admitting another flit to the media. As long as the media is deadlock free, no flit will stall within it.

As illustrated in Figure 5, we model a connection as a sequence of ALG links, with a direct circuit between the input port and the VC buffer reserved for the connection. As explained in Section 2, this assumption is valid for the MANGO router architecture in which the GS router implements output buffers and non-blocking switching [5]. The VC buffers in the figure implement unshare- and shareboxes on their inputs and outputs respectively. Latencies involved are the link access latency $t_{access}$ which is the time it takes for a flit to be granted access to the link, the link forwarding latency $t_{link}$ which is the latency of a flit across the link, through the GS router and into the next VC buffer, once link access has been granted, and the unlock latency $t_{unlock}$ which is the time it takes for the unlock signal to travel back to the sharebox in the previous VC buffer, indicating that another flit can be granted access to the link. All latencies apart from the link access latency are constant, as no congestion occurs.

The end-to-end latency bound of a connection consisting of a sequence of $X$ ALG links, each implementing $N$ VCs, is similar to $t_{total}$ introduced in Section 3: $t_{end2end} = t_{access1} + t_{link1} + t_{access2} + t_{link2} + \ldots + t_{accessX} + t_{linkX}$. For simplicity $N$ is herein considered to be the same on all links. The link access time is determined by the priority, $Q_1, Q_2, \ldots, Q_X$, of the VC reserved at each link: $t_{access1} = Q_1$ flit-times, $t_{access2} = Q_2$ flit-times, etc. The maximum $Q$ on the connection, $Q_{max}$, dictates the BW guarantee of the connection, according to Section 4.3, since this is the bottleneck of the path: $BW_{min} = BW_{link}/(N + Q_{max} - 1)$.

We now need to determine the requirements for the

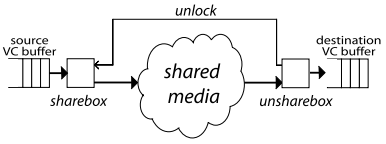



**Figure 6. Share-based VC control.**

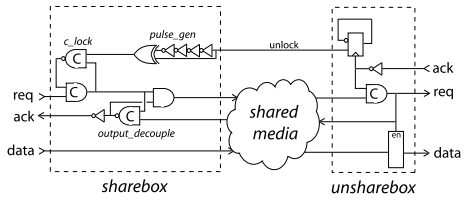



**Figure 7. Share- and unsharebox schematic.**

sharebox to always be unlocked when a flit matures for access to the SPQ, i.e. when it is 0 time ahead of its schedule. If this is so, the flit will be presented to the ALG admission control, and according to the ALG discipline, it will make its deadline. In the following, we will prove that under the flit interval condition and the *link cycle* condition that $t_{link} + t_{unlock} < N - 1$ flit-times, a single element VC buffer is enough to allow the ALG scheduling discipline to function properly.

***The Single Buffer Theorem:*** *Under the flit interval condition $t_{interval} \geq N + Q_{max} - 1$ flit-times and the link cycle condition $t_{link} + t_{unlock} < N - 1$ flit-times, a single element flit buffer, for each VC in each node, is enough to ensure the validity of the Sequence of Links Corollary.*

***Proof:*** As illustrated in Figure 5, consider a section of a connection having reserved VCs $(.., Q_i, Q_j, ..)$ on a sequence of ALG links, each implementing $N$ VCs. The VC buffers VCbuf$_i$ and VCbuf$_j$ each have buffer space for one flit. At reset they are empty, thus the first flit transmitted on the connection is not limited by VC control, and will according to the ALG discipline make its deadline.

Now consider a flit B following a flit A which is making its deadlines. Since flit A is making its deadlines, it will gain access to SPQ$_j$ latest at a time 0 which corresponds to it being 0 time ahead of its schedule. At this time VCbuf$_j$ will signal VCbuf$_i$ that it is ready to accept another flit. Thus VCbuf$_i$ will open its output for the next flit, flit B, at a time $t_{unlock}$ later. Flit A must have left SPQ$_i$ no later than at time $0 - t_{link}$, thus adm1 will allow flit B to enter SPQ$_i$ no later than $0 - t_{link} + N - 1 = N - 1 - t_{link}$ flit-times. If this time is later than the time VCbuf$_i$ lets flit B through, then VCbuf$_i$ will not be the limiting agent of the flow. The requirement for the VC control not to be the limiting agent in the system is thus: $N - 1 - t_{link} > t_{unlock} \Rightarrow t_{link} + t_{unlock} < N - 1$ flit-times. This constitutes the link cycle condition. If the link cycle condition holds, flit B will arrive at VCbuf$_j$ at a time: $Q_1 + t_{link} + N - 1 - t_{link} = Q_1 + N - 1$ flit-times, which is less or equal to the required flit interval of $Q_{max} + N - 1$ flit-times. Thus flit B made its schedule in arriving at adm$_j$.

Under the interval condition of a minimum flit interval of $Q_{max} + N - 1$ flit-times at the source, and under the link cycle condition that $t_{link} + t_{unlock} < N - 1$ flit-times, any flit following a flit which made its deadline will also make its deadline. Since the first flit makes its deadline, by induction all flits on the connection make their deadlines. □

## 6. Implementation

According to Figure 3, an ALG link consists of three basic subsystems: VC control, admission control and SPQ.

### 6.1. VC Control

The functionality of the VC control scheme that we employ, which we first presented in [4], was described in Section 5. Figure 7 shows the schematic for our implementation of the share- and unshareboxes of one VC. The single wire unlock signal functions as a 2-phase acknowledge. The pulse generated by *pulse_gen* must be long enough to reset the C-element *c_lock*. The *output_decouple* circuit at the output of the sharebox decouples the shared media from the VC. Thus a free flow of flits is ensured, regardless of individual VCs being slow.

### 6.2. Admission Control

The novelty of ALG scheduling lies in the admission control stage, which controls the flow of flits, allowing the SPQ to provide appropriate latency bounds.

Each channel of the admission control implements a status register of one bit for each channel of lower priority. When one or more of the status bits of a given channel are set, the admission control stops admission of flits, to the SPQ, on that channel. When a flit on the channel is granted access to the link, the status bits are set according to a snapshot of the *occupancy* of the SPQ. The occupancy indicates which channels that have flits waiting in the SPQ, while the given channel is granted access to the link (being prioritized). The status bits are subsequently reset as these waiting flits are granted access to the link. When all have been transmitted, the status bits are all clear, and the admission control admits another flit on the given channel.

Figure 8 shows the schematic of the admission control for channel $n$. The status bit registers $[n - 1..0]$, one for each lower priority channel, are implemented as RS-latches. Consider channel $n$ as being the highest priority channel contending for access to the link at a given time. The SPQ generates the occupancy vector, and its value is stable while the acknowledge of $n$ is high. The *set* inputs of a status bit registers is a logical AND of $n$'s acknowledge and the occupancy vector. This way the appropriate status bits are set according to the occupancy of the SPQ when the channel is granted access to the link (indicated by its acknowledge going high). The *reset* inputs of the status bit registers are simply connected to the acknowledge signals of the corresponding channels. When a channel is granted access to the

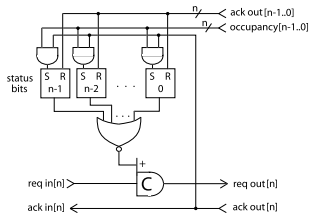



**Figure 8. Admission control schematic.**

link, its acknowledge goes high, and accordingly the status bit corresponding to this channel, in the admission control of each of the higher priority channels, is reset. When all status bits are low, the input request is allowed to propagate to the output. Since the local acknowledge causes the status bits to be set, a C-element – rather than an AND-gate – is needed in the request path. This ensures that the output request is not lowered until the input request is lowered.

Note that *set* and *reset* of the status bit registers are mutually exclusive, since only one channel can gain access to the link at a given time.

### 6.3. Static Priority Queue

The SPQ prioritizes the VC queues. At any time, the input with the highest priority is serviced before any queue of lower priority. Our SPQ is based on the SPQ in [11], which is based on ideas first presented in [6].

## 7. Results

We have implemented a 16-bit, 8-VC ALG link using commercially available 0.12 $\mu$m CMOS standard cells. Applying typical timing parameters, the design simulates at a speed of 702 MDI/s, corresponding to a flit-time of 1.42 ns. The shared, physical link implements a 3-stage pipeline. The cell area, i.e. prelayout, of the entire link was 0.012 mm$^2$, the core of the ALG scheduler (the admission control and the SPQ) taking only 0.004 mm$^2$. This is *less* than the area used in the simple fair BW allocation scheme presented in [4], implemented using an arbiter tree. This shows that the benefits of ALG are not at all costly in terms of area. But due to the priority arbiter there is a slight performance degradation (702 MDI/s as opposed to 795 MDI/s in an improved version of the solution presented in [4]).

A test setup emulating connections on a sequence of three ALG links was simulated. Two connections were observed; a *fast* connection reserving high priority VCs, and a *slow* connection reserving low priority VCs. The latencies of flits were recorded while random background traffic was induced on all other VCs. Figures 9 shows the distribution of flit latencies for different network loads, recorded

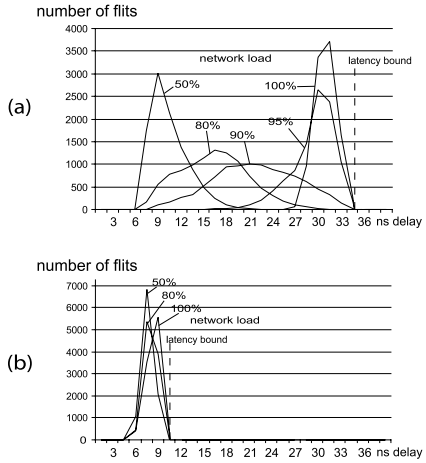



**Figure 9. Flit latency distribution versus network load: (a) slow path, (b) fast path.**

over 10000 flits. It is seen how even at 100% network load, the flits on the connections make their deadlines. As the network load is increased, the latency distribution graph pushes up against the latency bound, but never crosses it. Forward latency bounds from 3.6 ns/hop upwards, in increments of 1.42 ns (one flit-time), are obtainable. This includes the ALG access latency and the constant forward latency across the link (sharebox, merge, pipeline, split and unsharebox – app. 2.2 ns). The BW guarantee on the fast connection was $1/8 * 702\ MDI/s = 88\ MDI/s$, while it was $1/15 * 702\ MDI/s = 47\ MDI/s$ on the slow one.

Table 1 compares the guarantees of ALG to those of existing scheduling schemes used in NoC. In the table, $N$ is the number of VCs on each link and $h$ is the number of hops spanned by a given connection. TDM is used in synchronous NoCs, and provides bounds on the connection latency down to $N + h$, given that some sort of end-to-end flowcontrol is implemented. If not, the latency bound is reduced to the level of asynchronous fair-share, i.e. $(N+1)*h$. The table shows that ALG provides far better bounds on latency, and that it is generally more flexible in terms of variety of types of connections that can be instantiated.

## 8. Conclusion

We have presented a new flit scheduling discipline called *Asynchronous Latency Guarantee* (ALG) scheduling. ALG allows the reservation of GS connections in both synchronous and asynchronous networks, for which latency and bandwidth guarantees that are not inversely dependent on each other can be provided. It thus overcomes

| | Synchronous | Asynchronous | | |
|---|---|---|---|---|
| | **TDM** | **fair-share** | **ALG fast path** | **ALG slow path** |
| $t_{admit}$ | $N$ | 0 | 0 | 0 |
| $t_{access}$ | 1 | $N$ | 1 | $N$ |
| $t_{link}$ | 1 | 1 | 1 | 1 |
| **Latency** | $N+h$ | $(N+1)*h$ | $h$ | $(N+1)*h$ |
| **Bandwidth** | $1/N$ | $1/N$ | $1/N$ | $1/(2N-1)$ |

**Table 1. Latency and bandwidth guarantees of different GS schemes.**

the drawbacks of present solutions based on fair-fluid queuing (FFQ): time division multiplexing (TDM) in synchronous and fair-share in asynchronous networks. A fully asynchronous 16-bit, 8-VC ALG link was implemented using commercial 0.12 $\mu$m CMOS standard cells. Simulations show the correct functionality of the circuit, in accordance with the theoretical ground work.

The results clearly indicate that the ALG discipline is a valid scheme for providing hard latency and BW guarantees in shared interconnection networks.

## Acknowledgment

The authors thank Tiberiu Chelcea for a standard cell mutex, Thomas Bolander for reviewing the proofs and several anonymous reviewers for helpful comments.

## References

[1] International technology roadmap for semiconductors (ITRS) 2001. Technical report, International Technology Roadmap for Semiconductors, 2001.

[2] Open Core Protocol Specification, Release 2.0. www.ocpip.org, 2003.

[3] L. Benini and G. D. Micheli. Networks on chips: A new SoC paradigm. *IEEE Computer*, 35(1):70 – 78, January 2002.

[4] T. Bjerregaard and J. Sparsø. Virtual channel designs for guaranteeing bandwidth in asynchronous network-on-chip. In *Proceedings of the IEEE Norchip Conference*, 2004.

[5] T. Bjerregaard and J. Sparsø. A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE05)*, 2005.

[6] A. Bystrov, D. J. Kinniment, and A. Yakovlev. Priority arbiters. In *Proceedings of the Sixth International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 128–137. IEEE Comput. Soc., 2000.

[7] D. Chapiro. *Globally-Asynchronous Locally-Synchronous Systems*. PhD thesis, Stanford University, 1984.

[8] W. J. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *Proceedings of the 38th Design Automation Conference*, pages 684–689, June 2001.

[9] J. Dielissen, A. Radulescu, K. Goossens, and E. Rijpkema. Concepts and implementation of the phillips network-on-chip. In *Proceedings of the IPSOC'03*, 2003.

[10] J. Duato, S. Yalamanchili, and L. Ni. *Interconnection Networks - an Engineering Approach*, chapter 9, pages 475–558. Morgan Kaufmann, 2003.

[11] T. Felicijan, J. Bainbridge, and S. Furber. An asynchronous low latency arbiter for quality of service (QoS) applications. In *Proceedings of the 15th International Conference on Microelectronics*, pages 123–126. IEEE, December 2003.

[12] T. Felicijan and S. B. Furber. An asynchronous on-chip network router with quality-of-service (QoS) support. In *Proceedings IEEE International SOC Conference*, pages 274–277. IEEE, 2004.

[13] A. Jantsch and H. Tenhunen. *Networks on Chip*. Kluwer Academic Publishers, 2003.

[14] M. MillBerg, E. Nilsson, R. Thid, and A. Jantsch. Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip. In *Proceedings of the Design, Automation and Testing in Europe Conference (DATE'04)*. IEEE, 2004.

[15] R. Mullins, A. West, and S. Moore. Low-latency virtual-channel routers for on-chip networks. In *Proceedings of the International Symposium on Computer Architecture (ISCA2004)*. IEEE, 2004.

[16] J. Muttersbach, T. Villiger, K. Kaeslin, N. Felber, and W. Fichtner. Globally-Asynchronous Locally-Synchronous Architectures to Simplify the Design of On-CHIP Systems. In *Proc. 12th International ASIC/SOC Conference*, pages 317–321, Sept. 1999.

[17] L.-S. Peh and W. J. Dally. Flit-reservation flow control. In *Proceedings of HPCA: 6th International Symposium on High-Performance Computer Architecutre*, pages 73–84. IEEE Computer Soc., 1999.

[18] E. Rijpkema, K. G. W. Goossens, A. Radulescu, J. Dielissen, J. V. Meerbergen, P. Wielage, and E. Waterlander. Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip. In *Proceedings of the Design, Automation and Test in Europe Conference (DATE'03)*, pages 350–355. IEEE, 2003.

[19] H. Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proceedings of the IEEE*, 83:1374–1396, 1995.

[20] H. Zhang and D. Ferrari. Rate-controlled static-priority queueing. In *Proceedings of the Twelfth Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM'93*, pages 227–236. IEEE Comnput. Soc. Press, 1993.

Paper E

# An OCP Compliant Network Adapter for GALS-based SoC Design using the MANGO Network-on-Chip

Tobias Bjerregaard, Shankar Mahadevan, Rasmus Grøndahl Olsen and Jens Sparsø

Accepted for *Proceedings of the International Symposium on System-on-Chip*, Tampere, IEEE 2005.

# An OCP Compliant Network Adapter for GALS-based SoC Design Using the MANGO Network-on-Chip

Tobias Bjerregaard, Shankar Mahadevan, Rasmus Grøndahl Olsen* and Jens Sparsø
Informatics and Mathematical Modelling
Technical University of Denmark (DTU), 2800 Lyngby, Denmark
{tob, sm, jsp}@imm.dtu.dk, *s961394@student.dtu.dk

## Abstract

*The demand for IP reuse and system level scalability in System-on-Chip (SoC) designs is growing. Network-on-chip (NoC) constitutes a viable solution space to emerging SoC design challenges. In this paper we describe an OCP compliant network adapter (NA) architecture for the MANGO NoC. The NA decouples communication and computation, providing memory-mapped OCP transactions based on primitive message-passing services of the network. Also, it facilitates GALS-type systems, by adapting to the clockless network. This helps leverage a modular SoC design flow. We evaluate performance and cost of 0.13 $\mu$m CMOS standard cell instantiations of the architecture.*

## I. Introduction

Recent research has demonstrated the advantages of using shared, segmented interconnection networks – so called networks-on-chip (NoC) [1][2] – to implement global communication in system-on-chip (SoC) designs. By pipelining, NoC enables parallelism and counteracts the physical effects of long wires. NoC thus facilitates a scalable design approach, while accomodating the advert effects technology scaling has on wire performance. Managing the design flow of large complex chips however presents a non-trivial challenge in its own right. In order to effectively exploit the growing amount of on-chip resources a move from ad hoc SoC design to modular design is necessary. Orthogonalization of computation and communication is essential in order to enable fast design space exploration and IP reuse [3][4].

There are two basic paradigms of communication in computer systems: message-passing and memory-mapping. While much published NoC research focusses mainly on the implementation of message-passing networks, IP blocks for SoC designs, such as microprocessors and memories, are generally characterized by memory-mapped interfaces. There exist a number of socket specifications to this end, such as OCP (*Open Core Protocol*) [5] and AXI (*Advanced eXtensible Interface*) [6]. Since most NoCs are message-passing by nature, an adapter is needed.

There are a number of works published on the design of novel network architectures, but few publications have addressed issues particular to the design of an adapter module. In [7] an adapter implementing standard sockets was presented for the Æthereal NoC. The adapter however has quite a high forward latency. In [8] an OCP compliant adapter for the ×pipes NoC was touched upon. The adapter has a low area but it supports only a single outstanding read transaction. Also, both Æthereal and Xpipes are purely clocked designs, and as such do not address issues related to global synchronization in large-scale SoC designs.

In this paper we present an OCP compliant network adapter (NA) for the MANGO NoC (*Message-passing Asynchronous Network-on-chip providing Guaranteed Services over OCP interfaces*) [9]. The NA is a key component of a modular SoC design approach. Our contributions include identifying key issues of NA design and developing an efficient NA architecture. On the basis of several 0.13 $\mu$m CMOS standard cell instantiations, we evaluate the area and performance overheads of implementing NA tasks. The presented architecture enables GALS-type SoCs (*Globally Asynchronous Locally Synchronous*) [10], in that the NA implements synchronization between the clocked OCP sockets and the asynchronous or *clockless* MANGO NoC. Its design is a balanced mix of clocked and clockless parts, appropriately leveraging the advantages particular to either design style. It has a very low forward latency, is highly flexible with regards to the choice of transaction protocol and network packet format, and handles any number of outstanding transactions on the network.

The paper is organized as follows: in Section II we give an overview of MANGO. In Section III we describe the basic functionality of an NA, and in Section IV we detail the implementation of the MANGO NA. In Section V we present results, and Section VI provides a conclusion.

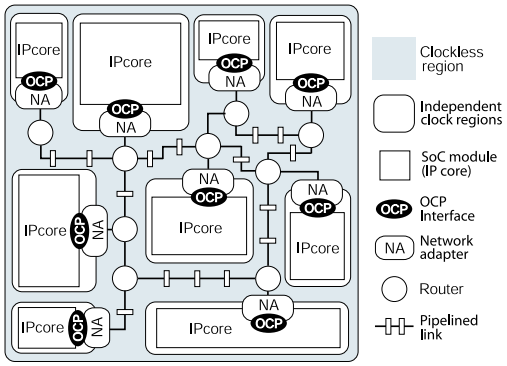



**Fig. 1. A MANGO-based SoC.**

## II. MANGO Overview

Figure 1 illustrates a MANGO-based SoC. IP cores reside in locally clocked regions, connected by a heterogeneous network through NAs. The NAs use the primitive message-passing functionality implemented by the network, to transparently provide the cores with read/write transactions. Key features of MANGO are (i) clockless implementation, (ii) guaranteed communication services and (iii) standard socket access points. These are briefly presented below. For more details, please refer to [9][11].

A clockless implementation of the network routers and links enables a GALS-type system, in which the integration of cores with different timing characteristics is inherently supported. An advantage of clockless circuits is that they use zero dynamic power when idle. Also, their forward latency can be made much lower than that of comparable clocked circuits, helping to minimize the performance overhead of using a network for e.g. memory accesses.

MANGO provides connection-less best-effort (BE) routing, as well as connection-oriented guaranteed services (GS), in terms of hard latency and bandwidth bounds. The predictability of GS makes system analysis much easier, leading to advantages at all levels of SoC design.

This paper addresses the third key feature: standard socket access points. While the choice is arbitrary, we implement these as OCP sockets. OCP is a family of synchronous point-to-point interfaces for memory-mapped read/write transactions. Support for other similar interfaces such as AXI is a trivial extension.

## III. NA Functionality

Though much simplified, the layered communication approach enabled by NoC is similar to the OSI model used in macro networks. The transport of messages in the network is decoupled from higher level transactions. The basic datagram of network-level flow control is called a *flit* (flow control unit). In the NA, OCP transactions are packetized and transmitted across the network as a stream of flits. A layered approach enables a mix of different sockets in the system, enhancing system-level composability. The overhead – of packetizing and depacketizing – is tolerated, as the approach enhances design-productivity: it becomes possible to design and verify SoCs hierarchically, and to plug together off-the-shelf IP cores from different vendors.

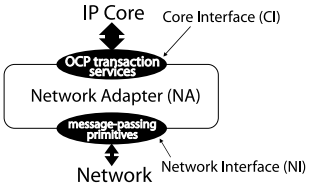



**Fig. 2. The Network Adapter provides high-level transactions to an IP core, based on the message-passing primitives of the network.**

As illustrated in Figure 2, the NA implements a Core Interface (CI) at the core side and a Network Interface (NI) at the network side. The functions implemented in the NA are *transaction handshaking* according to the CI protocol (in this case OCP), *encapsulation* of the transactions for the underlying packet-switched network, and *synchronization* between the IP core and the network.

## IV. NA Implementation

Read/write transactions require two types of NAs: an *initiator* connecting to a master core such as a microprocessor, and a *target* connecting to a slave core such as a memory. Figure 3 shows the structural design of the NAs, illustrating how the functions mentioned in Section III are implemented in separate modules. It is also seen how each NA consists of a *request* and a *response* path, and – orthogonally to this – of a *clocked* and a *clockless* part. The presented version of the MANGO NA supports all the required OCP v2.0 basic signals and a consistent subset of burst, thread and interrupt extensions, allowing support for single reads and writes, single-request burst reads and writes, threads, connections and interrupts. The initiator is configured through its CI, the target through its NI.

The NAs provide a number of input and output network ports, each corresponding to a GS connection or BE service. Output ports are pointed to using the OCP signal *MConnID*, and several threads may use each port without restrictions. Transmitting on a BE port, packets need a header flit for routing information. On GS connections no routing information is needed, as these implement a virtual circuit to another NA in the network [9].

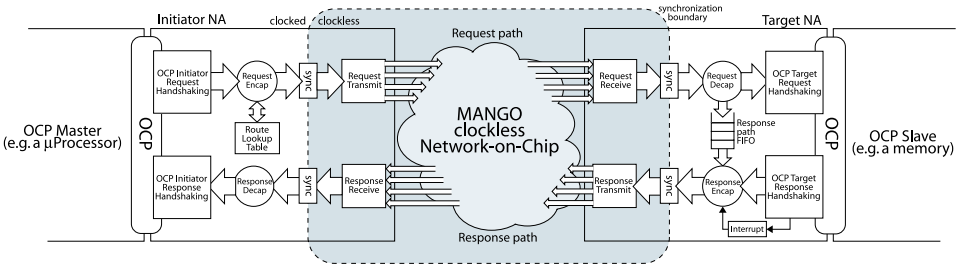



**Fig. 3. The initiator NA connects to a master core, the target NA connects to a slave core.**

An OCP transaction originates at a master core. In the initiator NA, the *OCP Initiator Request Handshaking* module implements part of the OCP slave socket with which the master transacts. The *Request Encap* module maps the OCP signals to packet signals. During BE transmissions it also uses the 8 MSBs of the OCP address field (*Maddr*) to look up a routing path, appending this to the payload (the routing path lookup table can be programmed through the OCP interface). This all takes 1 OCP clock cycle. In order to retain the flexibility of synthesized design, clocked circuits are used here. In any case, there is little performance to be gained by using clockless circuits in the handshaking and encapsulation modules, since the OCP interface is clocked itself. Serialization of packet flits is done by the *Request Transmit* module, in the clockless domain. This makes the flit serialization independent of the OCP clock speed. Also, the synchronization overhead is minimized by synchronizing an entire packet to the clockless domain in one go, instead of one flit at a time.

In the target NA packets are reassembled in the clockless *Request Receive* and decapsulated in the clocked *Request Decap* modules. The *OCP Target Request* and *Reponse Handshaking* modules then conduct an OCP transaction accordingly. Packet reassembly and buffering is done independently per port, such that a packet is forwarded to the clocked part of the NA only when the entire packet has arrived. During bursts however the packet is forwarded as soon as the control information and the first data word has arrived. If the transaction requires a response (e.g. a read), the response is encapsulated by the target NA, serialized in the clockless *Response Transmit* module and transmitted across the network. While a request is being processed by the slave core, the *Response path FIFO* stores the return path. Response packets are reassembled in the *Response Receive* module, decapsulated in the *Reponse Decap* module, and the transaction is completed by the *OCP Initiator Response Handshaking* module.

The OCP signal *MThreadID* is stored in request packets and returned in the corresponding response. The NAs thus handle any number of outstanding transactions, as they do not need to keep track of these themselves. If ordering is guaranteed by the network, e.g. on GS connections, multiple transactions can even be initiated on each thread.

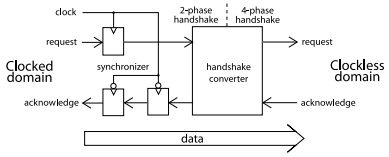



**Fig. 4. Synchronizing between the clocked and the clockless domains.**

Interrupts are implemented as *virtual wires*: an interrupt triggers the target NA to transmit an interrupt packet. Its destination is defined by configuring the target NA with a routing path or connection ID. Upon receiving this packet, the initiator NA asserts the local interrupt signal pin.

In order to minimize the synchronization overhead, we implement a 2-phase handshake channel, employing a two-flop synchronizer [12]. As illustrated in Figure 4 this is converted, in the clockless domain, to a 4-phase protocol compliant with the handshaking of MANGO. A complete clock cycle is available for resolving metastability in the flip-flop. Using flip-flop settling time and susceptibility window values of $\tau = 60\ ps$ and $T_W = 120\ ps$ (conservative values for a 0.13 $\mu$m standard cell technology), the mean time between failure (MTBF) at a 400 MHz OCP clock is calculated as follows [13][12]:

$$MTBF = \frac{e^{T/\tau}}{T_W f_C f_D} = 259 \times 10^9\ s (> 8000 yrs)$$

$T, f_C, f_D$ are the settling window (one clock cycle), and the clock and data frequencies, $f_D$ set to one fourth $f_C$.

The presented architecture is very flexible and modular. Changing the *Handshaking* modules a different CI protocol can be implemented, and customizing the packet format is merely an exercise of mapping CI signals to packet signals in the *Encap/Decap* modules. Adapting to a different network is done by modifying the *Transmit/Receive* modules.

**TABLE I. Results for 0.13 $\mu$m CMOS standard cell implementations (worst-case process corner) .**

| NA Type | OCP Speed | Area | | Port Transmit Speed | | OCP Read | | OCP Write | | OCP Interrupt | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | initiator | target | initiator | target | BE | GS | BE | GS | BE | GS |
| BE only | 400 MHz | 0.058 $mm^2$ | 0.020 $mm^2$ | 725 MHz | 1.08 GHz | 6 clks | - | 4 clks | - | 3 clks | - |
| BE + 3GS | 400 MHz | 0.064 $mm^2$ | 0.039 $mm^2$ | 483 MHz | 633 MHz | 8 clks | 6 clks | 5 clks | 4 clks | 4 clks | 3 clks |
| BE + 3GS | 200 MHz | 0.060 $mm^2$ | 0.037 $mm^2$ | 483 MHz | 633 MHz | 6 clks | 6 clks | 3 clks | 3 clks | 3 clks | 3 clks |

## V. Results

NAs with 32-bit OCP data and address fields and 32-bit network ports have been implemented using 0.13 $\mu$m standard cells. Netlist testing was done using the OCP tool CoreCreator[14], which provides OCP compliancy detectors. In the test setup an initiator NA was connected head-to-head with a target NA. Table I shows three classes of results for a number of NA instantiations: area, port transmit speed and transaction latency overhead.

The area values are purely cell area (pre-layout). Due to the GS input buffers, the area of the target NA supporting GS ports is twice that supporting only a BE port. The area of the initiator NA however is not very different in the two configurations, its main part being the BE routing path lookup table. A pure GS initiator would be smaller.

The port transmit speed indicates the flit speed on the clockless output ports. Since these are independent of the OCP clock, flits can be transmitted fast, even from a slow core. The NAs with only a BE port are simpler than those with multiple GS ports, hence their port speed is higher.

Finally, the table shows the latency overhead of conducting transactions through the NAs, relative to transactions of a master and slave core connected directly. The latency is displayed in OCP cycles. The overhead in the clockless circuits is independent of the OCP clock, hence the overhead in terms of *cycles* is less for slower cores.

BE transactions necessitate transmitting the routing path in a header flit, thus the total latency is increased. For a slow core however, an entire packet can be transmitted in less than one OCP clock cycle, even when needing a routing header, hence the overhead of BE transactions is not greater than for GS transactions.

In comparison with that of the Æthereal [7], the MANGO NA has a shorter total forward latency – only a single cycle for packetization. The area is also smaller, but this is not directly comparable; the Æthereal and the MANGO networks both implement support for BE and GS routing, but since this functionality is realized differently, the tasks required in the NAs are different. Compared with ×pipes [8], the MANGO NA implements more complex functionality in terms of GS connections, interrupts and support for multiple outstanding transactions. In the clockless MANGO NoC, there is no synchronization overhead in the forward path of entering the network, hence the total synchronization overhead is at least one clock cycle faster than both these purely synchronously clocked counterparts.

## VI. Conclusion

We have presented an OCP compliant NA architecture for the MANGO NoC. The NA enables modular, GALS-type SoC design by providing synchronous, memory-mapped interfaces, based on the clockless message-passing services of the network. The flexible architecture, which mixes clocked and clockless circuits, can easily be configured for different sockets and/or networks. Several 0.13 $\mu$m standard cell designs were implemented. Simulation results indicate that the performance and area overheads, of a layered approach to on-chip communication, are reasonable.

## References


[1] L. Benini and G. D. Micheli, "Networks on chips: A new SoC paradigm," *IEEE Computer*, vol. 35, no. 1, pp. 70 – 78, 2002.

[2] A. Jantsch and H. Tenhunen, *Networks on Chip*. Kluwer Academic Publishers, 2003.

[3] M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, J. Rabaey, and A. Sangiovanni-Vincentelli, "Addressing the system-on-chip interconnect woes through communication-based design," in *Proceedings of the 38th Design Automation Conference (DAC'01)*, June 2001, pp. 667 – 672.

[4] K. Keutzer, A. Newton, J. Rabaey, and A. Sangiovanni-Vincentelli, "System-level design: Orthogonalization of concerns and platform-based design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, pp. 1523 –1543, 2000.

[5] "Open Core Protocol Specification, Release 2.0," www.ocpip.org, OCP-IP Association, 2003.

[6] ARM, "AMBA AXI Protocol Specification, version 1.0," www.arm.com, ARM, March 2004.

[7] A. Radulescu, J. Dielissen, K. Goossens, E. Rijpkema, and P. Wielage, "An efficient on-chip network interface offering guaranteed services, shared-memory abstraction, and flexible network configuration," in *Proceedings of the 2004 Design, Automation and Test in Europe Conference (DATE'04)*. IEEE, 2004.

[8] S. Stergiou, F. Angiolini, S. Carta, L. Raffo, D. Bertozzi, and G. D. Micheli, "Xpipes lite: A synthesis oriented design library for networks on chips," in *Proceedings of the 8th Design, Automation and Test in Europe Conference*. IEEE, 2005.

[9] T. Bjerregaard and J. Sparsø, "A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip," in *Proceedings of Design, Automation and Testing in Europe Conference 2005 (DATE05)*, 2005.

[10] D. Chapiro, "Globally-asynchronous locally-synchronous systems," Ph.D. dissertation, Stanford University, 1984.

[11] T. Bjerregaard and J. Sparsø, "A scheduling discipline for latency and bandwidth guarantees in asynchronous network-on-chip," in *Proceedings of the 11th IEEE International Symposium on Advanced Research in Asynchronous Circuits and Systems*, 2005.

[12] R. Ginosar, "Fourteen ways to fool your synchronizer," in *Proceedings of the Ninth International Symposium on Asynchronous Circuits and Systems (ASYNCŠ03)*. IEEE, 2003.

[13] C. Dike and E. Burton, "Miller and noise effects in a synchronizing flip-flop," *IEEE Journal of Solid-State Circuits*, vol. 34, pp. 849–855, 1999.

[14] www.ocpip.org/socket/corecreator, OCP International Partnership.

Paper F

# Programming and Using Connections in the MANGO Network-on-Chip

Tobias Bjerregaard

*To be submitted.*

# Programming and Using Connections in the MANGO Network-on-Chip

Tobias Bjerregaard
Informatics and Mathematical Modelling
Technical University of Denmark (DTU)
2800 Lyngby, Denmark
tob@imm.dtu.dk

## Abstract

*The scaling of CMOS technology causes a widening gap between the performance of on-chip communication and computation. This calls for communication-centric design. The MANGO network-on-chip architecture enables globally asynchronous locally synchronous (GALS) system-on-chip design, while facilitating IP reuse by standard socket access points. Two types of services are available: connection-less best-effort routing and connection-oriented guaranteed service (GS) routing. This paper presents the core-centric programming model for establishing and using GS connections in MANGO. High predictability of communication is demonstrated in a MANGO-based GALS system.*

## 1 Introduction

While transistor speeds increase with each new CMOS fabrication technology, wire speeds worsen. Scaling wires, the resistance per mm increases. The capacitance stays roughly constant, being mostly due to edge effects [12]. Hence, as shown in Figure 1, the RC delay for a constant length wire increases [2]. With a projected processor speed of 40 GHz in 2016, the latency cost of driving data 1 mm across a chip, on broad, top-level, *global* wires, will be 32 clock cycles. While wire segmentation and pipelining can help, ultimately the result is a widening gap between communication and computation. Thus it has been evident for some time now, that data traffi cking – not processing – will be the performance bottleneck in future single-chip systems. This calls for a communication-centric design fbw.

In addition to physically oriented design challenges, chip designers are under increasing pressure to exploit the exponential availability of design resources, in terms of transistors and wiring layers. Between 1997 and 2002 the market demand reduced the typical design cycle by 50%. As a result of increased chip sizes, shrinking geometries and the availability of more metal layers, the design complexity increased 50 times in the same period [1]. To keep up with productivity requirements, a scalable design approach which allows a greater degree of modularity is pertinent.

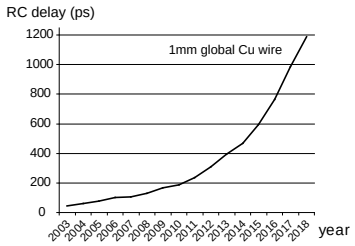



**Figure 1. Wire delay increases with technology scaling [2].**

It is generally accepted that segmented interconnection networks, so called networks-on-chip (NoC), constitute a realistic solution space to communication challenges of future system-on-chip (SoC) designs [10][4][13]. In previous works, we have presented novel solutions to a number of issues relevant to the development of such a NoC, MANGO (*Message-passing Asynchronous Network-on-chip providing Guaranteed services over OCP interfaces*) [6][8][7][5]. Key features of MANGO, which enable scalable, modular SoC-design are: (i) clockless implementation, (ii) guaranteed services (GS) and (iii) standard socket access points. A clockless implementation facilitates globally asynchronous locally synchronous (GALS) design [9][14][15], GS routing incurs predictability, accelerating design and verifi cation (as also argued in [11]), and OCP (*Open Core Protocol*) [3] sockets promote IP reuse.

As drafted in [13] Chapter 12, it is clear that a major challenge lies not only in the implementation of on-chip networks, but also in the optimal programmability and use of them. Effi cient and intuitive programming models are required, in order for software designers to take full advantage of the available hardware features. System design encom-

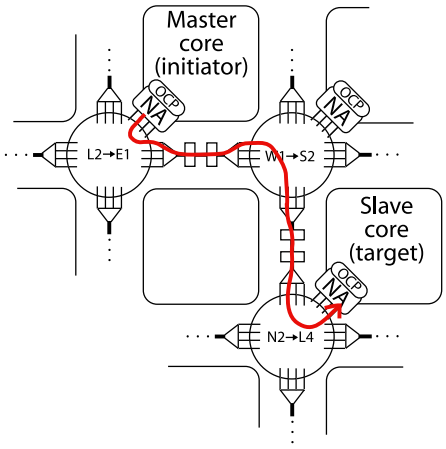



**Figure 2. A guaranteed service connection is established by reserving a sequence of virtual channels through the network.**

passes a range of software layers, from application, across middleware, to operating system (OS).

In this paper, I present the core-centric programming model for establishing and using GS connections in MANGO. This constitutes the boundary between hardware and software, defining the programmability on which OS system calls are based. I demonstrate with a 0.13$\mu$m MANGO-based system showing end-to-end performance results. In Section 2 I describe encapsulation of OCP transactions in packets, and two types of routing services available in MANGO. In Section 3 I explain the programming model for establishing GS connections, and in Section 4 I present experimental results from the demonstrator system. Section 5 provides a conclusion.

## 2 Packet Routing in MANGO

In a MANGO-based SoC, OCP compliant IP cores are attached to network adapters (NAs) as illustrated in Figure 2. The NAs encapsulate OCP requests and responses in packets, routing these as sequences of *flits* (flow control units) on the network. Packets may be either streamed on GS connections (see Section 2.1) or source-routed as connection-less best-effort (BE) traffic (see Section 2.2). Figure 3 details different types of packets generated by the NAs.The first bit in a flit is the end-of-packet bit, which is high only in the last flit of a packet. The first flit of BE packets is the header which holds the routing path. Transmitting on GS connections, this header flit is not needed as a connection uniquely defines a complete path, from source to destination.

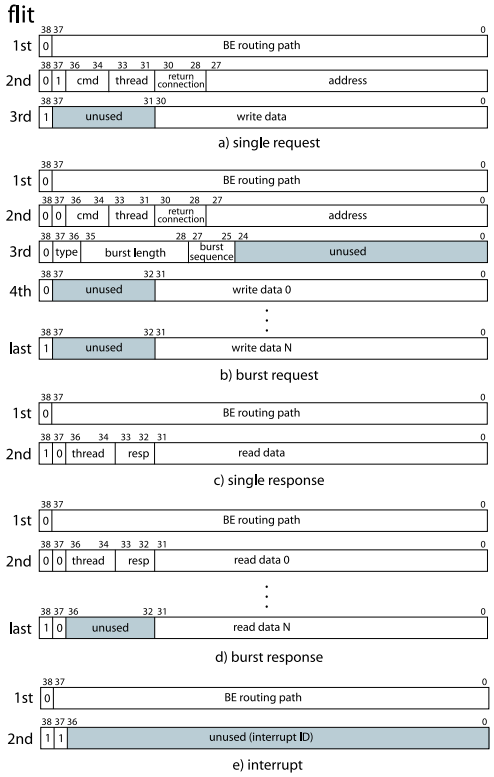



**Figure 3. Packet formats for encapsulating OCP transactions for network transmission.**

### 2.1 Guaranteed Service Connections

The routers in MANGO are output buffered, and the links implement a number of separately buffered virtual channels (VCs). Each VC on a link is associated with a certain *hop-guarantee*. This is the service guarantee – in terms of latency and bandwidth – in moving flits from the given VC buffer, to a VC buffer on an output in the next router. A GS connection is established by reserving a sequence of VCs through the network, as shown in Figure 2 (further detailed in Figure 4, which will be explained in Section 3). In each router, a mapping from an input VC to an output VC is made, hence an independently buffered multi-hop *virtual circuit* is established from source to destination. The end-to-end guarantee, $L_{end2end}$ being latency and $BW_{end2end}$ being bandwidth, is determined as:

$$\begin{aligned} L_{end2end} &= L_{hop1} + L_{hop2} + \ldots L_{hopX} \\ BW_{end2end} &= min(BW_{hop1}, BW_{hop2}, \ldots, BW_{hopX}) \end{aligned}$$

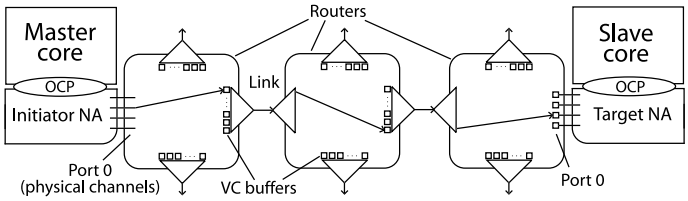



**Figure 4. A GS connection constitutes a sequence of reserved VC buffers through the network (a virtual circuit), and a mapping to it from the OCP socket.**

The latency is the sum of the per-hop latencies, while the bandwidth is the bottleneck of the path. A connection is accessed through the OCP interface of the NA, by addressing it with the OCP signal *MconnID*.

The switching of GS flits, inside the MANGO routers, is congestion-free, hence the hop-guarantees of the connection are determined by the link access scheduling scheme [7]. Herein we use so called ALG (Asynchronous Latency Guarantee) scheduling [8]. Each link implements 8 VCs, denoted $Q \in \{0, 1, \ldots, 7\}$. Of these, 0 through 6 can be used for GS connections, while the last is used for connection-less BE routing, as will be explained in Section 2.2. Since the network is clockless, we specify latency and bandwidth guarantees in terms of the time unit *flit-time* ($t_{flit}$), which is the time it takes to transmit one flit on a link. This corresponds to a clock cycle in a synchronous network. In MANGO, $t_{flit}$ depends on the actual link implementation: the link encoding and pipelining, the flit width, etc. We use delay insensitive dual rail encoding [16] in order to improve timing robustness in the system, pipeline the links, and the flits are 32-bit wide. This configuration results in $t_{flit}$ = 3.6 ns, in the worst-case process corner.

The hop-guarantees for VC $Q$ are [8]:

$$\begin{aligned} L_{hop} &= ((Q+1) * t_{flit}) + t_{link} \\ BW_{hop} &= \frac{1}{t_{flit} * (N+Q-1)} \end{aligned}$$

Here, $N$ is the total number of VCs on a link and $t_{link}$ is the forwarding latency of a flit from one VC buffer, across the link, through the next router, to the next VC buffer on the connection. VC control [6] ensures that a flit can only gain access to a link, if the target VC buffer is free, hence once access is granted, the flit will experience no congestion. Therefore $t_{link}$ is constant [7]. The latency guarantee is given on a flit by flit basis. The bandwidth guarantee marks the timing between two consecutive flits. If there are a large number of VCs on a link, this time may become large, since it is dependent on $N$. Looking at the packet formats in Figure 3, it is seen how non-burst reads, on GS connections, require only a single flit, likewise for non-burst responses and interrupts. This makes them particularly suitable for exploiting the ALG latency guarantees.

In the MANGO instantiation presented in this work, the links are pipelined. In clocked networks, pipelining has the side effect of increasing the forward latency by one clock cycle per pipeline stage. In clockless pipelines however, the forward latency is only part of the cycle time. Even though we have placed two pipeline stages between each router, the increase in forward latency is only 240 ps.

### 2.2 Best-Effort Routing

Packets in MANGO may also be routed in a connectionless fashion, as source-routed BE traffic. Hard performance guarantees can be made on GS connections because each stream is transmitted on a logically separate virtual circuit. No guarantees apart from completion and correctness are given for BE traffic, since all BE packets are transmitted on a shared subset of VCs.

OCP commands are issued as BE transactions by addressing connection 0 (setting *MconnID = 0* on the OCP interface). Write transactions on the BE connection are used to program the routers and NAs, setting up connections. The upper 8 bits of the 32-bit OCP address field are used as a global address space. Upon issuing an OCP transaction on connection 0, these bits are used as a lookup key in the route lookup table [5]. This table, which can be programmed through the OCP interface, maps the 8-bit global address to a routing path. The coarse geography of the system-level memory map is thus completely user definable. A BE routing path is a hop-by-hop specification of the path through the network.

## 3 Programming Model

As illustrated in Figure 4, a GS connection consists of a virtual circuit, a sequence of VCs forming a path through the network, and a mapping from an OCP connection ID to its start point. In order to establish a virtual circuit two pointers must be programmed into each router on the path. At the VC output buffer, a pointer *forward* on the path, to a VC buffer in the *next* router defines the data flow channel. This is the ***steer*** pointer, which uniquely defines an output port in this next router, and a VC on that port. Since flow

**Table 1. Memory map of a router.**

| Adr | description |
|---|---|
| x100 - x106 | outport 1, *steer0 - steer6* |
| x200 - x206 | outport 2, *steer0 - steer6* |
| x300 - x306 | outport 3, *steer0 - steer6* |
| x400 - x406 | outport 4, *steer0 - steer6* |
| x501 - x503 | inport 0, *select1 - select3* |
| x508 - x50E | inport 1, *select0 - select6* |
| x510 - x516 | inport 2, *select0 - select6* |
| x518 - x51E | inport 3, *select0 - select6* |
| x520 - x526 | inport 4, *select0 - select6* |

control is handled on a hop-by-hop basis [7], a fbw control channel *backwards* on the path, to the previous VC buffer, must also be established. This is the ***select*** pointer, which for each VC at each input port selects an output port and a VC on that port.

Having both ***steer*** and ***select*** pointers really incurs storing the same information twice in the network, but since it results in some very simple circuits, the overhead is accepted. In any case, the pointer storage elements are a very limited part of the total router area [7]. The pointers can be written to the routers by any OCP master in the system, using OCP write commands on connection 0 (the BE connection). Table 1 illustrates the memory map of a router. The presented system is constructed of 5x5 routers with 8 VCs on each bidirectional network port. Hence 5 bits are needed by each pointer. Hereof, 2 bits are used for identifying a port. A value of 0 maps to port 4. Routing back on a link is not allowed, and pointing to the same port number as the input, indicates pointing to the local port (port 0, to which the NA – and hence an IP core – is connected). The remaining 3 bits point to one of 8 VCs on each port. Note that pointing to VC 7 is illegal, as this VC is used for BE routing. When programming connections into the routers, a global view of existing connections is needed, in order not to allocate VC buffers which are already in use.

The NAs implement four physical channels on the NA-to-router interface (port 0), addressed by the OCP signal *MConnID*. As explained earlier, channel 0 (connection 0) is used to access the BE routing network. Channels 1-3 are used to access virtual circuits. To form a GS connection, a channel must be steered to the start point of the virtual circuit which forms the connection path. This is done by the *select* pointer of the channel's input in the router. The reuse of this pointer for steering purposes is feasible, since only one connection makes use of any one of the physical channels on port 0 (no VCs share the physical channel).

A connection to be used for read operations must also map to a return connection, at the slave core. A virtual circuit is setup for the response, and the target NA maps between its input channel and an output channel. Any combination of BE and GS routing may be used by a request and its response, as explained in [5].

The programmability described above is what the OS can use to establish GS connections in the network. Its use, together with route calculation algorithms, form the building blocks of OS system calls, used by the middleware of the IP cores. Examples of such system calls could be:

*int* **open**( *destination, service* ) : Opens a connection to the *destination* core, with the *service* guarantee. Returns a connection ID, or 0 for failure.

*int* **close**( *connection_ID* ) : Closes the *connection_ID* connection. Returns a meassure of success.

## 4 Demonstrator and Results

As depicted in Figure 2, our test system is composed of three routers, the bidirectional links between them pipelined in two stages. Results are based on simulations of a 0.13 $\mu$m standard cell implementation, using worst-case process corner timing parameters. First GS connections are established between the master and slave cores. Then a number of connections are setup between the other network ports, and these are loaded with random traffi c. This provides a variable background traffi c load, simulating the master/slave subsystem being part of a bigger system. In order to illustrate the GALS capabilities of MANGO, the master and slave cores are run at different frequencies, 250 MHz and 333 MHz. The master could be a microprocessor, while the slave could be a shared memory. While there are many solutions to gaining throughput in communication networks (e.g. pipelining), the fundamental drawback of scaling technologies, leading to the necessity for communication-centric design, concerns the latency of communication. Therefore latency will be the focus in the following.

The end-to-end latency $L_{conn}$ of using a GS connection for OCP transactions, can be broken down into components:

$$L_{conn} = L_{initiatorNA} + L_{fwd} + L_{congestion} + L_{serialization} + L_{targetNA}$$

In the following each component is explained.

$L_{initiatorNA}$ is the latency in the initiator NA (the master core's NA). It is one OCP clock cycle plus some forward latency in the clockless part of the NA [5].

$L_{fwd}$ is the forward latency, in the network, of a flit which does not encounter any congestion. It is $t_{link}$ for each link traversed, plus a latency for engaging the virtual circuit from the initiator NA, $t_{engage}$.

$L_{congestion}$ is latency due to stalling in the network. It is the time that a flit is waiting in a VC buffer, for access to a link. Its value depends on the link access arbitration.

$L_{serialization}$ is the latency penalty due to serialization of the packetized transactions into flits. This is dictated by the bandwidth guarantee of the connection: as explained

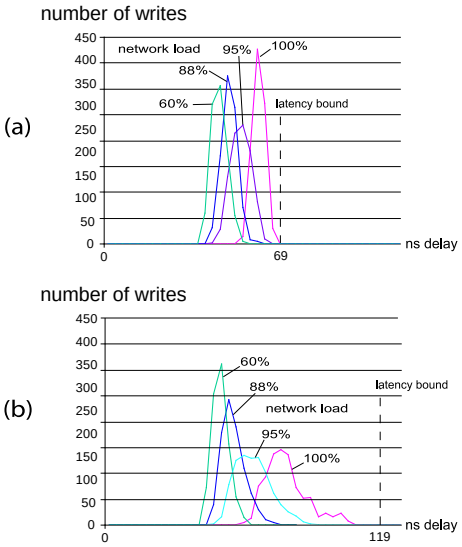



**Figure 5. Latency distribution of OCP write transactions on a) connection 1 and b) connection 2.**

in [8], the latency guarantee of ALG is given at the requirement of a separation, in time, between flits on the VC. This is the inverse of the bandwidth guarantee. Hence, in a fully loaded network, the separation between flits will be close to maximum (one over the bandwidth guarantee), while in an uncongested network it will be much lower, as more bandwidth than guaranteed is given.

$L_{targetNA}$ is the forward latency in the target NA (the slave's NA). Resynchronization takes one OCP clock cycle, while there is also half a cycle latency in the clocked part of the NA. In addition, there is some latency in the clockless part, plus an unknown latency of up to one OCP cycle, due to the uncertainty of the arrival time of the last flit. If this occurs immediately after the clock tick, the packet will need to wait a complete cycle before resynchronization can begin.

By simulating a transaction in an unloaded network, we can measure $L_{initiatorNA}$, $L_{fwd}$ and $L_{targetNA}$. $L_{serialization}$ is the difference between the arrival time of the first flit and the last (zero for single-flit packets, such as a GS read request). In an unloaded network, $L_{congestion}$ is zero.

Two connections between the master and the slave are tested: *conn1* and *conn2*. For *conn1*, low latency VCs have been reserved, while *conn2* has reserved VCs which provide worse latency guarantees. The connections consist of three hops: one from the initiator NA to the first VC in the virtual circuit, and one across each of the two links in the path. The hop from the initiator NA does not require access to a shared link, hence its latency ($t_{engage}$) is constant. From simulations, we obtain the values: $t_{engage} = 3.2\ ns$, $t_{flit} = 3.6\ ns$ and $t_{link} = 10.5\ ns$. Having reserved VCs 0 and 0 for *conn1*, and VCs 3 and 6 for *conn2*, on the two links between master and slave, the theoretical latency guarantees, of transmitting a flit on the virtual circuits, are:

$$\begin{aligned} L_{circuit1} &= t_{engage} + (1+1) * t_{flit} + 2 * t_{link} \\ &= 31.4\ ns \\ L_{circuit2} &= t_{engage} + (4+7) * t_{flit} + 2 * t_{link} \\ &= 63.8\ ns \end{aligned}$$

Note that $L_{circuit} = L_{fwd} + L_{congestion,max}$.

Now we get the total end-to-end latency guarantee by adding to this the latency of the NAs and the serialization penalty:

$$\begin{aligned} L_{conn1,max} &= L_{initiatorNA} + L_{circuit1} \\ &\quad + L_{serialization} + L_{targetNA} \\ &= 4.9ns + 31.4ns + 25.2ns + 7.5ns \\ &= 69ns \\ L_{conn2,max} &= 4.9ns + 63.8ns + 43.2ns + 7.5ns \\ &= 119.4ns \end{aligned}$$

The worst case NA latency occurs when the synchronization clock is just missed in the target NA. The serialization penalty for a write (2 flits) is one over the bandwidth guarantee of the connection.

Figure 5 shows end-to-end latencies of issuing OCP write transactions across the connections. Test results are sampled over 1000 transactions at different background loads. Read transactions simply result in a double up, plus the response time of the slave. Hence write commands are sufficient for illustration purposes.

The results illustrate how the latency does not exceed the guaranteed maximum, even under 100% link load. This shows how a high degree of predictability can be obtained in using the shared network, despite a shift in the use of the network by other entities in the system, and despite its asynchronous nature.

Table 2 shows typical examples (i.e. not worst-case) of the breakdown of the latency of write transactions on the two connections in an unloaded and fully loaded network scenario. It is seen that the forward latency is not exactly the same on the two connections, as one might expect. This is because the clockless implementation of the link access circuits is not symmetrical, since some VCs must be prioritized over others [7]. We see how the effect of $L_{congestion}$ is much smaller on connection 1 which has reserved low latency VCs. We also see that the serialization penalty is quite high (GS writes consist of two flits). Note that since

**Table 2. Examples of end-to-end latency of a write, on two connections at varying background loads.**

| | ***conn1* / 0% load** | ***conn1* / 100% load** | ***conn2* / 0% load** | ***conn2* / 100% load** |
|---|---|---|---|---|
| $L_{initiatorNA}$ | 4.9 ns | 4.9 ns | 4.9 ns | 4.9 ns |
| $L_{fwd}$ | 17.8 ns | 17.8 ns | 18.6 ns | 18.6 ns |
| $L_{congestion}$ | 0 ns | 12.3 ns | 0 ns | 35.0 ns |
| $L_{serialization}$ | 12.0 ns | 21.3 ns | 12.3 ns | 25.0 ns |
| $L_{targetNA}$ | 7.0 ns | 7.0 ns | 7.0 ns | 7.0 ns |
| ***Total*** | 41.7 ns | 63.3 ns | 42.8 ns | 90.5 ns |

the tested connections only traverse two links, the serialization penalty contributes a relatively large portion of the total. This value is independent of the number of hops on a connection, and will dwindle relatively on longer connections. The latency due to congestion is per-link on the other hand, hence it will grow on longer connections. The benefit of scheduling for hard latency guarantees in the link access thus increases as the connections get longer.

## 5 Conclusion

In this paper, I have demonstrated the programmability and end-to-end performance of guaranteed service connections in the MANGO NoC. I have described a programming model for setting up connections in the network, and shown how a high degree of latency predictability can be obtained in issuing OCP commands, despite different levels of background traffic. Such predictability is important in a modular system-on-chip design flow, as it facilitates analytical verification, and a decoupling of sub-systems. Also, globally asynchronous locally synchronous system composition is facillitated, by the network adapters synchronizing the clocked OCP interfaces with the clockless network. I have also shown how the performance penalty of inserting pipeline stages on the network links is minimal in a clockless network, as the forward latency of clockless pipelines is only a fraction of the total cycle time. The work illustrates the advantages of the architecture, both from a performance and programmability point-of-view.

## References


[1] The importance of sockets in soc design. White paper downloadable from www.ocpip.org.

[2] International technology roadmap for semiconductors (ITRS) 2003. Technical report, International Technology Roadmap for Semiconductors, 2003.

[3] Open Core Protocol Specification, Release 2.0. www.ocpip.org, 2003.

[4] L. Benini and G. D. Micheli. Networks on chips: A new SoC paradigm. *IEEE Computer*, 35(1):70 – 78, January 2002.

[5] T. Bjerregaard, S. Mahadevan, R. G. Olsen, and J. Sparsø. An OCP compliant network adapter for GALS-based SoC design using the MANGO network-on-chip. IEEE, 2005.

[6] T. Bjerregaard and J. Sparsø. Virtual channel designs for guaranteeing bandwidth in asynchronous network-on-chip. In *Proceedings of the IEEE Norchip Conference*, 2004.

[7] T. Bjerregaard and J. Sparsø. A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip. IEEE, 2005.

[8] T. Bjerregaard and J. Sparsø. A scheduling discipline for latency and bandwidth guarantees in asynchronous network-on-chip. In *Proceedings of the 11th IEEE International Symposium on Advanced Research in Asynchronous Circuits and Systems*. IEEE, 2005.

[9] D. Chapiro. *Globally-Asynchronous Locally-Synchronous Systems*. PhD thesis, Stanford University, 1984.

[10] W. J. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *Proceedings of the 38th Design Automation Conference*, pages 684–689, June 2001.

[11] K. Goossens, J. Dielissen, O. P. Gangwal, S. G. Pestana, A. Radulescu, and E. Rijpkema. A design flow for application-specific networks on chip with guaranteed performance to accelerate SOC design and verification. In *Proceedings of the Design, Automation and Test in Europe Conference, 2005.*, pages 1182–1187. IEEE, 2005.

[12] R. Ho, K. W. Mai, and M. A. Horowitz. The future of wires. *Proceedings of the IEEE*, 89(4):490 – 504, April 2001.

[13] A. Jantsch and H. Tenhunen. *Networks on Chip*. Kluwer Academic Publishers, 2003.

[14] T. Meincke, A. Hemani, S. Kumar, P. Ellervee, J. Oberg, T. Olsson, P. Nilsson, D. Lindqvist, and H. Tenhunen. Globally asynchronous locally synchronous architecture for large high-performance asics. In *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems 1999 (ISCAS'99)*, volume 2, pages 512 –515, June 1999.

[15] J. Muttersbach, T. Villiger, and W. Fichtner. Practical design of globally-asynchronous locally-synchronous systems. In *Proceedings of the Sixth International Symposium on Advanced Research in Asynchronous Circuits and Systems, 2000 (ASYNC 2000)*, pages 52–59. IEEE Computer society, April 2000.

[16] J. Sparsø and S. Furber. *Principles of Asynchronous Circuit Design - a Systems Perspective*. Kluwer Academic Publishers, Boston, 2001.

# Bibliography


[1] Arteris - the network-on-chip company. www.arteris.com.

[2] Silistix ltd. www.silistix.com.

[3] Sonics - smart interconnects. www.sonicsinc.com.

[4] Open Core Protocol Specification, Release 2.0. www.ocpip.org, 2003.

[5] J. Bainbridge and S. Furber. Chain: A delay-insensitive chip area interconnect. *IEEE Micro*, 22(5):16–23, October 2002.

[6] Luca Benini and Giovanni De Micheli. Networks on chips: A new SoC paradigm. *IEEE Computer*, 35(1):70 – 78, January 2002.

[7] Tobias Bjerregaard and Jens Sparsø. Virtual channel designs for guaranteeing bandwidth in asynchronous network-on-chip. In *Proceedings of the IEEE Norchip Conference*, 2004.

[8] Tobias Bjerregaard and Jens Sparsø. A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip. IEEE, 2005.

[9] Daniel Chapiro. *Globally-Asynchronous Locally-Synchronous Systems*. PhD thesis, Stanford University, 1984.

[10] William J. Dally and Brian Towles. Route packets, not wires: On-chip interconnection networks. In *Proceedings of the 38th Design Automation Conference*, pages 684–689, June 2001.

[11] John Dielissen, Andrei Radulescu, Kees Goossens, and Edwin Rijpkema. Concepts and implementation of the phillips network-on-chip. In *Proceedings of the IPSOC'03*, 2003.

[12] Kees Goossens. Networks on chip for consumer electronics. In *Proceedings of the International Summer School on Advanced Computer Architecture and Compilation for Embedded Systems (ACACES), 2005.*, 2005.

[13] Kees Goossens, John Dielissen, Om Prakash Gangwal, Santiago Gonzalez Pestana, Andrei Radulescu, and Edwin Rijpkema. A design flow for application-specific networks on chip with guaranteed performance to accelerate SOC design and verification. In *Proceedings of the Design, Automation and Test in Europe Conference, 2005.*, pages 1182–1187. IEEE, 2005.

[14] J.A.Kahle, M.N.Day, H.P.Hofstee, C.R. Johns, T. R. Maeurer, and D. Shippy. Introduction to the cell multiprocessor. *IBM Journal of Research and Development*, to appear, 2005.

[15] Axel Jantsch and Hannu Tenhunen. *Networks on Chip*. Kluwer Academic Publishers, 2003.

[16] T. Meincke, A. Hemani, S. Kumar, P. Ellervee, J. Oberg, T. Olsson, P. Nilsson, D. Lindqvist, and H. Tenhunen. Globally asynchronous locally synchronous architecture for large high-performance asics. In *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems 1999 (ISCAS'99)*, volume 2, pages 512 –515, June 1999.

[17] Mikael MillBerg, Erland Nilsson, Rikard Thid, and Axel Jantsch. Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip. In *Proceedings of the Design, Automation and Testing in Europe Conference (DATE'04)*. IEEE, 2004.

[18] Jens Muttersbach, Thomas Villiger, and Wolfgang Fichtner. Practical design of globally-asynchronous locally-synchronous systems. In *Proceedings of the Sixth International Symposium on Advanced Research in Asynchronous Circuits and Systems, 2000 (ASYNC 2000)*, pages 52–59. IEEE Computer society, April 2000.

[19] Umit Y. Ogras, Jingcao Hu, and Radu Marculescu. Key resarch probkems in noc design: A holistic perspective. In *Proceedings of the International Conference on Hardware/software Codesign and System Synthesis, 2005.*, 2005.

[20] Matteo Dall Osso, Gianluca Biccari, Luca Giovannini, Davide Bertozzi, and Luca Benini. Xpipes: A latency insensitive parameterized network-on-chip architecture for multi-processor SoCs. In *Proceedings of the 21st International Conference on Computer Design (ICCD03)*. IEEE, 2003.

[21] Andrei Radulescu, John Dielissen, Kees Goossens, Edwin Rijpkema, and Paul Wielage. An efficient on-chip network interface offering guaranteed services, shared-memory abstraction, and flexible network configuration. In *Proceedings of the 2004 Design, Automation and Test in Europe Conference (DATE'04)*. IEEE, 2004.

[22] E. Rijpkema, K. G. W. Goossens, A. Radulescu, J. Dielissen, J. Van Meerbergen, P. Wielage, and E. Waterlander. Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip. In *Proceedings of the Design, Automation and Test in Europe Conference (DATE'03)*, pages 350–355. IEEE, 2003.

[23] Jens Sparsø and Steve Furber. *Principles of Asynchronous Circuit Design - a Systems Perspective*. Kluwer Academic Publishers, Boston, 2001.

[24] Wolf-Dietrich Weber, Joe Chou, Ian Swarbrick, and Drew Wingard. A quality-of-service mechanism for interconnection networks in system-on-chips. In *Proceedings of the Design, Automation and Test in Europe Conference, 2005*, pages 1232–1237. IEEE, 2005.

[25] Drew Wingard. Micronetwork-based integration for socs. In *Proceedings of the 38th Design Automation Conference (DAC)*, pages 673–677. ACM, June 2001.